Web Engineering as a Specialization of *Software Engineering*: Differences in Project Management Education

Herwig MAYR
Department of Software Engineering
Upper Austria University of Applied Sciences
Hauptstr. 117, A-4232 Hagenberg, Austria

ABSTRACT

We present the motivation and our concept of introducing "Web Engineering" as a specialization of our "Software Engineering" curriculum. Our main focus lies on the differences in project management education for both areas as well as the necessary process models and tools.

First we discuss the principal differences of software project management and web project management, focusing on the main difficulties of teaching such management skills to primarily technophile students. Then we analyze the composition of modern software development teams and changes within such teams implied by the development of web applications. We illustrate this transition showing how a merely document-driven process – as can be found in many traditional software development projects – is turned into a highly tool-supported, agile development process, which is characteristic for web development projects.

This paper is based upon [17], whose contents have been updated and extended.

Keywords: web project management, software project engineering, undergraduate university education

1. THE RISING IMPORTANCE OF WEB ENGINEERING IN A SOFTWARE ENGINEERING EDUCATION

After offering a degree program for software engineering at the University of Applied Sciences in Hagenberg, Austria, for more than ten years, we are now in the transition from the classical (European) diploma-engineer system towards the bachelor/master system. In the frame of this transition we have restructured our curriculum, allowing software engineering sophomores to choose from three areas of specialization,

1. business software,
2. medical software, and
3. web engineering.

Whereas the first two specializations originate from previously separate degree programs now integrated into the software engineering curriculum, the "web engineering" specialization is a reaction to the changing market of web application development. Whereas early web sites mainly consisted of static hypertext pages, modern web applications are full-size software systems that include a highly complex, frequently distributed, control logic as well as comprehensive database access. However, many web-aimed companies still develop such applications in the very same style as they did with static hypertext pages. Implementing the established software engineering knowledge in the area of web application development must, therefore, be in the core focus of any modern, market-oriented software development education.

2. MANAGING SOFTWARE PROJECTS VERSUS MANAGING WEB PROJECTS

Management Objectives

Software project management enables an engineering-style software development through extending the technical product development cycle (plan – produce – check) with the economical and social tasks of management, development, and monitoring [16]. Thus, software development becomes an iterative, feedback-controlled process that includes a controlled, continuous adaptation of the orientation towards the objectives (see Figure 1). Software project management therefore combines the technical development of software with its economical production.

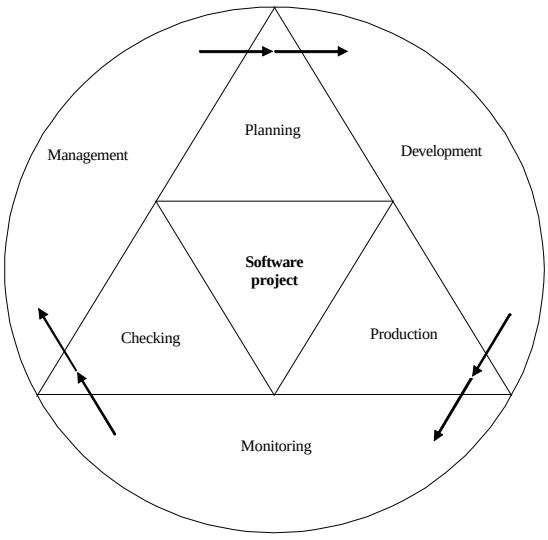

Figure 1: Project Management Objective: An Engineering Approach to Software Development

Distinguishing Web Projects from Software Projects

Generally one can observe that many monolithic software applications of former times are being replaced by a number of highly interacting, small web applications [25]. This trend ne-

cessitates shorter development cycles, reducing the necessity to develop software from scratch according to specified requirements. Instead, components are combined following an agile process and a – hopefully – useable design is being created "on the job" by means of refactoring [26]. This circumstance leads to a different characteristics for web project management from general software project management, as detailed in Table 1 (adapted from [25]).

Many young developers are not familiar with traditional models and methods that ensure and increase development maturity (such as CMMI or ISO 15504). And time to learn and apply these models is frequently not available. Process development, discipline, or estimation skills are typically shed as unnecessary ballast.

