MAGISTERARBEIT

Titel der Magisterarbeit

Evaluation of Software Development Paradigms and Processes for Web Application Engineering

Verfasser

Thomas Lichtenecker, Bakk.rer.soc.oec.

Angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften (Mag. rer.soc.oec.)

Wien, 2011

Studienkennzahl lt. Studienblatt:	A 000 926
Studiengebiet lt. Studienblatt:	Wirtschaftsinformatik
Betreuerin:	o.Univ.-Prof. Dr. Gertrude Kappel

Erklärung zur Verfassung der Arbeit

Thomas Lichtenecker
Gertrude-Wondrack-Platz 2/3.10, 1120 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit — einschließlich Tabellen, Karten und Abbildungen —, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Kurzfassung

Die Diplomarbeit beschäftigt sich mit dem Fachgebiet des Software Engineerings. Im Speziellen wird hierbei auf das Gebiet des Web Engineerings eingegangen. Das Hauptziel ist es, festzustellen, ob ein signifikanter Unterschied zwischen der Entwicklung klassischer Software (wie etwa für den Desktop) und der von Web-Applikationen besteht. Weiters werden sowohl agile, als auch traditionelle Entwicklungs-Methotologien auf ihre Anwendbarkeit auf den diskutierten Einsatzzweck hin untersucht.

Um das allgemeine Verständnis zu gewährleisten, gibt die Arbeit einen kurzen Einblick in die Technologien, auf denen das Web aufbaut — wie etwa HTTP und HTML. Darauf folgt eine Einführung in die Disziplin des Web Engineerings anhand von relevanter Literatur. Dieses Thema wird weiter vertieft, indem eine Kategorisierung anhand der Faktoren Zeit und technischer Komplexität vorgenommen wird. Ferner werden die Charakteristiken des Web Engineerings anhand der Perspektiven des Entwicklers, des Users und des Managements betrachtet. In einem weiteren Schritt wird untersucht, ob gängige Software Engineering Methodologien auf Web basierte Entwicklung anwendbar sind. Zu diesem Zweck werden drei bekannte Methodiken im Detail präsentiert: der Rational Unified Process, Scrum und Extreme Programming (XP).

Um sie auf ihre Andwendbarkeit hin zu beurteilen, werden diese Methodiken kategorisiert und nach ihren Stärken, Schwächen, sowie bezüglich ihrer Abdeckung des Entwicklungsprozesses bewertet. Nach dem Abwägen dieser Information im Verhältnis zu jenen Charakteristiken, die Web Engineering auszeichnen, stellt die Arbeit noch einen Risiko-basierten Ansatz vor, welcher die Auswahl eines Entwicklungs-Ansatzes anhand diverser Faktoren ermöglicht.

Basierend auf den Erkenntnissen, die aus dem theoretischen Teil gezogen werden können, konzentriert sich der Implementiertungs-Teil der Arbeit darauf, diese auf ein Beispiel aus der Praxis anzuwenden. Im Besonderen wird auf die Kommunikationsbedürfnisse eines Teams eingegangen, welches Web Projekte nach agilen Gesichtspunkten abwickelt. Konkret wird die Entwicklung einer Aufgaben-Verwaltungs-Lösung basierend auf dem Microsoft Team Foundation Server beschrieben. Hierfür werden bestehende Templates extensiv angepasst und erweitert um alle benötigten Anwendungsfälle abzudecken.

Abstract

This thesis deals with the process of software development in general and Web engineering in particular. Its main purposes are to determine whether there is a significant difference between developing classical (desktop-centric) software and applications being distributed via the World Wide Web (WWW) and to introduce agile and traditional software development methodologies to evaluate if using these is valuable for the discussed purpose.

To enhance common understanding, the thesis gives a brief introduction into the technologies behind the Web, such as HTTP and HTML. This leads to a general introduction of Web engineering as it is described in the relevant literature. Web engineering is then further categorized according to complexity and technological novelty and common characteristics affecting users, management, and development. Further, this thesis examines if well known software development paradigms are suitable for Web centric development, or if new solutions are needed. To support this, three well-known development methodologies are presented in detail — the Rational Unified Process, Extreme Programming, and Scrum.

Consequently, the strengths, weaknesses, and process coverage of all three processes are discussed in order to judge on their applicability. The processes are then weighed against the Web engineering characteristics found in earlier parts. Based on this information and by combining it with an established risk driven approach, a recommendation for choosing the right methodology for the different engineering categories is provided.

The practical part concentrates on the insights gained in the theoretical chapters and describes the support of a Web engineering process in a real live work environment by leveraging the capabilities of Microsoft's Team Foundation Server. Here the work focuses on the communicative part of the process and heavily customizes the standard templates in order to create a central communication tool. This tool acts as a central communication hub providing a unified interface for all departments.

Contents

Chapter 1

Introduction

The Internet in all its forms provides people with many great benefits and opportunities. The global nature of this medium allows worldwide communication, collaboration, and sharing of ideas. It provides consistent access to widely distributed information and has radically changed the way individuals are accessing it. This environment has introduced new challenges to the field of software engineering and thereby creating a separate discipline *Web Engineering*. In order to facilitate reuse of existing software engineering methodologies, this thesis concentrates on related aspects, introduces characteristics that set Web engineering apart, describes standard approaches to software engineering, discusses their applicability and, finally, showcases a customized software solution supporting the Web engineering process.

1.1 Problem Description

Web applications adhere to the same criteria of software quality as every other software product — such as maintainability, usability, and efficiency. Nevertheless, questions arise about the extent that Web applications fall into these criteria, and if there are additional criteria to describe this type of software. Highlighting possible differences, the term Web engineering was first published in 1997 [1]. Though younger publications exist [2, 3], the amount of academic literature is limited and often focuses on the early phases of what is described as a young discipline [4]. Acknowledging the growing possibilities of using Web technologies requires researching the characteristics and categories [5] of Web engineering, especially in comparison with standard software development.

When looking for differences, it is necessary to look at this matter from different viewpoints. Focusing on the people involved, running Web projects usually requires expertise in media design and content creation in order to attract potential users *on first sight*. Both disciplines are typically not that relevant for other software projects.

Incorporating them into the development process creates new relationships between other project stakeholders like management, development, and the customer.

Another important aspect of Web offerings is that if not updated regularly, they quickly become outdated. This leads to rapidly evolving/changing requirements. Though collecting requirements early in the project is still important, features change over time leading to constant adaptations to the code base and other assets. This has consequences for code, processes, and tools. Regarding source code, maintainability becomes a very important factor. In order to push updates live quickly, rapid deployment and release practices are required [6]. This requires appropriate tool support.

It is not only the products themselves that evolve quickly, but also the technologies that are used to build them. This requires a very adaptive developer base which consequently makes specialization very likely to happen. This is facilitated by the existing ways of clearly separating code, design, and content [7].

A typical Web application is not limited to only a small user base. Instead, the vast number of potential users makes it hard to determine specific user requirements. This, however, is a key aspect, as users have individual preferences and significantly varying domain knowledge. Not only do cultural factors have to be taken into account but also technical conditions, such as different screen sizes (especially mobile devices) and bandwidth influence the application design.

Today, software engineering is covered by a huge range of development methodologies. The concept of writing software in a structured manner was first widely discussed at the 1968 NATO Conference on Software Engineering [8]. Even before that, in 1956 H.D. Benington [9], and in 1961 W.A. Hosier [10] were considering approaches for the improvement of the development standards existing at that time. 1970 was an important year in the journey of software development, as Winston Royce invented a model that would later been known as the “Waterfall Model” [11]. Though Royce had already planned feedback loops between consecutive steps in his model, it was Barry Boehm’s “Spiral Model” of 1988 [12] that paved the way for iterative development and ultimately led to modern approaches like the Rational Unified Process. Putting an even greater focus on iterativity, *agile* methods [13] see change as a chance for better products. In contrast to what they describe as *heavyweight*, these methods concentrate on human beings and working code, instead of enforcing standards and process compliance. What unites all these approaches is a common belief that the software development process has to be structured in a certain way. A goal of this thesis is to determine what from this knowledge can be transferred to Web engineering.

1.2 Expected Result

After analyzing the special characteristics of Web engineering, the challenge is to decide which concepts, methods, techniques, and tools of traditional software engineering have to be adapted to the needs of Web engineering [5], and to find a base for choosing the right development method for the use case.

As mentioned, Web engineering requires short release cycles. This requires constant communication and tools that support the information flow. This thesis describes what has been done in a real world environment to streamline cross- and inter-departmental communication, and how this affects the adopted Web development process.

1.3 Methodical Approach

One major part of the thesis is to investigate what makes Web engineering a discipline in its own right. Two dimensions of categorization are established to identify which parts of the development team and which phases of the development process are affected. This is essential in order to later judge the usability of software development paradigms. The first dimension divides Web engineering into eight different categories determined by means of complexity and age (in terms of how long the category exists). The second classification aspect concerns characteristics [14] applicable to all categories. Here it is possible to identify three major perspectives: user, development, and management. Each role faces its own set of difficulties inherent to the application domain. This approach leads to a list of characteristics unique to Web engineering.

The main goal of applying a defined process to software development is to ensure software quality whilst meeting the agreed budget and timeframe. Web engineering, despite its organizational and technical specialities, still shares a lot of attributes with standard software development. Weighing software development against these factors and the specialities found depends on ascertaining which software development approaches exist and which practical models are based on them. The thesis provides an overview of existing models. With Extreme Programming (XP) [15] and Scrum [16, 17], the thesis presents two popular agile approaches, plus the Rational Unified Process (RUP) [18] as a more heavyweight, but nevertheless modern counterpart.

Subsequently, the thesis elaborates on both the differences and the similarities of the processes described to find a common set of attributes and thereby making sensible comparison possible. This part demonstrates which aspects of the development lifecycle the methods discussed focus on [19]. This indicates whether there is a single paradigm that fits all process requirements, or if a combination of aspects from different approaches is more suitable. Consecutively, a possible method of deciding on the right methodology is presented: the risk-based approach [20]. This approach states that risk should be the major driving factor when choosing a development approach, and that the more risk there is, the more heavyweight the process of choice should be. However, this leaves out the special Web engineering characteristics, so their effect on the process selection has to be illustrated and respected.

The practical part of the thesis showcases the application of the described theory to a real-world project. It outlines the development lifecycle that was found to be useful in this environment, and gives detailed insight into its main communication platform (which was extensively adapted in order to support the process). For this purpose, the requirements were discussed with a small group of stakeholders. After finalizing a beta version, the solution found was presented to and discussed with an extended user group. Their input was incorporated into the final product, and after a short test period the system was put into production. Since then, it has been improved on request. It has proven to be central for streamlining the communication efforts in a very agile environment, and thus nicely complements the lean structures of the given work environment.

Chapter 2

Web Engineering

The term “Web engineering” was first published in 1997 in the conference paper “WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle” [1]. Since that time Web applications have developed from a simple text based information medium into an application medium for fully-fledged and sophisticated software systems. This chapter provides information to help to understand the different aspects of Web engineering. Based on scientific research and practical experience, it gives insight into the field of Web engineering and explains the motivation behind this thesis.