Web projects differ from traditional software projects in their results, too:

- Traditional software systems are comprised of parts grouped by functions, where the key metric of these parts is functionality. In contrast, software functionality and content depend on each other in web applications, and the joint availability of both elements is essential from the very first delivery on.
- The design and the creation of the content are at least as important as the application's functionality. For web applications, the structuring into design components is done in different ways by the different development communities, using different naming.

As mentioned in the literature (e.g., [8]), these areas have to be coordinated and – ideally – developed jointly. While information design aims at the content, interface design deals with user interaction and navigation in the web application. Program design comprises the functionality and communication with the application in the backend (databases, data warehousing systems, etc.). The main objective of web project management is to optimally match the presentation of information, access, and functionality of a web application, and coordinate all these areas with the content from the product perspective.

3. GENERAL CHALLENGES OF SOFTWARE PROJECT MANAGEMENT

Conventional project management in traditional software projects is confronted with challenges in all kinds of management. These challenges also apply to the development of web applications, as described in the following subsections.

Leadership Challenges

- **Unique software systems**: Software systems are frequently developed from scratch. The experience drawn from past projects is too limited to be able to make reliable cost estimates. web project management counters these challenges by a much higher degree of reusability and reuse.
- **Extremely technical leadership perspective**: Project management has been dominated by technology freaks, particularly technology-minded engineers. In many cases, this has led to a neglect of organizational development in favor of software development. In addition, engineers tend to plan overly optimistic. This attitude is often "benevolently" supported by marketing and sales people. web project teams are much more heterogeneous and less technophile. However, this doesn't necessarily mean that they are more experienced in project management, and it can cause other problems within the group.
- **Poor planning**: Many software products are characterized by unclear or incomplete planning objectives, frequent changes to the planning objectives, and deficiencies in the project organization. Compared to traditional software development, these problems arise even more frequently in the development of web applications, as we will discuss in the next section.

Development Challenges

- **Individuality of programmers**: Even today, many software development projects are seen as an art rather than a technique. Software programmers are individualists and their performance differs a lot. This is the reason why it is particularly difficult to estimate the actual manpower needed. Moreover, it is difficult to put individualists into an organizational straight-jacket. This problem arises especially due to "artists" in web teams, because their creativity is subject to a high degree of individuality.
- **The high number of alternative solutions**: In software development, there is virtually an unlimited number of alternatives to solve a specific problem. In many cases, it is impossible to compare and evaluate different solutions in advance. This problem is slightly smaller in the creation of web applications, because many components and semi-finished products can be used, but it shouldn't be underestimated.
- **Rapid technological changes**: The rapid technological development of hardware and software makes it more difficult to plan and organize software projects. It often happens that, while a large software system development is under way, new and better performing components (e.g., enhanced graphics functionalities) enter the market. This means that novelties introduced to the market while a project is in the works can make the system conception appear outdated and require a change to the design, making the plan obsolete. On the other hand, it can mean that new software tools become available, and their benefits are hard to tell. This problem is typical for web projects.

Monitoring Challenges

- **The immateriality of software products**: The "intangibility" of software products makes them hard to control. It is very difficult to determine how much of a software product is actually completed, and the programmer has a wide range of possibilities to veil the actual development state. Since web projects are characterized by parallel development of functionality and content, the product is more "tangible" for customers and the project manager. And since web projects are subject to short iteration cycles, they are usually easier to check. For these reasons, this challenge is of lesser significance in web projects.

In addition to these challenges that have to be dealt with in any kind of software development, the environment and restrictions in the development of web applications lead to particular difficulties and challenges. The challenges posed to web project management in dealing with these characteristics are discussed in the following section.

4. SPECIFIC CHALLENGES FOR WEB PROJECT MANAGEMENT

Process Aspects

The following properties are characteristic for the *development* of web projects and therefore constitute major challenges for web project management education [18]:

- **Novelty**: Web applications frequently address new, unknown groups of users. Therefore, it is very difficult to gather user requests and requirements prior and during the development [24]. New and changed requirements have to be tackled during a web project at an even higher rate than in ordinary software projects.
- **Dynamics**: Many developments of web applications are characterized by a high time-to-market pressure and, consequently, very short development cycles ([19] found the average duration of a web project to be 3 months). The main reasons for this time pressure are the quick changes in and of the web together with the high pressure of competition, the market being an expanding market and supplanting market at the same time [13].

Parameter	Software Project Management	Web Project Management
Objective	Create a quality product at minimum cost	Create a usable product as quickly as possible
Project Size	medium to large (10 to 100+ people)	generally small (6 +/- 3 people)
Length	12 to 18 months	3 - 6 months
Costs	several M €	several K €
Development Method	requirements based, phase oriented / incremental, document driven	agile methods, assembly of components, prototyping
Technology Used	object-oriented methods, CASE tools	component based methods, visual programming, multimedia
Processes	CMM, ISO etc.	ad hoc (“agile”)
Product	code-based, low level of reuse, complex applications	high level of reuse, standard components, many standard applications
Team Member Profile	professional software developers with several years of experience	multimedia designers, web programmers (Java, PHP, etc.), PR/marketing people

Table 1: Traditional Software Project Management vs. Web Project Management

- **Parallelism:** Web applications are frequently highly structured into components of the respective application domain (such components are, e.g., authentification, search tool, chatroom). Because of the short time-to-market period, many web applications are being developed highly parallel in sub-groups. It is the duty of the project management to assure that experts with similar domain knowledge communicate between the sub-groups and multiple developments of the same components are avoided. In [11], this duty is suitably compared with the task of a conductor of a chamber orchestra.
- **Continuity**: Objectives of web applications, the tools used for their development as well as the web itself are exposed to a continuous change. As a consequence, it is no longer meaningful for a web project to distinguish between a development phase and a maintenance phase. A web application is continuously maintained while being developed further at the same time [23]. Since a web application should be available to potential users all the time (“24x7 operation”), maintenance frequently has to be done on an up-and-running system, which makes it very difficult.
- **Juvenility**: On the average, developers of web applications are considerably younger and more inexperienced than the average software developer [25]. They frequently are self-taught programmers, who are very eager to absorb new technologies and tools, but quite ignorant to available knowledge and best practices. This often leads to a perplexing variety of tools and technologies used in parallel by the same organization unit, for which there is no other reason than developer preference. It is the task of the web project manager to utilize the enthusiasm of the team members for well-guided education and training and to impose a clear acquisition and update policy for technologies and tools.
- **Immaturity**: Many current web development environments are so immature that patching annoying errors, extending/adapting inadequate interfaces etc. are essential tasks in order to increase the productivity of the tool users. Many of these tools are only used due to the lack of alternatives and are dropped as soon as a more mature tool enters the market. As a consequence, nearly no web application can utilize the technology of its predecessor and development know-how gets lost or is not established at all. One hope for a solution to this problem is the rising availability of open-source tools from reliable sources (e.g., the GNU foundation or Apache), that have become interesting alternatives through the (development) support by the web community.

Product Aspects

The following properties are characteristic for the *use* of web applications and therefore constitute major challenges for web project management education [18]:

- **Complexity**: Early web applications, consisting mainly of static hypertext pages, were quite simple systems. Modern web applications have become full-fledged software systems that contain, additionally to the user interface, a complex control logic, connection to comprehensive data bases etc. But since they look similar to their simple ancestors because they are accessed via the same browser tools, many users (and even system suppliers) do not understand the much higher development effort and system resources needed.
- **Aesthetics:** The World Wide Web (or its applications, respectively) has been called “the most fashionable software application area” [23]. To remain en vogue with the web design is a key factor for success. This necessity of product alterations due to fashion trends increases the pressure for changes, That is already inherent in the web area due to its dynamics, even more.
- **Spontaneity**: The web is a very spontaneously used media. One cannot expect any loyalty from a user of a web application to its developer. [13] stress that customers use web applications only when the reward or satisfaction is immediate. Negative or even positive feedback is very rare.
- **Usability**: Users of web applications are even less willing to read online tutorials or printed manuals than users of

other software systems. Ideally, the use of a web application must be possible without any documentation! Good usability is a quality aspect of a web application that cannot be regarded enough at planning level.