In 2001, S. Murugesan described Web engineering as “an emerging new discipline, that advocates a process and a systematic approach to development of high quality Web-based systems” [4]. The paper further stresses the establishment and use of sound scientific, engineering and management principles, and disciplined and systematic approaches to development, deployment and maintenance of Web-based systems. To better understand what is covered by the term Web engineering, it is important to shed some light on the discipline’s history, its origins, as well as on the technological foundations. Consequently, this chapter further explains technological and organizational characteristics, and names those specific to the discipline discussed. This understanding allows weighing the features of existing software development methodologies against these peculiarities.

2.1 The World Wide Web

Tim Berners-Lee can be considered as the architect of the World Wide Web (WWW). In 1990, he not only designed its architecture but he also coined the accompanying Hypertext Markup Language (HTML) [21, 22]. Since then, the WWW has become an unavoidable part of our lives. Both ICANN [23] and other sources [24] conclude that about 1.5 billion people use the WWW. Over the last 10 years, the WWW has continued to grow at an exponential rate [2]. The July 2010 Netcraft report [25] counted 205,714,253 individual Websites. Figure 2.1 shows the growing number of Websites over time.

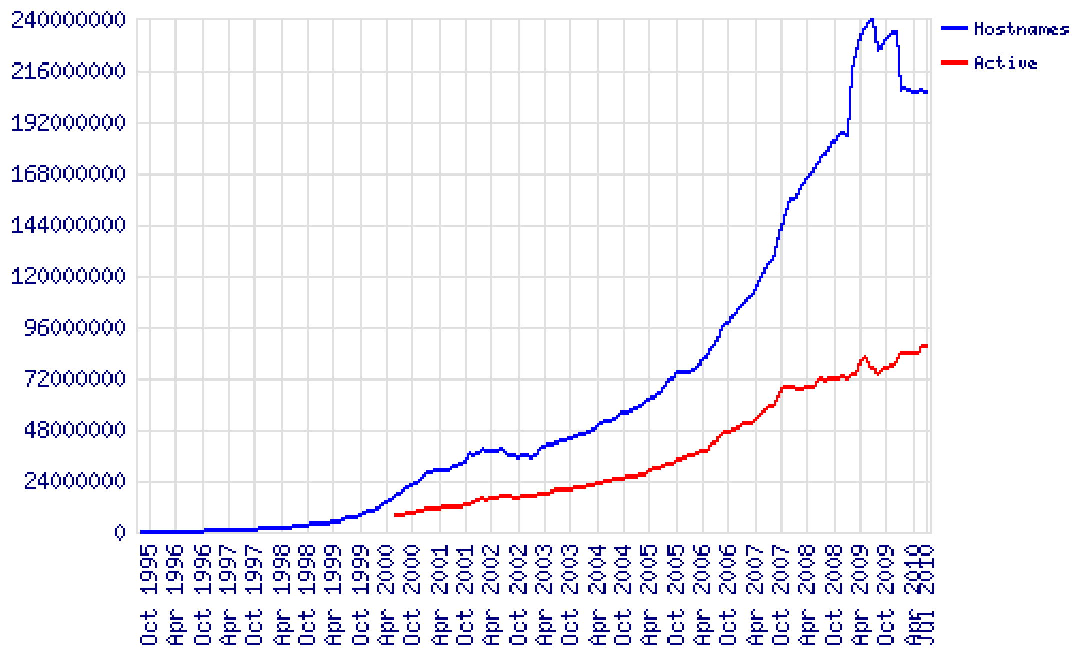


Figure 2.1: Number of Websites [25]

This development indicates the increasing influence of the Web on today's society. Although the Web is *big* in terms of numbers, it is maybe even *bigger* when it comes to its social and cultural influence. Banks, enterprises, governments, educational institutions, universities, and individuals depend on large-scale Web based applications to run their operations. Thanks to technological advances, more and more services can be provided over the Web. Recent developments in wireless technologies, such as UMTS or IEEE 802.11n, and advanced mobile devices allow these services to be used almost anytime and anywhere. As a consequence of this evolution, serious Web application development requires sound, scientifically backed engineering and management approaches in order to meet business requirements and aspects of software quality. This involves structuring the raw information to be delivered in a way that is meaningful, building a packaged presentation that is organized, aesthetic, ergonomic, interactive (where required), and delivering it to the Web client in a manner that initiates a conversation [14].

To introduce the characteristics of Web engineering to a broader audience, this section gives a basic understanding of the technologies the WWW is built on. It shows how these technologies interact and function with each other.

2.1.1 Basics

In the context of Web engineering, it is fundamental to know the three parts that a Web application is typically composed of:

- Client: typically a browser, performing the *request*
- Server: holding the Web application, performing the *response*
- Protocol: defining the rules of communication between the client and the server

According to Tim Berners-Lee's proposal for an information management system in March 1989 [22], the architecture of the hypertext world consists of data stored on server machines, and client processes on the same or other machines, both linked by a network. The WWW is built on top of the existing Internet structure and the protocols that come along with it. HTTP is the most important of them as network traffic generated by Web software is currently the largest single category of traffic on the internet, exceeding e-mail, file transfers, and other data transmission traffic [26].

2.1.2 Protocols

As the previous section mentioned, communication over the Web is regulated by a set of protocols. Two very prominent ones are HTTP and SMTP, with each of them being designed to serve different purposes.

HTTP - Hypertext Transfer Protocol. Probably the best-known protocol of the WWW is the Hypertext Transfer Protocol, or short HTTP. HTTP is a text-based protocol for distributed, collaborative, hypermedia information systems [27]. It is stateless and controls how resources such as HTML files are accessed. Typically exposed through port 80, HTTP builds on the TCP/IP stack. Together with TCP/IP, HTTP is one of the fundamentals of the WWW. HTTP, therefore, acts as a layer on top of the TCP/IP protocol as Figure 2.2 demonstrates.

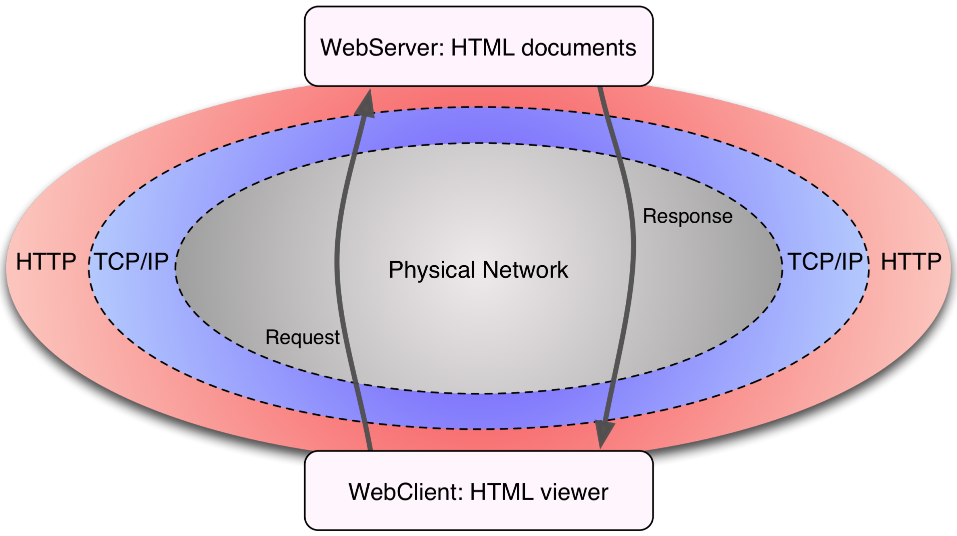


Figure 2.2: Functional view of the WWW [28]

The HTTP protocol owes its high profile to the nature of forming the prefix for a large number of URLs on the WWW. URL (Uniform Resource Locator) is the most prominent representative of what is called URI (Uniform Resource Identifier). Typically a URL may look like *http://www.example.org/path/resource.html*. This URL gives information about HOW (*http://*), and WHERE (*www.example.org/path/*) the resource is accessed, as well as its NAME (*resource.html*). In combination with the DNS (Domain Name Service), it is possible to assign resources unique and user friendly names. The combination of HTTP with the SSL/TLS protocol allows creating a secure channel via an insecure network. This is commonly referred to as HTTPS. HTTPS uses port 443 by default.

SMTP - Simple Mail Transfer Protocol. Proposed in 1982 [29], the Simple Mail Transfer Protocol (SMTP) is a standard for electronic mail (e-mail) transmission across networks. Today it is only used to send e-mails from a client to a mail server for relaying. Client applications usually use either the Post Office Protocol (POP3) [30] or the Internet Message Access Protocol (IMAP) [31] to retrieve e-mails from accounts on mail servers.

2.1.3 Document Representation

HTML - HyperText Markup Language. "The HyperText Markup Language (HTML) is an SGML application" [5]. SGML itself "is a system for defining markup languages" [32], and HTML is one sample implementation [33]. The first version of HTML was created by Tim Berners-Lee in December 1990 [7]. Since 1996, the HTML specifications have been maintained by the World Wide Web Consortium (W3C)[1], with input from commercial software vendors. In the year 2000, HTML became an international standard (ISO/IEC 15445:2000). The most recent HTML specification published by the W3C in 1999 is the HTML 4.01 Recommendation [32]. However, on January 22, 2008, the W3C has published the first public working draft of HTML 5. Although the specification has not reached its final recommendation status, parts of it are already implemented by major browser developers[2].

Session Tracking and Cookies. As Web applications interact with multiple users at the same time, they have to be able to distinguish between simultaneous requests to identify the ones coming from the same user. This client-server dialogue is called *Session*. As HTTP is stateless, the Web server cannot rely on an established TCP network connection for longer than a single HTTP operation. The most common way to overcome this challenge is to let the client store the session information (e.g., a session ID) and to send it with each request. This data is typically stored in *Cookies*. Cookies are little text files stored by the client's Web browser, holding key-value pairs of information. The cookie itself is generated by a Web server and included into the HTTP response header. The lifetime of a cookie can either be set to a certain duration, to be permanent, or to cover only the current session.

2.2 Categorization

With Web technologies advancing over time, applications steadily cover more complex requirements. This development allows the categorization of Web applications based on functionality. As Figure 2.3 shows, it is possible to define categories and to classify these in a two-dimensional scale, comprised of the two axes *Complexity* and *Development History*. The categorization presented in this document is based on existing literature [5, 3]. In addition to these classifications, the one described in this chapter adds current Web developments.