- **Ubiquity**: Due to the boom of mobile devices, the web is not only accessible world-wide, but practically anywhere. This implies that the classes of actual users are extremely difficult to estimate, both in size and characteristics (just think of 56K modem access). Consequently, during the development of web applications it is generally impossible to access a representative selection of future users in order to elicit the requirements.
- **Compatibility**: Most web applications are accessed by the user via a browser tool. Although the market for such tools is dominated by very few products (Microsoft Internet Explorer, Netscape Navigator, Opera, to name the most important), the compatibility behavior of these products (and their various revisions still in use) is very divertive and certain standards (like HTML, CSS, Java) are supported at a broadly varying level. The creation of a tool with the same functional operability and user guidance independent from the chosen browser is very tricky, to say the least.
- **Stability**: Users of web applications expect to have accessibility around the clock, every day ("24x7 operation"). This implies a maximum level of reliability for the application, but also for the underlying hardware and network components. Additionally, maintenance of web applications is difficult, because it must be done either on the up-and-running system ("visible" to the user), or parallel and synchronously to the running system.
- **Scalability**: The ubiquity of web applications, together with the spontaneity of its users, requires web applications to be extremely scaleable, the amount of which being difficult to estimate during the development. Whereas a high level of scalability frequently means a general performance loss, low scalability means a drastic loss of performance when the maximum number of users/accesses per time unit is reached. Inappropriate scalability (of an e-banking system, for example) may lead to material damage (incomplete or incorrect financial transactions) as well as immaterial damage (loss of reputation).

5. SOCIAL ASPECTS OF MODERN SOFTWARE DEVELOPMENT TEAMS

Software Development: A Human-centered Task

Due to the rapid changes web application development has to deal with, and due to the fact that modern software developments are all managed by groups of people and no longer by individuals [16], the communication among the team members and their motivation and coordination by the project manager are among the most important success factors for a project. For this reason, software development is often called a *human-centered activity* [14].

Technical managers are particularly inclined to underestimate the psychological and social aspects in development teams. As soon as they have been identified, conflicts have to be addressed and resolved. In the field of technical development, there is often no room for compromise; there are "winners" and "losers", which leads to more conflicts. In the long term, however, it is unbearable for the staff and the project to try to avoid conflicts at any cost.

Characteristics of a Web Development Team

Teams that develop web applications are particularly characterized by the following three properties:

1. **Multi-disciplinary background**: Since every web application comprises content, hypertext structure and presentation [15] for a possibly very divertive audience, web developers must supply a broad variety of domain knowledge.
2. **Concurrent development**: Whereas in classical software projects tasks are subdivided from a developer's view (database, GUI, etc.), tasks within web projects are subdivided from a user's view. This results in sub-teams that are composed in a similar way regarding the domain expertise. Consequently, a frequent number of concurring parallel developments have to be coordinated and the communication overhead within web teams is generally higher than within "ordinary" software development teams. This type of communication is only very rudimentarily supported in classical software development environments and CASE tools [13].
3. **Small size***:* Due to the short development cycles and the generally rather small budget size of web projects, the size of web teams nearly always is low. [25] and [19] state that the average web team comprises 6 people and very rarely exceeds 10. Larger tasks are split and assigned to sub-teams working in parallel.

It is essential that each team member is completely aware of his taken roles and responsibilities. When responsibilities do overlap, it is the task of the team leader or – in case of several sub-teams – the project manager, to solve the resulting conflicts in the best suitable way for the overall project goal. Since web projects are characterized by short development periods, it is important to resolve conflicts quickly, even when from a global perspective this leads to a sub-optimal solution.

A special problem relates to assistance during operation and maintenance of a web application, which is particularly important in Web Engineering. All roles of a web project team are required for this assistance. Since the individual team members generally focus on other projects as soon as the development of a web application is completed, it is important to introduce a tool-supported, joint project management for parallel projects, a so-called *multi-project management*, in companies that develop web applications.

Profile of a Web Project Manager

The key facility of a web project manager, that distinguishes him from classical software project managers, is that he has to run a team of people with very different abilities and backgrounds. The team members are specialists with different education, skills, habits, and value scales.

It seems that any type of developer has problems in valuing the contributions from team members with a different education background [19]. On the other hand, experts of a certain domain show severe deficits in even only roughly estimating the size of tasks that are beyond their scope.

This behavior is not specific to web teams, as was experienced by the author, who once had to coach a group of professional game developers. Within this group, experts like game designers, asset managers, and programmers showed a similar conflict-prone behavior [29]. Frequently, it is the duty of the project manager to operate as a translator and mediator that has to translate not only the contents from one domain into the other, but also their value and motivation behind it.

Another important task of the web project manager is the assistance and integration of the customer during the development of a web application. There are two peculiarities to this task, compared to conventional software projects:

1. The transition of a web project from development to regular use is fluid. Also, it is frequently difficult for the web project manager to determine when a web application has been fully taken into operation, and thus when the actual development project has been completed and regular use (including maintenance) of the developed web application has begun.
2. In addition, it is often unclear whether or not a web project manager should still be involved with the project once the application has moved to the operation and maintenance phase. This becomes more critical by the fact that, due to the special knowledge of single members of a web project team, the main contact with the customer in web projects is not maintained through the web project manager, but directly through experts (see [7]). This fact could be another sign of the immaturity of web project management as currently practiced.

Table 2 lists the most important rules that should be considered by a web project manager in order to run a web development project successfully.

	Ten Golden Rules for the Web Project Manager
1	Encourage a professional attitude of each team member. Keep high ethics and morale within the team.
2	Emphasize the importance of different domain knowledge for the project.
3	Solve conflicts quickly. Not everyone can be a winner all the time. Take care that the loser(s) is/are not always the same person(s).
4	Continuously explain to each team member her/his role(s) and responsibilities.
5	Make evident parallel developments and use possible synergies.
6	Distribute documentation work related to the members' tasks and fair with respect to effort.
7	Encourage and coordinate the continuous usage of tools from the beginning of the project.
8	Translate effort scales and importance factors into the different project domains.
9	Urge the customer to be continuously involved with the project.
10	Keep an eye on project progress *and* project objective.

Table 2: Ten Golden Rules for the Web Project Manager

6. MODELS AND TOOLS FOR MANAGING MODERN SOFTWARE PROJECTS

Tailoring Documentation Models

Web applications are – inherently necessarily – developed using very flexible and adaptive processes, which are characterized by a high level of reuse, agile processes and close customer relationship through frequent deliveries of interim products [15].

Considering the used tools and generated (interim) products one typically finds a significant transition from document-driven processes found in traditional („rigorous") software development towards highly tool-supported processes in agile developments [21]. This transition of increasing tool support is illustrated in Table 3, using the document model of [16].

Focused in Rigorous Processes	**Of Same Importance**	**Focused in Agile Processes**
	Organization Chart, Roles	
	Project Library, Diary	
	Protocols	
Progress Reports		Interim Products, Prototypes
	Configuration Management (**Tool!**)	
	Quality Characteristics	
	Objectives Report	
Requirements Specification		Requirements List (**Tool!**)
	Project Plan	
Master Plan		Strategic Idea, Operative Plan (**Tool!**)
		Risk Management
User Manual		Interim Products, Online Help
System Specification		Design Model (**Tool!**)
		Continuous Modeling Language
	Source Code	
	System Documentation	Application Data
	Executable Code	
		Test Plan, Test Suite
Error Reports, Error Log		Error Management (**Tool!**)
	Installation & Acceptance Protocol	
	Final Report Maintenance Plan	
	Project Archive	

Table 3: Increased Tool Use in Agile Process Models

Tailoring Tools

Agile methods show an increased need of tools for the following tasks [18]:

- requirements management,
- planning,
- design / implementation,
- test management (planning, execution, and reporting),
- (integrative) configuration management.

Tools are particularly necessary for younger, less experienced developers, who otherwise may easily lose control over a highly iterative development process. In [9], one of the very few studies that explicitly deals with management problems of web projects, it is stated that " … without tools for measuring progress, managing changes, and verifying results … [as well as] … careful tracing of the requirements both by the developer and the customer iterations may turn into ad-hoc-development" [9]. The same study stresses that for web projects, too, development must be based on a documented plan, and careful, comprehensive testing is essential, even when a project is already running late.

A good combination (integration, if possible) of tools is critical for increasing the efficiency of a development (cf. the *Eclipse* project, http://www.eclipse.org). However, one must care that the chosen development process must be independent from the used tools and technologies [19]. Since the technology for developing web applications changes so rapidly and unpredictably, only a process can be durable that is as clearly as pos-

sible separated from the used implementation tools, technologies, and languages.

When transiting towards agile methods, people frequently overlook the fact that the necessary tools (cf. Table 3) not only must be available, handling these tools must be learned and trained. The time that is necessary to get familiar with the selected tools is frequently not reserved within the tight schedule of rather short web projects. Also, it is difficult to justify this effort for a planned, coordinated training in the frame of a single project. As a result, the – frequently inexperienced – developers tend to teach agile methods and the necessary tools themselves, which is a proven way to omit every "overhead" that is individually considered as unimportant, and hacker mentality enters the stage [4].