[1] `http://www.w3c.org`

[2] For example in Firefox: `http://blog.mozilla.com/standards/category/html5/`

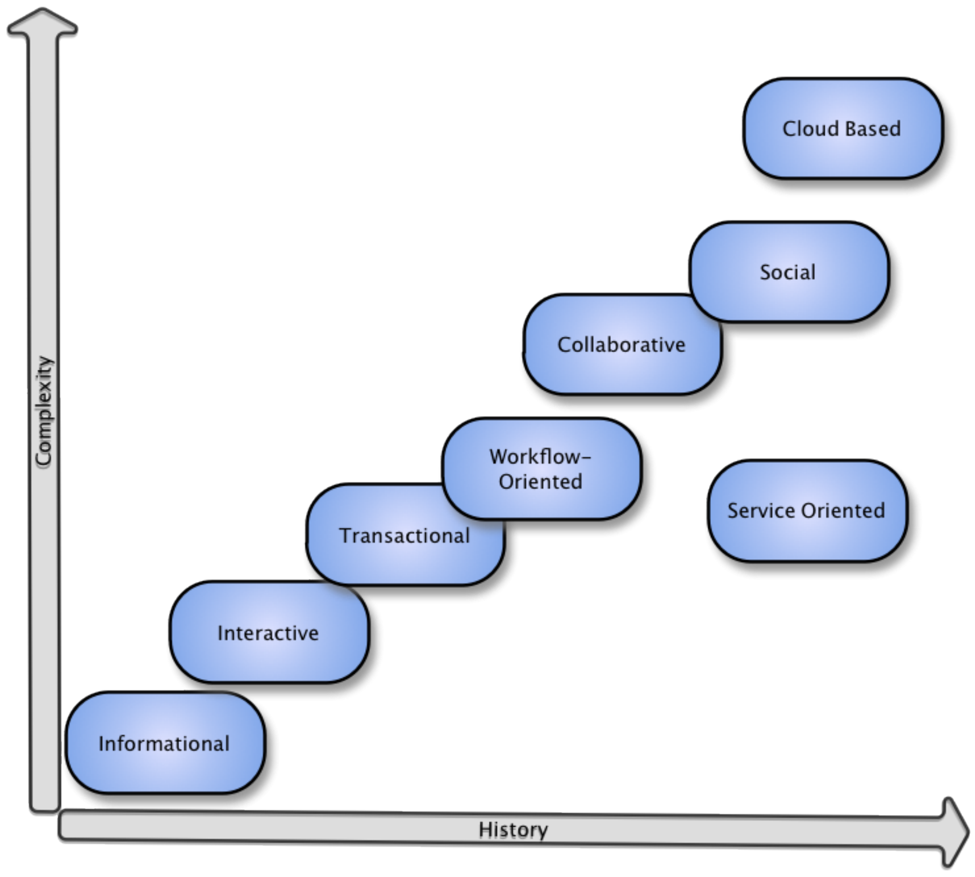


Figure 2.3: Web application categories [5, extended]

Informational. Informational Web sites are the simplest form of information sharing. They consist of text which can be combined with graphics. A Web server can host this kind of content as single, ready made (static) HTML documents. Typical examples are online newspapers, product catalogues, newsletters, manuals, reports, and online books.

Interactive. The invention of the Common Gateway Interface (CGI)[3] made it possible to create Web applications which could react based on user input. This can be used to provide more dynamic content, such as registration forms, customized information presentation, online games, etc.

[3]http://www.w3.org/CGI/

Transactional. Taking the possibilities of interactivity to the next level, transactional applications allow the user to actually change the available content by, for example, buying a certain good, booking a service, or by paying a bill. Doing this requires a way of storing and querying this content in a transactional manner. A common approach is to use databases to realize this functionality.

Workflow Oriented. A lot of activities rely on well defined workflows. This is especially true at Business-to-Business (B2B) and Business-to-Government (B2G) operations, which follow strict processes. Enabling the execution of such operations over networks, adds another layer of complexity to Web applications. Possible examples include online planning and scheduling, inventory management, status monitoring, and supply chain management.

Collaborative. Comprehensive communication and information sharing defines another category — Collaborative Web applications. The literature [3] names distributed authoring systems and collaborative design tools as examples.

Service Oriented. Service Oriented Architecture (SOA) is an architectural paradigm and discipline that enables connecting any sort of client with information providers via services across disparate domains of technology and ownership. Different kinds of architectures rely upon SOA as the enabling foundation, including the automation of Business Process Management (BPM), mashup applications (applications that aggregate multiple functionalities from different sources to create new services), and the multitude of new architecture and design patterns generally referred to as Web 2.0 [34].

Social. A lot of discussion around Web 2.0 concentrates on so-called "social networks" like Facebook[4], Myspace[5], LinkedIn[6], etc. These platforms allow people to socialize, share information, and express their opinion on almost any topic. They also provide interfaces allowing external programmers to integrate their applications to create a richer user experience.

[4]http://www.facebook.com
[5]http://www.myspace.com
[6]http://www.linkedin.com

Cloud Based. Cloud computing incorporates a number of young and emerging aspects of Web based application development. Tim O'Reilly defines three different categories in one of his blog entries[7] on O'Reilly Radar[8]:

1. **Utility computing.** Utility computing sis the provision of virtual machine instances, storage and computation at pay-as-you-go utility pricing and hosted off-premises on the Internet. This means that one can get an almost infinite amount of operating systems, storage space or computing power just by utilizing those service over the Web, while paying on a base of usage. Amazon.com[9] is a forerunner in this area providing all of the services described above.
2. **Platform-as-a-Service.** Though Amazon's offerings are seen as a Platform-as-a-Service (PaaS) solution [35], Google App Engine[10] is an even more integrated solution with its integrated Phyton and Java execution frameworks and a set of APIs[11]. These allow running Web applications without having to care about hosting.
3. **Cloud-based end-user applications.** This term describes applications which were traditionally available as desktop software and are now delivered through the Web. This includes applications like word processing, spreadsheets, photo editing or e-mail. The challenge here is to combine the user interface functionality of desktop software applications with the wide reach and low-cost deployment of Web applications. The desktop part in this includes providing an interactive user interface for tasks like validation and formatting, fast interface response times with no page refresh (and thus omitting the usual request-response data exchange flow typical for the Web), common user interface behaviors such as drag-and-drop and the ability to work online and offline. The Web component includes capabilities such as instant deployment, cross-platform availability, and leveraging widely adopted Internet standards [36]. With data being processed and hosted in the cloud in contrast to the user's local PC, a greater focus has to be put on security and reliability.

2.3 Characteristics

To understand what Web engineering is about, it is essential to get a basic perception of the technologies involved. This section therefore introduces the roots and

[7] `http://radar.oreilly.com/2008/10/web-20-and-cloud-computing.html`
[8] `http://radar.oreilly.com`
[9] Amazon Web Services: `http://aws.amazon.com/`
[10] Google App Engine: `http://appengine.google.com`
[11] such as the Google Maps API, Google Base API, and the Social Graph API

organizational fundamentals. It also further investigates whether Web engineering is just a new application sub-domain of software engineering, only “slightly adapting already existing approaches in terms of methodologies, principles, standards and best practice guides” [37, p. 2]; or if it is in fact a discipline on its own right, requiring new solutions. A prerequisite for answering this question is to define the characteristics of this discipline and to outline what separates it from other, related areas.

2.3.1 Involved Disciplines

One might argue that a Web application is nothing more than a client-server application with an emphasis on aesthetic presentation (e.g., layout, graphics, audio and video elements) and provisioning of information and that both Web applications and conventional client-server applications share the same attributes [14]. On the other hand, there are reasons that indicate that Web engineering consists of more parts than software engineering. This approach leads to seeing Web engineering as a multidisciplinary field encompassing diverse principles such as management, information systems, and computer science. Major contributing areas include systems analysis and design, software (and requirements) engineering, design and layout creation, hypermedia engineering, human-computer interaction, data mining, project management and artificial intelligence [2]. This explains how Web engineering combines a variety of other — not only technical — disciplines with the classical IT world. In total, four actively involved parties can be identified and are depicted in Figure 2.4. In this model, the customer is displayed as an indirect contributor as this role is usually not involved in designing the application from the ground up but plays an increasingly important role where customer created content adds real value.

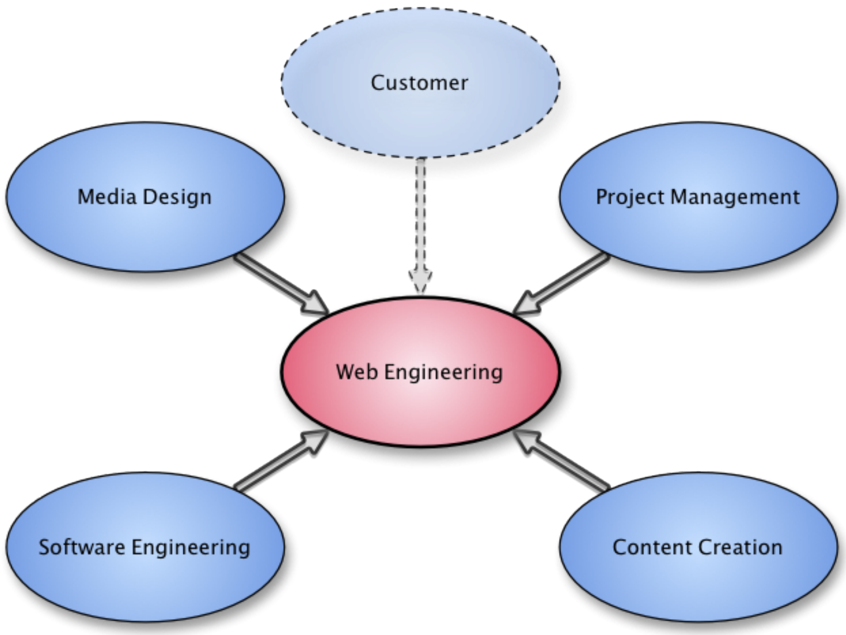


Figure 2.4: The roots of Web engineering

Project Management. The project manager's role is to coordinate efforts and timelines between the customer and the other three involved parties. As the leader of a project, a project manager is ultimately responsible for aligning everybody involved in respect of project progress and outcome.

Content Creation. Content is maybe the most sensitive part of a Web project as it delivers the core message. As with other media, such as newspapers, or magazines, the complexity is to keep content easy to understand whilst communicating the desired information. Spreading certain content might be legally restricted, therefore necessary approvals might significantly slow down the update rate of a Website.

Software Engineering. Depending on which category (cf. Section 2.2) a Web applications falls into, it more or less relies on software engineering to be realized. The more complex the application becomes, the more likely software engineering will play an important role in achieving the desired outcome. This includes creating server-side solutions as well as sophisticated client-side scripts.

Media Design. Together with the raw content of a site, how it looks is what users perceive. Media design covers a broad range of topics, from design oriented tasks such as sound, video, and picture editing to HTML and CSS coding and client-side scripting. As "anything that is a great print design is likely to be a lousy Web design" [38], these two disciplines are clearly separated. Also the wide scope of duties differentiates the Web designer from traditional print design.

Though theoretically one single person might be responsible for all these roles, often subject matter experts contribute with their expertise. Only by guaranteeing a steady flow of information and collaboration between all contributors, it is possible to successfully deliver Web projects.