Since the web is obviously familiar to web developers, web-based tools are an ideal basis for web project management. Web-based project management tools like, e.g., *eProject* (http://www.eproject.com) enable classical project management tasks (like keeping time logs and task logs, storage and versioning of product documents, protocols, etc.) to be easily done via the internet. Additionally, blackboards, chat rooms, ICQ, e-mail dispatchers, etc. considerably increase communication within a web team, even when it is located distributedly.

Table 4 summarizes recommendations for an efficient and effective tool use in the frame of web projects.

	Recommendations for Effective Tool Use
1	Clearly separate the development process from tools, models, and languages used.
2	Select tools that can be integrated and used throughout the whole development process.
3	Start using tools as early as possible. A later capturing or conversion of the models is cumbersome and unsatisfactory.
4	Use processes and tools that efficiently support iterative, evolutionary development and easily integrate customer feedback.
5	Reserve enough time for training and getting acquainted with each tool.
6	Prior to each tool change or release change of a tool, check its necessity and consequences.
7	Do not only measure project progress, but also the degree of objective coverage.

Table 4: Recommendations for Effective Tool Use

Managing Configurations

One of the most essential tools for guaranteeing a well-structured progress of a web project is a *configuration management system* [10]. Due to the short iteration cycles of web projects, configuration management tools are mainly used in the frame of web engineering for

- *managing versions* of the source code and the application content together with its access policy,
- *creating configurations* of, both, source code and application content, in order to guarantee a well-organized release policy,
- *administrating* requirements *changes* as well as defect and error requests,
- *monitoring* the status of the project *documents* in order to determine the project progress.

Variants (i.e. branches in the development tree) are created rather rarely in web engineering. One situation where variants may occur is to finish a certain version for the customer in an undisturbed way, while – due to the short iteration cycles – other members of the development team realize the current product progress in a separate product branch.

Many web projects start quite small in size and continuously grow into a quite comprehensive endeavor later on. Although for this reason most projects are too small for a comprehensive tool-based configuration management at their initial phase, it is crucial to use configuration management tools from the very beginning of a project. A later transition implies an enormous effort and loss of time; also the development history will not be documented prior to the introduction of the configuration management tool. Particularly when a web project is subdivided into a number of small sub-projects (as discussed in Section 4), whose results frequently have to be integrated into (interim) products, a homogeneous configuration management tool will soon become an indispensable part of a web development environment [3].

Measuring Progress

During the development of web applications, frequently only two documents are created [19],

1. the *system specification*, containing the results of the requirements elicitation together with the main design decisions, and
2. the *web application* itself in its respective states of completion.

As a consequence, a web application is frequently generated as a quick sequence of interim results using highly iterative, evolutionary prototyping. [27] observe that the duration of the single iterations should be as short as reasonably possible, focusing on a clear definition of functionality to be expected. After each iteration a review should take place, incorporating the customer whenever possible.

This process is identical to "Rapid Application Development" (RAD), a strategy that has been known to and used in classical software engineering for almost 20 years [30]. What distinguishes the development of web applications, however, is that the requirements of a web application are much more difficult to define prior to its development compared to standard software systems, making the estimation of project size and project costs a high-risk gamble [25]. Additionally, frequently high pressure is put onto a specific, short-term delivery date, which implies that it makes much more sense to "… estimate the quality of a time-fixed product instead of the costs of a well-specified system." [22]

A list of the most important characteristics for estimating project progress is given in Table 5, comparing web engineering to traditional software engineering. This list has been adapted and extended from the work of D. Reifer [25], who used these criteria for developing his own "web project metrics". This metrics regards the use of web-specific building blocks (cookies, ActiveX controls, COM-components, etc.) and high percentage of reuse within web projects in the following way: Analogously to the "software objects" of the COCOMO II model [5], Reifer defines so-called "web objects" which allow the transfer of the concept of "object points" (as used in COCOMO II) to web components [25]. The weight functions, however, have to be considerably altered. Alternative approaches to develop metrics suitable for web development can be found in [20].

Clearly defined counting rules for web objects are crucial for good estimations using these approaches – like with any

other metrics. The major problem that remains, however, is that design components are counted which are developer-related instead of counting customer-relevant entities. Project progress can be measured quite well using this approach. Whether it leads to an increase of the degree of goal coverage – being the only measure of ultimate relevance – remains an open question, however.

Criterion	Traditional Software Engineering	Web Engineering
Estimation Process	analogies, experience	structuring by developers
Measurement	from requirements	projection of frameworks
Measure	SLOC, FP, OP	- (varying)
Development Effort	cubic root relation [5]	cubic root too high, square root?
Calibration	experience, similar projects	- (varying)
Risk Estimation	quantitatively	qualitatively (no models available)
ROI	estimation models	- (varying)

Table 5: Estimation in Traditional Software Engineering and Web Engineering

7. NO LIGHT AT THE END OF THE TUNNEL?

In the field of software development methods the discussion of "agile" vs. "rigorous" process models has more or less started a religious war (cf. [21]), which has in the meantime swept over to project management, too. [28] for instance offers an "extreme project management", which at closer inspection turns out to be a very conventional approach. On the other hand, techniques like Scrum [9, 21] show that the transfer of Extreme Programming ideas to project management can be quite fruitful, particularly when the concepts are not adopted too radically.

A further trend is set by closely inspecting proven managing approaches and generalizing these into "best practices". Even the "pattern" terminus, originally coined in the design and analysis areas of software development, has been transferred to project management [1, 2] and is being complemented by collections of examples of negative management performance ("anti-patterns", cf. [6]).

When considering the success statistics of software projects, at first glance the results are disillusioning. Publicly accessible statistics by various institutions monitoring the results of software development projects (mainly within the U.S.A.) are presented in Table 6. The summarized data are based upon general software projects (based upon several 100,000 software projects, according to the investigating institutions). Up to now, there are no large-scale investigations known to the author that are specific to web projects, with the exception of [9]. This report, however, is also partially based upon data from general software projects and indicates findings specific to web projects only for certain aspects.

The results show (cf. Table 6) that during the Mid-Nineties as well as at the beginning of the new Millennium, only approximately 25 % of the investigated projects have been classified as successful. 75 % of the projects ran considerably over time and/or over budget (on the average by 70 %!), or were canceled completely.

Investigations on the Success of Software Projects in the U.S.A.	Project successful	Project over Budget or over Time	Project canceled
Standish Group (1994) (http://www.standishgroup.com)	16 %	53 %	31 %
Center for Project Management (1995) (http://www.center4pm.com)	25 %	50 %	25 %
Standish Group (2000) (http://www.standishgroup.com)	28 %	49 %	23 %
Cutter Consortium (2000) (http://www.cutter.com)	16 %	63 %	21 %
Gartner Group (2000) (http://www.gartner.com)	24 %	51 %	25 %
Standish Group (2004) (http://www.standishgroup.com)	29 %	53 %	18 %

Table 6: Development of the Success Statistics of Software Projects in the U.S.A.

At first glance, these results do not shine a bright light onto software project management and its development during the last decade. However, one must take into consideration that during this time the size of software projects continuously grew bigger and the imposed constraints (deadlines, resources) became increasingly restrictive. Without an improvement in project management, it is safe to assume that the success rate of software projects would have dropped during the last years as a consequence.

The available data implies that, during the next years, there will not be much of a change with respect to the success rate of software projects. Regarding the increasing aggravation of constraints, paired with the exploding evolution of the available technology, this development should not be considered stagnation but success, particularly in the area of web project management.

Acknowledgements

The author would like to thank his colleagues DI Hans Heinzelreiter and DI (FH) Peter Kulczycki for many valuable discussions on the subject and Dr. Christoph Steindl from IBM Austria for his comments on agile project management.

8. REFERENCES

[1] S.W. Ambler, **Process Patterns: Building Large-Scale Systems Using Object Technology**, Cambridge University Press, 1999.

[2] S.W. Ambler, **More Process Patterns: Delivering Large-Scale Systems Using Object Technology**, Cambridge University Press, 1999.

[3] R. Baskerville, L. Levine, **How Internet Software Companies Negotiate Quality**, IEEE Computer, pp. 51-57, May 2001.

[4] K. Beck, P. McBreen, **Questioning Extreme Programming**, Addison-Wesley Longman, 2002.