For further investigation it is important to define the differences between software engineering and Web engineering. This provides the basis to further show the extent to which classical development approaches are suitable for developing Web applications. In order to deepen the understanding of what makes Web engineering a discipline in its own right, the following parts of this chapter give a detailed and categorized list of characteristics specific to the application domain discussed.

2.4 User Perspective

This section describes the specific characteristics of a Web application from a user's perspective. It concentrates on aspects related to product quality and usage. Following the ISO 9126 model for external and internal quality [39], software quality consists of six main quality characteristics, namely:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

As software engineering is an integral part of Web engineering, these factors apply to the latter in a similar manner as to the former. However, as Websites heavily focus on content and aesthetics these are key factors influencing the overall quality and usability of projects, targeted at the end-user. Hence, the following pages concentrate on these aspects.

2.4.1 Usability and Accessibility

“Bad usability equals no customers!” [40, p. 14]

According to [41], six usability factors can be named to describe “ease of use” or “user friendliness”:

- Fit for use: The system provides all functionality the user needs in a specific situation.
- Ease of learning: Defines how easy the system can be learned by different users.
- Task efficiency: Defines how efficiently regular users can accomplish tasks.
- Ease of remembering: Defines how much users have remembered from their last session(s).
- Subjective user satisfaction
- Understandability: Describes if users comprehend what the application actually does and why certain information (e.g. error messages) appears.

These factors apply whether the application is Web based or not. One of the fundamental requirements of any Web application is that it is possible to use it for its intended purpose. However, it must be designed to be easily accessed by possible visitors. Not following this basic requirement might quickly lead to a drop in usage. Accessibility, therefore, does not only cover the ease of use, but also describes whether an application can be used by physically handicapped people, especially by those with a visual impairment. “More specifically, Web accessibility means that people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web” [42]. There is a number of guidelines available on this topic, such as Authoring Tool Accessibility Guidelines (ATAG), Web Content Accessibility Guidelines (WCAG), User Agent Accessibility Guidelines (UAAG), and W3C technical specifications (HTML, XML, CSS, SVG, SMIL, etc.) [42, 43].

The level of usability of document-oriented (cf. Section 2.4.3) software very much depends on content aesthetics. Inappropriate colors, badly edited text or a confusing layout will negatively influence ease of use and hence the rate of users returning. Moreover, there might only be a quite vague concept of both content and design in early phases of a Web project, as both develop gradually over time. The basic usability concept must therefore be flexible enough to scale with changing requirements.

Desktop-oriented software typically follows recommended interface guidelines like the ones available for Windows[12], Mac OS X[13], or the GNOME project[14] and uses GUI libraries of the operating system they are written for. Hence, colors, styles, basic layout and the overall look-and-feel are very much a given. As they are basic requirements from the start of a project, this reduces the need to research these parameters and allows this step to be completed in an early phase of the development process. As stated in [44], considering “data as separate from the controls used to display is new, because applications on other platforms combine data and controls as a seamless whole”. Many users do not distinguish between their browser and *the Internet*, thus mixing up features of the actual Web-application and the displaying component.

2.4.2 Design

To a certain extent, usability and quality can be measured. Design, on the other hand, is very much subject to personal preference or taste. Web applications are competing with an overwhelming number of similar offerings distributed through a medium, leaving it to the choice of the user which application to use, or which Website to visit. Before knowing the actual content, the first thing attracting a user is the way it is presented visually. If the aesthetics discourage a potential customer from further exploration of the site, the best application may go virtually unnoticed by a broader audience.

Another factor accounting for the relatively high importance of visual design is the showcase role a Website plays not only for companies but all sorts of organizations, such as clubs, societies, governments, non governmental organizations, communities, etc. The Web representation might even be responsible for the main source of income for the organization behind it. Beside the actual content, visuals are a major way of communicating the desired statement through a Website. According to [40], one can distinguish between two basic approaches to design. These two extremes see the artist either as a person expressing personal views and emotions or as a service provider, fulfilling customer requirements. The design of a corporate Website frequently ends somewhere in between of these two concepts. As Websites often represent organizations, acting as the *face* to the public, they often have to follow corporate or brand guidelines — including specific colors, logos, or fonts — even if these standards may not be appropriate for Web usage. This can be a hindrance to creativity. Designers not only have to fulfill given prerequisites in

[12]http://msdn.microsoft.com/en-us/library/ms997506.aspx
[13]http://developer.apple.com/documentation/userexperience/index.html
[14]http://library.gnome.org/devel/hig-book/stable/

terms of design standards, but also considering usability and overall attractiveness of their product. The skills and early integration of designers, therefore both play a vital role in various kinds of Web projects.

2.4.3 Content

The major part of a Web offering is about providing content. Content can be presented in different forms — video, audio, text, pictures, etc. — thereby content maintenance and frequent updates are important factors in raising the profile of a site. In many cases this content is not created by the development team, but by external providers such as the marketing department, journalists, external agencies, etc. The developer in turn is responsible for providing a framework allowing the entry, styling and display ´of the content. Literature on this subject summarizes this characteristic as the "document-oriented" [4], or "document-centric" [5] nature of the Web.

2.4.4 Non-linear Navigation

The Web provides almost unlimited ways of navigating through information. In contrast to linear navigation systems — such as presentations, books or cinema — non-linear systems provide more options. Newspapers, for example, to a certain extent, allow the user to jump between content without losing information. For the strongly non-linear Web and other hypertextual media, three information structures (as depicted in Figures 2.5, 2.6, and 2.7) are described as being relevant [45].

The structure of the Web provides the user with a multitude of different ways of finding information, such as browsing (e.g., on a newspaper site), queries (e.g., searching on a shopping portal), and guided tours (e.g., e-learning or e-detailing applications) [5]. Even though this diversity suits human learning ability [5], failure to systematic design a Web application's navigation flow may result in what is referred to as the "lost in hyperspace syndrome" [46], confusing people to the extent that they are unable to find the information available.

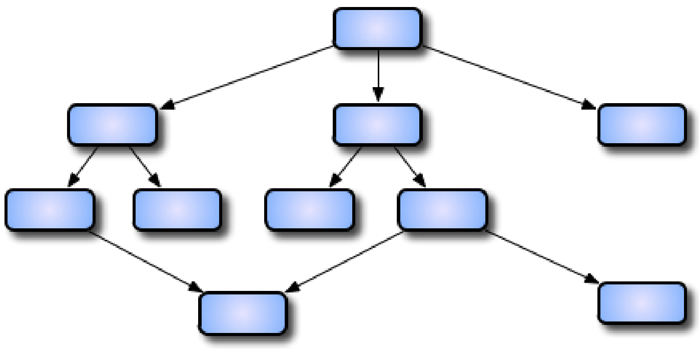


Figure 2.5: Navigation: Hierarchy

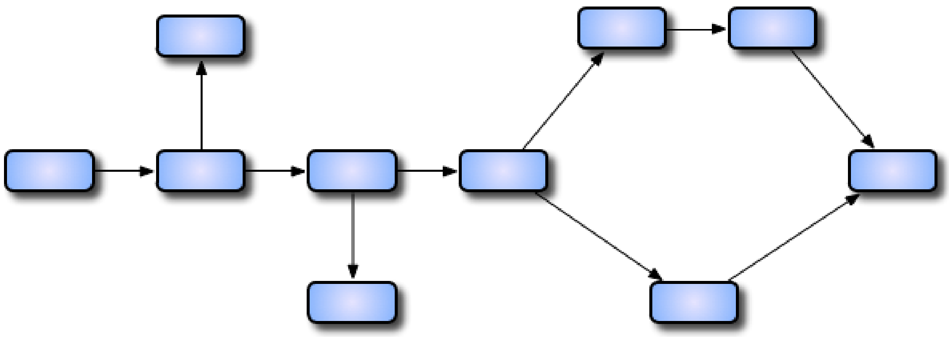


Figure 2.6: Navigation: Multipath

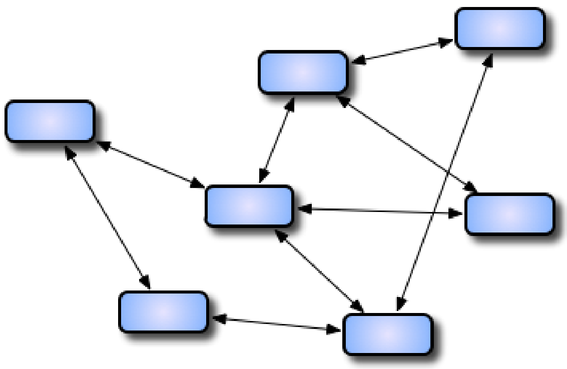


Figure 2.7: Navigation: Web-like

2.4.5 Persistence

Non-Web applications usually provide operating system specific mechanisms for closing programs. This can be realized via a *Close/Exit* button or menu entry, or simply via a specific symbol in the window chrome. In contrast, Web applications cannot clearly be *stopped*, as they are accessed from within another application — most commonly a Web browser. Leaving a website by following a link implicitly *closes* the previously opened site but leaves no information on whether the work in progress is saved. Reopening the browser or clicking the *Back* button might lead the user back to a previously opened application but usually the old state is lost, as no action has been performed to persist it [44].

2.5 Developer Perspective

The literature [47, pages 49, 50] identifies four issues, or areas of interest when it comes to Web development:

1. Data/Information: problems with data representation
2. Navigation: problems with navigation structure and behavior
3. Functionality: all problems not related to navigation and affecting application functionality
4. Presentation/Interface: issues related to interface design and (data) representation

The order of the list represents the historical importance of the individual items with data/information being central to the Web as a medium. This contrasts with a more software engineering-centered view, which emphases functionality since this aspect is taken into account by all traditional software design methods. Navigation has been mainly introduced by Web application engineering as it is used as an abstraction in the context of hypermedia and the Web. The next paragraphs explain technical attributes defining Web applications in contrast to their desktop-centric counterparts and thus influencing the way in which they are developed.

2.5.1 Hypertext and Hypermedia

Hypertext and hypermedia refer to Web pages and other kinds of on-screen content that employ hyperlinks [33]. Hypertext refers to the use of text and static graphics linked by so called *hyperlinks*. Hypermedia on the other hand describes the visualization of multimedia content, such as video, audio, or interactive media. It is common for people to use *hypertext* as a general term that includes hypermedia [33]. This kind of representation makes use of the concepts of *nodes* and *links* to connect content. A node typically refers to pieces of content (such as Websites) and links represent the route to navigate through this information. It is this linking capability which allows a non-linear organization of text and other media [48]. In contrast to linear media, such as books (e.g., novels), or movies, this concept gives the user the ability to freely choose which part of information to consume next. Links do not have to be statically predefined by, e.g., an editor, but can be generated dynamically according to the currently presented content. For example, one could imagine a Web shop for HiFi products presenting an aggregated list of shop items, filtered by the search-query *amplifier*. The generated list is presented across a number of sequential Websites to limit the number of displayed items to a certain amount. To navigate from one page to another, links are created dynamically by the underlying software (*computed link*).