[5] B.W. Boehm, C. Abts, A. Winsor Brown, A. Winsor Brown, **Software Cost Estimation with COCOMO II**, Prentice-Hall, 2000.
[6] W.J. Brown, H.W. McCormick, S.W. Thomas, **Anti-Patterns in Project Management**, John Wiley & Sons, 2000.
[7] J. Burdmann, **Collaborative Web Development: Strategies and Best Practices for Web Teams**, Addison-Wesley, 1999.
[8] S.S. Chan, **Teaching Team Projects in E-Commerce Development Courses: Design, Management, and Assessment Issues**, Proc. 17th Annual Conf. of the Intl. Academy of Information Management, 2002.
[9] Cutter Consortium, **Poor Project Management Number-One Problem of Outsourced E-Projects**, Cutter Research Briefs, Nov 7, 2000, http://www.cutter.com/research/2000/crb001107.html.
[10] S. Dart, **Configuration Management: The Missing Link in Web Engineering**, Artech House, 2000.
[11] A. Friedlein, **Web Project Management – Systematic Approach to Planning, Realizing, and Maintaining Web Sites**, dpunkt.verlag, 2002.
[12] J. Highsmith, **Agile Software Development Ecosystems**, Addison-Wesley, 2002.
[13] J. Holck, T. Clemmensen, **What Makes Web Development Different?**, Technical Report, Dept. of Informatics, Copenhagen Business School, Denmark, 2002.
[14] W.S. Humphrey, **Managing Technical People**, Addison-Wesley, 1997.
[15] G. Kappel, B. Pröll, S. Reich, W. Retschitzegger, **Web Engineering: Systematic Development of Web Applications** (in German), dpunkt.verlag, 2003.
[16] H. Mayr, **Software Project Engineering: Software Development in Project Teams** (in German), 2nd edition, Fachbuchverlag Leipzig / Carl Hanser Verlag, 2005.
[17] H. Mayr, **Project Management Education for Software Engineering and Web Engineering: Commonalities and Differences,** In F. Malpica, ed.: Proc. Intl. Conf. on Education and Information Systems: Technologies and Applications (EISTA 2004), pp. 277–282, IIIS/IFSR, Orlando, USA, 2004.
[18] H. Mayr, **Web Project Management** (in German), In G. Kappel et al., Web Engineering, Chapter 9, dpunkt.verlag, 2003.
[19] A. McDonald, R. Welland, **A Survey of Web Engineering in Practice**, Department of Computing Science Technical Report R-2001-79, University of Glasgow, Scotland, 2001.
[20] L.A. Olsina, G. Lafuente, O. Pastor, **Towards a Reusable Repository of Web Metrics**, Journal of Web Engineering, pp. 61-73, vol. 1 no. 1, 2002.
[21] K. Orr, **CMM versus Agile Development: Religious Wars and Software Development**, Cutter Executive Report #3.7, Cutter Consortium, 2002.
[22] R.S. Pressman, **Can Internet-Based Applications Be Engineered?**, IEEE Software, pp. 104-110, vol. 15, no. 5, 1998.
[23] R.S. Pressman, **Can WebApps Be Engineered?**, SPC Essentials, Software Productivity Center Inc., Mai 2000.
[24] A. Rahardjam, **Designing Interactivity: Who Are the Users and What Are the Techniques?**, Proc. 5th Australian World Wide Web Conference, pp. 119-128, Southern Cross University Press, Lismore, Australia, 1999.
[25] D.C. Reifer, **Estimating Web Development Costs: There Are Differences**, CrossTalk – The Journal of Defense Software Engineering, pp. 13-17, June 2002.
[26] K. Schwaber, M. Beedle, **Agile Software Development with Scrum**, Prentice Hall, 2002.
[27] M. Sharma, **E-Business – Building It Right From the Ground Up**, Cutter IT Journal, pp. 30-35, vol. 14, no. 1, 2001.
[28] R. Thomsett, **Extreme Project Management**, Cutter Executive Report # 2.2, Cutter Consortium, 2001.
[29] K. Wallner, **Asset Management in Game Development**, Master's Thesis, Department of Software Engineering, University of Applied Sciences Hagenberg, Austria, July 2001.
[30] E. Yourdon, **Modern Structured Analysis**, Yourdon Press / Prentice Hall, 1989.