When creating hypertext, authors have to take care not to confuse users by disorientating them or by presenting too much information and consequently overstraining their cognitive capabilities. Disorientation is the tendency to lose one's bearings in a non-linear document. Cognitive overload is caused by the additional concentration required to keep in mind several paths or tasks simultaneously [5]. Meaningful linking and link naming reduce cognitive overload [48]. The usage of breadcrumbs, sitemaps, or tag clouds can also help in keeping the user oriented and focused.

To achieve a unified representation for non-linear navigation across different environments, it is necessary to rely on a common specification for visualizing nodes and links. This is realized by using HTML (cf. Section 2.1.3).

2.5.2 Increasing Complexity

According to [49], "the complexity of tasks performed through Web applications is increasing". With Web offerings providing higher levels of interaction, including heavy client-side processing and extensive use of multimedia technologies, the demands on application developers are continuously increase. In another paper [50], the same team of authors reviews 15 traditional *Web methodologies* and finds that these, and the tools they bring with them "are incomplete or inadequate" in giving an answer to the demands this complexity entails. As applications increasingly provide the same feature-richness as desktop software, development processes and tools — as well as the software designers themselves — have to adapt in order to cope with this situation.

2.5.3 Device Heterogeneity

Web applications can be run on a multitude of systems and under different conditions. Screens vary in size and resolution and processing power is limited on mobile devices or old computers. Also network connection bandwidth limitations must be taken into account. But it is not only hardware that influences compatibility. Probably the most important factor is the variety of browsers available. Web applications often require a client-side application or environment in which to be executed or viewed. Usually this application is the preferred Web browser installed on the user's desktop or laptop. Relying on a secondary application means that the Web application's behavior is not only determined by itself, but is also influenced by external factors. Though institutions such as the "Web Standards Project"[15] strive to implement "standards that reduce the cost and complexity of development while increasing the accessibility and long-term viability of any site published on the Web" [51], not all browsers comply with these standards, such as the ACID3 [52]. This can potentially lead to Web pages not being rendered as expected in browsers that are not 100 % standards-compliant. Not only do the browsers vary in their compliance with web standards, the performance of their HTML and Javascript rendering engines differs too[16].

Developers therefore might want to concentrate on platforms with a large user base. To give an overview of which browsers are used to what extent, Table 2.1 below contains usage statistics from three different sources. These sources are W3Counter[17], MarketShare by Net Applications[18], and StatCounter GlobalStats[19].

The table summarizes the statistics for January 2011. Where no distinction could be made between browser versions, only aggregated numbers are displayed. Additionally, a calculated average is given. The table only contains browsers with more than 1 % of market share. Despite Microsoft Internet Explorer (IE) clearly leading the field, previous studies show it to be loosing ground to its competition. It must be added that all three IE generations use different versions of the underlying rendering engine *Trident*, all behaving differently when it comes to both

[15]The Web Standard Project: http://www.webstandards.com
[16]http://www.favbrowser.com/category/benchmarks/
[17]http://www.w3counter.com
[18]http://marketshare.hitslink.com
[19]http://gs.statcounter.com

standard compliance and performance. To give a clearer picture, the average usage percentages are presented in a separate Figure 2.8.

Browser	**W3Counter**	**MarketShare**	**StatCounter**	**Average**
IE	**41.27 %**	**54.84 %**	**46.86 %**	**47.66 %**
8.0	26.02 %	33.02 %	-	-
7.0	10.71 %	8.76 %	-	-
6.0	4.54 %	13.06 %	-	-
Firefox	**29.55 %**	**22.04 %**	**30.79 %**	**27.46 %**
3.6	25.36 %	18.50 %	-	-
3.5	2.76 %	2.22 %	-	-
3.0	1.43 %	1.32 %	-	-
Safari	**4.74%**	**3.24 %**	**4.81 %**	**4.26 %**
5.0	4.74 %	3.24 %	-	-
4.0	1.05 %	1.59 %	-	-
Chrome	**11.06 %**	**7.32 %**	**14.88 %**	**11.09 %**
8.0	11.06 %	7.32 %	-	-
7.0	1.45 %	1.62 %	-	-
Opera	**-**	**1.51 %**	**2.07 %**	**1.19 %**
10.x	-	1.51 %	-	-

Table 2.1: Browser Usage January 2011 by Source

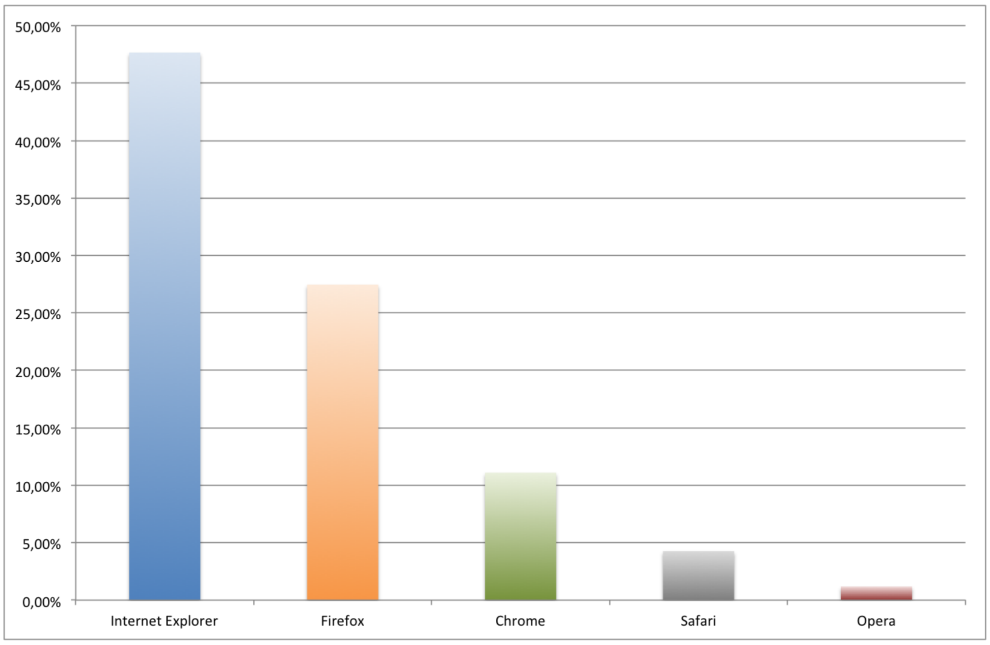


Figure 2.8: Averaged Browser Usage January 2011

2.5.4 Concurrency

Web applications may be accessed by a large number of users at the same time. The manner in which the site is used can vary greatly between these individuals. In some cases, the actions of one user (or one set of users) may have an impact on the actions of other users or the information presented to other users [14]. This is typically valid for collaboration and social platforms as well as transactional applications.

2.5.5 Unpredictable Load

The number of users of a Web service may vary significantly over time. This stresses the importance of scalability, availability, and performance for both software (e.g., the code base itself) as well as hardware (the hosting infrastructure). As a Web application is a collection of a number of (more or less) independent systems, performance bottlenecks can occur at numerous points:

- Hardware client- and server-side: server (RAM, storage, processing power, etc.), client PC, network adapters, routers, switches, network quality

- Software server-side: operating system, Web server, application code, database
- Software client-side: operating system, browser, other Web clients, application code (e.g., HTML, Javascript, Flash), network quality

Some of these possible error sources can be easily tested whilst for others it is difficult or impossible. For example, it is not feasible to replicate each and every combination of client PC, operating system, browser and network connection. Other factors, however, can be be secured against possible failure. Servers, application code and databases can be load tested and a standardized set of client infrastructure can be checked too.

2.6 Management Perspective

Using and building Web applications does not only require users (cf. Section 2.4) and developers (cf. Section 2.5), but also involves a management component, as this section shows.

2.6.1 Estimation Complexity

A successful project often means not only meeting timelines and requirements but also to stay within a certain budget. The Standish Group's CHAOS report for 2009 [53] gives a number of only 32 % of software projects "succeeding which are delivered on time, on budget, with required features and functions". Other sources [54] are less pessimistic, but still see a failure rate of 11.5 % to 15.5 %. One of the management objectives therefore is to realistically estimate costs, time, and effort. Since "80 % of software scrap and rework is caused by 20 % of the changes" [55], early estimates based on sound techniques and numbers allow managing resources effectively.

Cost estimation is not only crucial but also very challenging for a number of reasons. As the Web is a very fast moving medium, Web projects have short schedules with tight time-to-market timelines, usually from a few weeks up to six months [56]. The diversity of the technologies used increases the planning complexity as it requires specialized knowledge in predicting the efforts of all stakeholders. As a still quite *new* discipline, Web engineering attracts young people and teams tend to be rather inexperienced, which negatively influences the quality of effort and time estimations. Small teams and ad-hoc processes often prohibit using approaches established for software engineering.

2.6.2 Team Complexity

Web projects involve multidisciplinary teams made up of art designers, copywriters, frontend developers, backend developers, interface designers, testers, and project managers [6]. Sometimes marketing personnel can also be added to this list. This heterogeneity can make it difficult to set clear responsibilities, as project stakeholders with very diverse perspectives are concurrently working on one single product. Due to their individual and professional backgrounds, team members might not share a common understanding of what project success means. Here, project management can step in to provide a shared vision. Having numerous interfaces to other project participants creates a certain communication overhead that has to be accounted for.

Web engineering project teams are not only heterogenous but also very young by trend [57]. This lack of experience can potentially negatively affect quality and performance and increases the level of management necessary.

2.6.3 Heterogenous User Base

As Web applications and especially Websites are available to virtually anybody with an Internet connection, it can be difficult to determine their typical user base. This in turn complicates the design of software so that it covers the special needs of a certain audience. Varying levels of technical experience, taste, and ways of approaching a Web offering must be considered. Even in early project phases, management can define the target audience to limit complexity. Given that, formal experiments with a large number of test users (usually at least 20, often 50 or more) can, according to [58, p. 68] provide an:

- "objective measurement of performance of a single design, or"
- "objective comparison of the performance of two or more alternative designs".

These tests would ideally be performed in two different phases of the project [59]:

- Early mock-ups: Early in the design phase it is possible to present mock-ups of the final product to a selected user group to get feedback.
- Functional prototypes: After a "representative part of the site" is working, sample users can perform a set of predefined tasks to see if usage is straightforward.

2.6.4 Frequency of Updates and Rapid Deployment

The WorldWideWeb is a very fast moving medium. Outdated content, old fashioned design, or usability that is not on par with the most recent developments can quickly turn users away from a site. The resulting maintenance work thus creates significant effort even after a project had been successfully completed. Taking this into consideration before building the application can possibly reduce subsequent costs.

Chapter 3

Software Engineering Paradigms

"The only constant in life is change!"[1]

This chapter outlines existing, well-documented software engineering methodologies. It gives an overview on the history on software engineering and summarizes important historical approaches such as the *Waterfall* (cf. Section 3.1.2) and the *Spiral* (cf. Section 3.1.3) models. Subsequently, a general introduction into agile paradigms (cf. Section 3.1.4) is given, explaining their purpose and the features common to all of them.

The last three sections each describe one specific software development methodology, namely the *Rational Unified Process - RUP* (cf. Section 3.2), *Extreme Programming - XP* (cf. Section 3.3), and *Scrum* (cf. Section 3.4). Here, the RUP represents a more structured and process oriented solution, whilst the other two focus on agile principles.

3.1 The History of Software Engineering

One might question why *Software Engineering Methodologies* exist at all. Shouldn't it be quite straightforward to collect a few requirements and to translate these into working code? This thinking probably was the main reason behind what is known as the *Software Crisis*. This term is used to describe the poor state of software engineering at the time the first NATO Conference on Software Engineering took place in 1968 in Garmisch, Germany [8]. Then, IBM had developed OS/360 and TSS/360, two operating systems. The creation of the former of the two cost IBM over $ 50 million a year during its preparation, and more than 6000 man-years had to be invested during implementation. The TSS/360 project even had to be stopped at all. The huge investment in OS/360 and the failure of TSS/360 led to a common belief that it was necessary to enforce the "establishment and use of sound

[1]Heraclitus: http://stanford.library.usyd.edu.au/entries/heraclitus/#0th

engineering principles to obtain economically software that is reliable and works on real machines efficiently" as F.L. Bauer put it at the NATO conference [8]. This was the first time a larger audience had discussed approaches to professionalize the development of software, and it is often thought to be the birth of modern software development. The following paragraphs illustrate how this discipline evolved through the years.

3.1.1 1956, 1961: The Beginnings

The first explicit model of software engineering was outlined in H.D. Benington's "Production of Large Computer Programs", published in 1956 and reprinted in 1987 [9]. He describes a "Phase Model", dividing the development process into sub-sections with dedicated purposes. W.A. Hosier followed the same approach and, amongst other things, demanded testable requirements, precise interface specifications, and lean staffing in early phases [10]. The main purpose of these early methodologies was "to provide a conceptual scheme for rationally managing the development of software systems" to serve as a basis for "planning, organizing, staffing, coordinating, budgeting, and directing software development activities" [60].

3.1.2 1970: The Waterfall Model

Without explicitly mentioning the word *Waterfall*, Winston Royce planted the seeds of the *Waterfall Model* in his paper "Managing the Development of Large Software Systems: Concepts and Techniques" [11]. The actual term was defined by Barry Boehm in order to differentiate it with his own invention, the "Spiral Model" [12]. As Illustrated in Figure 3.1, this traditional model of software development sequentially lists the basic phases of the project lifecycle. Each of the phases ends with a formal assignment by involved stakeholders.

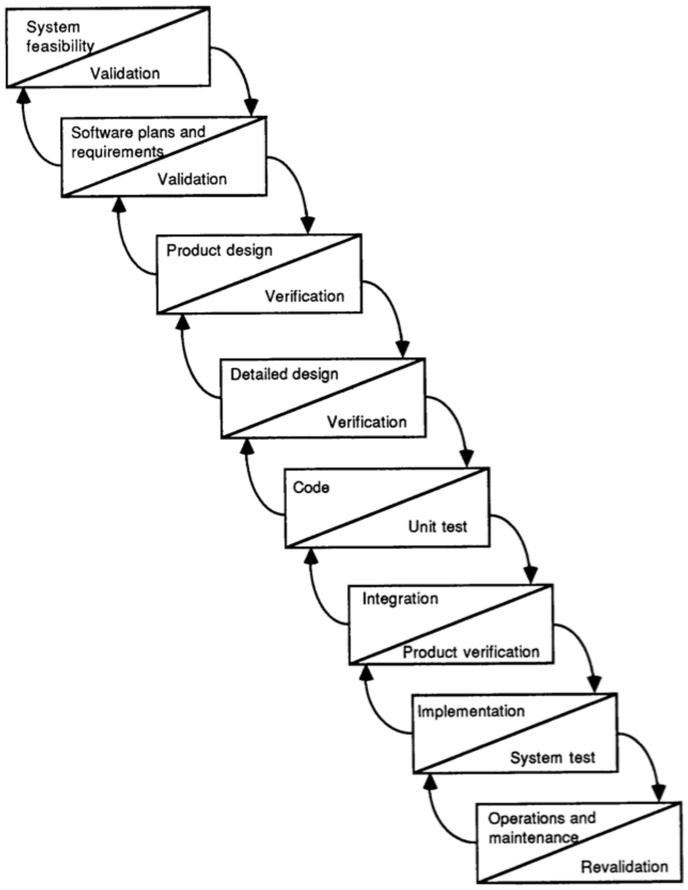


Figure 3.1: The Waterfall Model [12]

The following phases are part of the model:

Feasibility and Requirements. In this phase customer requirements are collected and documented. Starting from a high abstraction level, they are further refined to a degree where they can act as a basis for the subsequent phases. The requirements are stored in a repository and are worked on as resources become available. The phase ends when all requirements are signed off by all involved stakeholders, such as customers, managers, and development leads.

Design. During the design phase, the development team builds the system architecture. The finished design documentation is the foundation for further development.

Code and Test. Following the design document of the previous phase, the software is built and tested. The product of this phase is reviewed according to the quality gate checklist to see whether there are deviations from previous quality gate decisions [61].

Integration and Implementation. This is the release phase of the project. The individual software components are integrated into a releasable product. This is rolled out to the customer and user training takes place if required. The user checks whether the system meets the agreed requirements and has the opportunity to reject it if not.

Operations and maintenance. After the product has been released, it enters a stage of operation and maintenance. Bugs and problems are remediated and patches are applied.

Shortcomings. The Waterfall model suffers from a number of shortcomings, described throughout the literature (e.g. [12, p. 63]). A lot of problems are related to the great impact the requirements phase has on the project. A thorough collection and description of requirements must be performed at this very early phase as the project can only be continued when this phase is successfully completed. However, this phase is widely seen as the most complex part of building software [62] and making mistakes is very likely.

Though Royce already incorporated back-loops between two sequential phases and explicitly refers to prototyping as good practice, this is often overseen. Consequently, problems unnoticed in earlier phases must be resolved in a later phase, resulting in huge unforeseen costs and efforts. As there are no opportunities for customers to provide feedback on the system during the project, clients are only able to discover problems during system delivery and integration. This happens at a stage where huge efforts have already been put into development and it makes it very hard to respond to changes when the project is underway. This sort of big-bang integration at the end of the project can lead to unexpected quality problems, high costs, and schedule overrun. As each phase creates major artifacts, approving and correcting them can substantially delay further progress.

3.1.3 1988: The Spiral Model

The Spiral Model was first mentioned in 1988 by Barry W. Boehm in his article "A Spiral Model of Software Development and Enhancement" [12]. Instead of taking a *document-driven* or *code-driven* approach to the software process, it follows a *risk-driven* one. It is an incremental model and leads software development through an unspecified number of cycles or loops as demonstrated in Figure 3.2. Within each loop, a sequential development approach is taken, so that it "couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model" [63, p. 15]. The inner cycles of the spiral represent early analysis and prototyping, whilst the outer ones denote the classical software life cycle [60]. The radial dimension shows the cumulative costs until a certain time during the lifecycle [12].

The Spiral Model is divided into between three and six framework activities, also called *Task Regions*. These cut the spiral into slices. Figure 3.2 depicts a Spiral Model that contains four main Task Regions:

Determine objectives, alternatives, constraints. This covers customer communication, checking constraints and costs, and creating basic schedules.

Evaluate alternatives, identify, resolve risks. This includes activities for finding alternatives relative to the objectives, and addressing sources of project risk. Also prototyping and modeling are done at this stage.

Develop, verify next-level product. This is the main development step. The next prototypes are planned and requirements are refined to address remaining risks.

Plan next phases. This captures customer evaluation and feedback. Next iterations are planned.

Risk analysis is a key activity and occurs during each spiral cycle. It tries to determine all influences that might potentially cause the project to fail or to go over budget [60]. Prototyping is utilized as a manner of reducing risksto a point that a sequential software development process can be followed.

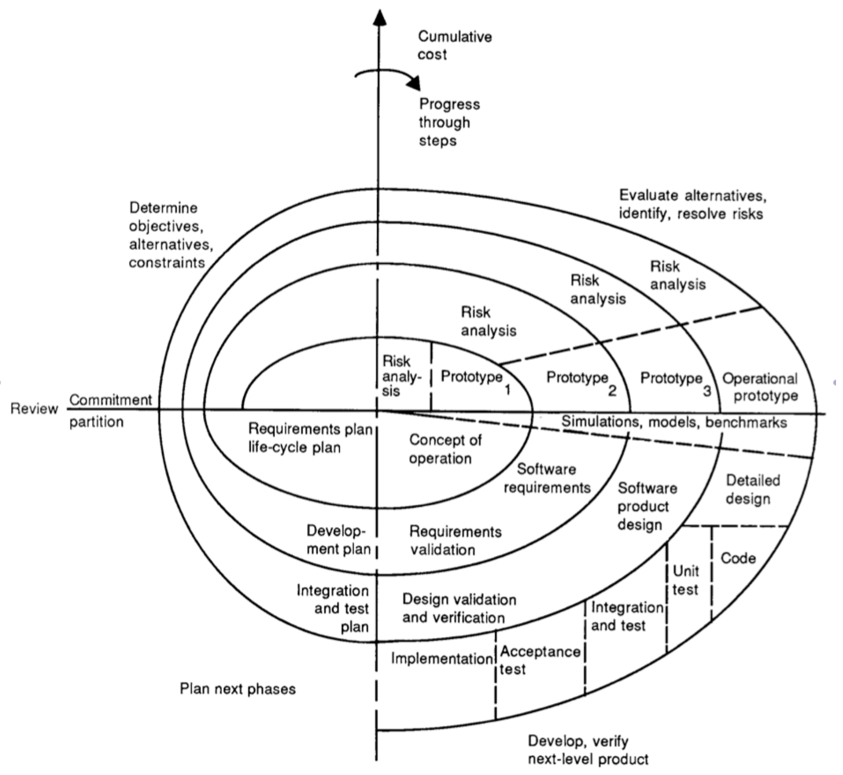


Figure 3.2: The Spiral Model [12]

The Spiral Model tries to solve the problems of the Waterfall model, which are caused by the following incorrect assumptions of the software development process [64]:

- There exists a reasonable well-defined set of requirements and it takes a defined amount of time to understand them.
- Changes to requirements during the development process are minor enough to be handled without substantially rethinking or revising the initial plans.
- System integration is an appropriate and necessary process. Based on sound architecture and planning its behavior can be predicted.
- It is possible and practical to schedule and predict the efforts of creating a significant new software application.

It may therefore be referred to as a theoretical foundation for the younger agile methodologies.

3.1.4 Today: Agile Paradigms

It is the role of a software builder to iteratively extract and refine requirements [62]. Unfortunately, it is often impossible to define all specification upfront. Also requirements change as software systems and their level of detail grow. Like the Spiral Model, agile methods accept this fact and apply extensive iteration to the communication flow between designer and customer throughout the whole process. In contrast to the Spiral Model, agile approaches clearly define the duration of each cycle with each iteration producing a working product. In contrast to the more abstract Spiral Model and though the name might not indicate, agile methods apply strict restrictions and often give detailed practical guidance for various aspects of software development. A common principle is to only produce output that is of value to the customer. Agile ecosystems all describe the usage of "light-but-sufficient" [65, p. 8] rules for software development and project management. They follow iterative approaches and embrace change in order to respond to rapidly changing environments. Agile development paradigms do not disregard processes and documentation, but try to keep both to a necessary minimum. One could argue that agile is Spiral but that Spiral is not necessarily agile as only the rules that come with these processes enforce their lightness.

There is a multitude of methods available, each of them focusing on different aspects of software development. Well documented examples of agile paradigms are:

- Scrum (cf. Section 3.4)
- Dynamic Systems Development Method (DSDM) [66]
- Crystal Methods [67]
- Feature-Driven Development (FDD) [68]
- Lean Development [69]
- Extreme Programming (XP) (cf. Section 3.3)
- Adaptive Software Development [70]
- Agile Unified Process (AUP) [71]

The Agile Manifesto. In 2001, 17 leading proponents of lightweight methods gathered in Snowbird/Utah to discuss what agility means to them. The resulting *Manifesto for Agile Software Development* [72] states:

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:"

- “Individuals and interactions over processes and tools”
- “Working software over comprehensive documentation”
- “Customer collaboration over contract negotiation”
- “Responding to change over following a plan”

“That is, while there is value in the items on the right, we value the items on the left more.”

In addition, 12 principles form the basis for agile development [73]. These emphasize the importance of delivering valuable software while embracing change in every phase of the project lifecycle. This is supported by short development lifecycles (weeks, not months) and close collaboration between business and development through face-to-face communication. The principles further stress the importance of keeping the participating individuals motivated by providing them with a supportive environment and trusting them to get the job done in self-organized teams. Developers themselves should strive for technical excellence, simplicity, good design, and self-reflection.

Representing these values, the “Agile Alliance” supports organizations and individuals who spread and apply agile principles to make the software industry more “productive, humane, and sustainable”[2]. Currently it has more than 4000 members, both individuals and corporate/institutional organizations.

3.2 Rational Unified Process - RUP

The Rational Unified Process (RUP) was invented by the Rational Software Corporation in 1998 and — after a take-over — has been the intellectual property of IBM since 2003[3]. The RUP is based on the “Unified Software Development Process” [18] and is probably its best known representation. Being a commercial product, IBM provides a rich set of tools supporting the process as well as numerous best practice guides to encourage commercial adoption. A center piece of the process is the extensive use of UML (“Unified Modeling Language”[4]) throughout all project phases.

Similar to the Spiral Model, the RUP emphasizes iteration to overcome problems of more static approaches. In addition, it contains extensive guidelines and a com-

[2]Agile Alliance Web site: `http://www.agilealliance.org/`
[3]IBM Rationale Web site: `http://www-01.ibm.com/software/awdtools/rup/`
[4]Object Management Group UML Web site: `http://www.uml.org/`

prehensive number of artifacts for each of the process phases [19]. This holistic approach is what sets the RUP apart, making it a very well defined framework for software engineering [74]. Commonly, the methodology is not judged as being particularly agile and neither is IBM member of the the Agile Alliance (cf. Section 3.1.4), nor is the process itself connected to it. Literature has, however, examined the agility potential of RUP [75] and there are efforts[5] underway to push the process into this direction [76].

The RUP incorporates four different aspects of software engineering and therefore is [18]:

- a software engineering process: The RUP describes the software development process in great detail. This includes the distribution of tasks and responsibilities, as well as workflow descriptions.
- a process product: IBM has developed a set of tools around this process.
- a process framework: Composed of *process models*, the RUP describes *who* is doing *what*, *how* and *when*.
- a collection of best practices: Six practices are covered by the RUP:
  - Developing software iteratively
  - Managing of requirements
  - Encouraging component-based architectures
  - Modeling software visually
  - Verifying software quality continuously
  - Controlling change management

The RUP divides the software development cycle into four consecutive *Phases* and *Iterations* (cf. Figure 3.3). Each of the phases is completed by a well-defined *Milestone* — a point in time at which certain key goals must have been achieved and where critical decisions must be made. These milestones provide the stakeholders the option to assess the project, in order to review progress and to define the plans for future work [77]. The phases are:

- Inception phase (I)
- Elaboration phase (E)
- Construction phase (C)

[5]Scrum and RUP: `http://www.ibm.com/developerworks/rational/library/feb05/krebs/`

- Transition phase (T)

As depicted in Figure 3.3, nine *Disciplines* are carried out throughout a project cycle. These, also named *Worflows* [18] can further be divided into six *Core Disciplines* (Business Modeling, Requirements, Analysis & Design, Implementation, Test, Deployment) and three *Supporting Disciplines* (Project Management, Configuration and Change Management, Environment).

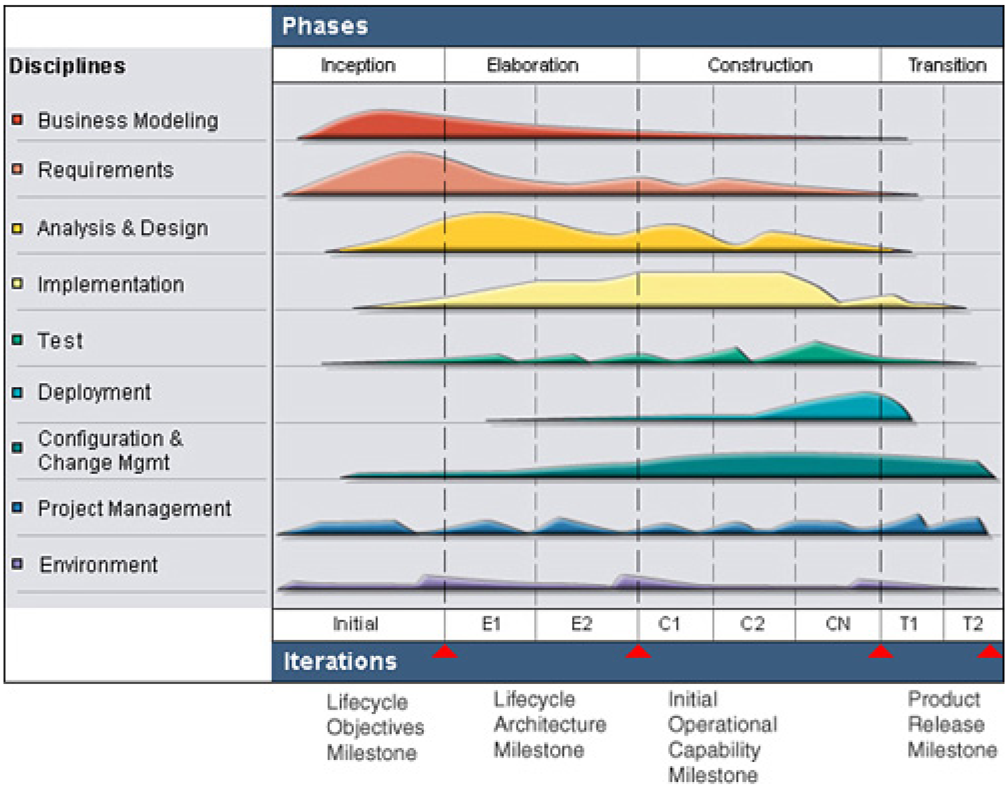


Figure 3.3: Nine RUP process disciplines [78]

A detailed view on all components as in Figure 3.3 is given in the following sections.

3.2.1 Underlying Principles

Being based on best practices [18] the RUP is founded on the base of eight principles [74]. These principles are:

- Attack major risks early and continuously, or they will attack you.
- Ensure that you deliver value to your customer.

- Stay focused on executable software.
- Accommodate change early in the project.
- Baseline an executable architecture early on.
- Build your system with components.
- Work together as one team.
- Make quality a way of life, not an afterthought.

3.2.2 Process Structure - The Structural Dimension

The Rational Unified Process is represented using four primary modeling elements: Roles — the *who*, Activities — the *how*, Artifacts — the *what*, and Disciplines — the *when*. Whereas the first three items are described in brief in this section, the RUP disciplines are listed in Section 3.2.3.

Roles. A role defines the part played by an individual or group when working in a team. One individual may have more than one role assigned and this assignment may change during the project lifecycle. Each role is associated with a defined set of activities.

Activities. As mentioned above, roles perform activities. Activities translate into work units. These have a clear purpose, usually expressed in terms of creating or updating such artifacts as models, classes, or plans [18, 77]. The duration of an activity usually ranges from a few hours to a few days. A comprehensive list of activities can be found in [18].

Artifacts. “An artifact is a piece of information that is produced, modified, or used by a process” [79]. When performing activities, roles both use artifacts as inputs and create them as outputs. Artifacts are tangible, as they constitute the actual productive output of a project during its lifecycle. The literature ([18, 77, 79]) categorizes artifacts into categories. Before actual development starts, documents such as the *Business Case* or the *Software Architecture* Overview” are written. *Models*, such as the Use-Case, or the Design models are created early in each development cycle to describe the system to be produced. Further breaking down models generates another artifact — the *Elements* within a model, such as a class, a use case or a subsystem. Later in the phase, actual value is produced in the form of *Source Code* or *Executables*.

3.2.3 Process Disciplines

A discipline is a sequence of activities that produces a result of observable value. Disciplines are grouped into nine categories as depicted in Figure 3.3. Each of the disciplines knows a defined set of roles, artifacts, and activities.

Business Modeling. Business modeling strives to create a common understanding of the target organization, usually adopted to the scope of the part of the business relevant to the software being developed. The goal is to improve mutual understanding in order to build a bridge between software engineering and business engineering by using a common language. As this might not apply to all organizations, “many projects may choose not to do business modeling” [79]. Working closely with project stakeholders, according to [77] it must give a complete overview on the status of the organization. This not only includes describing the *status quo* related to all “processes, roles, and responsibilities”, but also an outlook on strategies to adapt these. Consequently, a “domain model” is created to outline the affected parts of the organization.

Requirements. A requirement is a condition or capability to which a system must conform [18]. Consisting of functional and non-functional requirements, they basically describe what a software system should do. To achieve a common understanding between customers and developers, it is necessary to agree on the objects for development. This has to be captured in a universally understandable way. The RUP suggests Use Cases[6] as a human readable solution. They are created to represent the behavior of the system and consist of diagrams, together with comprehensive descriptions.

Analysis and Design. During the analysis part of this discipline, the goal is to translate requirements into a form of software architect *speech* — classes and subsystems. Design receives this information and further refines it. The “design model” [79] created predefines how the source code will be structured and written. Again, a number of UML diagrams support this process.

[6]For more on Use Cases and UML, e.g. read [80] or [81]

Implementation. The main purpose of this element is to perform the actual coding. This means organizing the code by implementing subsystems arranged in layers, and building classes and objects in terms of components. For each build (which represents an operational version of a subset of the end product), testing is performed in the form of unit tests. In bigger teams, setting up a build also includes the integration of the individual parts into an executable system. The RUP describes how to reuse existing components, and how to implement new components with well defined responsibilities.

Test. Testing is about assessing product quality. This includes verifying that the product is free of defects, that the system works as designed, and that it meets the given requirements. Following the iterative nature of the RUP, testing does not only apply to the final product, but is a task that is carried through throughout the project and includes artifacts like the software architecture [18]. It supports the team in finding defects as early as possible in order to reduce the cost of fixing them. Tests are carried out to ensure three aspects of quality — reliability, functionality, and performance. The RUP names a number of test stages, types, and models to provide sufficient coverage.

Deployment. The purpose of the deployment discipline is to deliver the software system to the customer. This not only includes packaging, distributing, and installing it, but also beta testing and performing end-user training to increase software acceptance. Also, existing data from legacy systems is migrated in this phase. In order to avoid problems related to the deployment activities at the end of the construction phase, these are not only part of the transition phase, but also have to be considered in earlier phases [79].

Configuration and Change Management. This discipline's purpose is to control the numerous artifacts that are produced during the project. Abbreviated *CCM* three interdependent functions can be identified:

- Configuration management (CM): Related to the product structure, it deals with artifact identification, versioning, and dependency control.
- Change request management (CRM): Related to the process structure, it manages and captures change requests, generated both by internal and external stakeholders.
- Status management and measurement: Related to the project control struc-

ture, it supports project management by providing meaningful figures about the project.

Project Management. In [79, p. 13] it is stated that "Software Project Management is the art of balancing competing objectives, managing risk, and overcoming constraints to deliver, successfully, a product which meets the needs of both customers (the payers of bills) and the users". The RUP tries to support project management by focusing on the iterative development process. This support is provided as guidance for planning the whole lifecycle of an iteratively conducted project and the iterations themselves, for accompanying risk management, and for progress monitoring using metrics and statistics. The RUP explicitly does not cover other aspects of project management, including managing people, managing budgets, and managing contracts.

Environment. As a support function, this discipline provides and maintains the software development organization's environment, both from a process and tool perspective. It continuously configures and improves processes and produces guidelines, supporting other team members in following them. By selecting, acquiring, and setting up the right tools, it helps other staff focusing on their work area. Also technical (IT) support falls into this category.

3.2.4 Process Phases — The Time Dimension

The RUP consists of four phases, covering the whole software development lifecycle and thus representing the dynamic organization of the process over time [79]. Each phase must meet a defined set of goals. To meet these milestones, the related tasks are addressed while iterating through the phase. "In many ways phases provide the glue which holds the RUP together, but the disciplines of the RUP capture its heart" [77, p. 2].

Inception Phase. Milestone: Lifecycle Objective (LCO)
The inception phase's primary goal is to establish a business case and to specify the project scope [79, 82]. This transforms the initial product vision into a real project. To successfully end this phase, it is necessary to reach a common understanding and vision among all stakeholders [18] and thus reach a decision on whether or not to continue the project. Depending on the complexity of the project, this phase may vary in length. To close the phase, the following artifacts should be created:

- The vision of the project, containing requirements, key features, and main constraints
- A business case, business context/scope, success criteria, and financial forecast
- The initial risk assessment
- The initial project glossary
- A project plan, containing all phases and iterations
- To be included optionally, there are architecture candidates, functional and non-functional prototypes, a business model, and an initial use-case model, also listing all actors

Elaboration Phase. Milestone: Lifecycle Architecture (LCA)
The purpose of the elaboration phase is to analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project [79, 18]. This requires having a very profound knowledge of the system, both in detail and at a high level. All functional and non-functional requirements must therefore be well known. The products of this phase are:

- A baseline product vision and software architecture description
- A working prototype
- A use-case model with at least 80 % coverage, containing all actors and most descriptions
- A revised risk list and business case, including staff, management, and test plans
- A development plan, covering the overall project, showing iterations and related evaluation criteria for each iteration

Depending on the complexity of the project, the work during this phase may be performed in one or more iterations (E1, E2 in Figure 3.3).

The lifecycle architecture milestone is the point at which stakeholders have to decide whether the architecture and vision can be seen as stable and it is sensible to continue.

Construction Phase. Milestone: Initial Operational Capability (IOC)
The main purpose of the construction phase is to build the software product. All features and components are implemented and tested to be ready for distribution.

More than the two preceding phases, this means hands-on work, so the emphasis is transferred from producing intellectual property to the development of a deployable product [18]. Again, this process is carried out iteratively.

At the end of this phase the IOC review is conducted where stakeholders assess the project's health and ensure that the software product is integrated on the agreed platforms, all user manuals are complete, and that the release is sufficiently documented [77].

Transition Phase. Milestone: Product Release (PR)
During the transition phase the product is rolled out to the user community. This includes testing by "both system testers and end-users, and corresponding re-working and fine tuning" [77, p. 4]. To fix final bugs, and to individually adapt the system, beta testing and user training is done together with the customers/users. This may be performed by iteratively releasing new versions until the desired level of quality is achieved. The primary objectives of the transition phase include [79, 18]:

- Achieving user self-supportability
- Achieving stakeholder approval that deployment baselines are complete and consistent with the evaluation criteria of the vision
- Achieving a final product baseline as rapidly and cost effectively as practical

At the product release milestone, the stakeholders decide if all objectives were met or if it is necessary to start another development cycle. This very much depends on user satisfaction.

Iterations. Iterations bring a number of advantages over sequential development. Managing risk as a part of each cycle makes mitigation easier. The strict top-down approach of the Waterfall — where only preceding phases may be refined — makes it very hard to integrate change in later phases. The more flexible nature of iterations makes change more manageable. According to [77], iteration increases the level of possible reuse as well as the ability of the project team to learn along the way. All these factors ultimately lead to better overall quality.

All RUP phases are divided into one or more iterations. Each of them only covers a certain portion of the entire system being developed. This sets the RUP apart from traditional approaches that complete one phase in one step [77]. For each iteration, there is a fine-grained plan and specific goals. Every iteration builds upon the results of its predecessors. These are depicted along the bottom of Figure 3.3, separating each phase into iterations. Every iteration results in an internal or external release of an executable product.

> "Traditionalists will often mistakenly think that the Inception phase corresponds to the traditional requirements phase, that Elaboration corresponds to design, that Construction corresponds to coding, and that Transition corresponds to testing. Nothing could be further from the truth." [77]

As Figure 3.3 depicts, nine different disciplines are performed throughout each phase and iteration.

The RUP utilizes iteration to cope with the following "wrong assumptions" [18] of sequential development:

- Requirements are frozen: Users, problems, technology, and the market change over time. This prevents successful capturing of all requirements at one single phase at the start of the project.
- The design can be *right on paper* before proceeding: Completing a design according to all obvious quality factors would require a formal definition of the problem. Commonly, this is not realistic due to lack of resources and methods.

3.3 Extreme Programming - XP

"The goal of Extreme Programming (XP) is outstanding software development" [15].

XP was originally developed by Kent Beck[7], Ward Cunningham[8], and Ron Jeffries[9] whilst working on a project for the American car maker Chrysler[10]. XP is a member of the Agile Alliance (cf. Section 3.1.4) and therefore shares its vision of how software development should be organized. XP is comprised of a number of different aspects [15]. XP is a philosophy of software development, endorsing the values of communication, feedback, simplicity, courage, and respect. It takes practices, proven useful in real world software development and provides a set of complementary principles, supportive of translating XP values into practice. These values and practices are shared by a dedicated community.

[7]Kent Beck: http://c2.com/ppr/about/author/kent.html
[8]Ward Cunningham: http://c2.com/~ward/
[9]Ron Jeffries: http://xprogramming.com/xpmag/bio
[10]Chrysler: http://www.chrysler.com

XP can be clearly characterized as lightweight or agile [83] and adheres to a set of paradigms [15]. Explicitly defining change as something inevitable, it endorses coping with rapidly changing requirements as a key claim. Another central principle is that developers only do what adds value to the customer and therefore reduce other, more process-related work. As XP was designed by developers for developers, it mainly addresses issues directly related to software development. It does not provide extensive coverage of topics related to project management, financial decision-making, operations, marketing, or sales.

The XP project flow is outlined in Figure 3.4. As plans are only valid for as long as the environment is stable, changing plans before they expire is fundamental [84]. *Prototypes* or *Architectural Spikes* help to create the *System Metaphor*. *User Stories* translate requirements into a common understanding. Constant refactoring and test driven development are used to harden the system and to uncover bugs. Frequently releasing software to the customer and passing the *Acceptance Tests* drives *Iterations* and helps to produce the feedback required to alter the software early in order to regularly produce *Small Releases*. Acceptance inherits the attitude that developers are responsible for proving to their customers that the code works correctly, not customers proving the code is broken [84]. Daily *Stand Up Meetings* steadily keep the management and development teams informed about the ongoing progress.

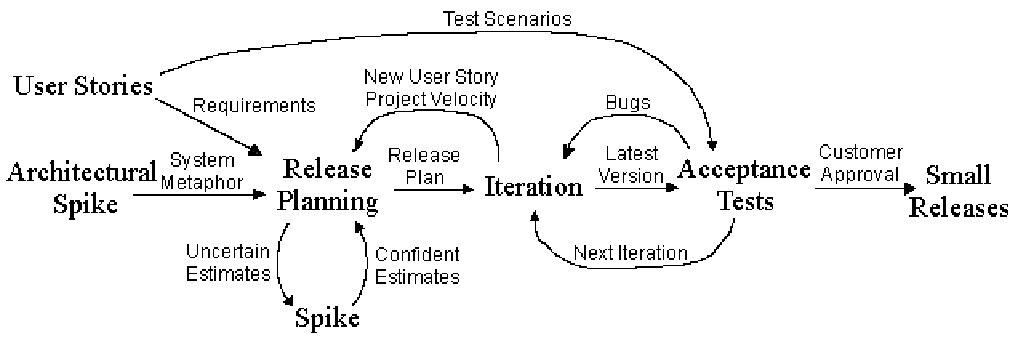


Figure 3.4: XP project flow [84]

XP is a developer centric methodology and tries to focus on productivity and customer satisfaction. Its success is documented by real-world studies. A JP Morgan Chase study [85] reported that — in comparison to using other software methodologies — implementing XP reduced the total amount of defects by 63 % on average. Even the amount of critical defects decreased by 79 % on average. For the given sample, development time and effort could be reduced by 44 and 47 % respectively. For the number of features not working, the study shows a 38 % advantage for the teams working with XP. At the same time, job satisfaction increased by 77 % for technology staff and 81 % for business stakeholders.

3.3.1 Values & Principles

XP is based on the values of communication, simplicity, feedback, courage, and respect [15]. In order to translate these values into something usable in a practical setting, it features a set of principles that every software project should adhere to. These are introduced in the following paragraphs.

Humanity. Software is developed by people and people have certain needs. XP values these and endeavors to fulfill them. The following needs are seen as particularly important:

- Basic safety: e.g., no fear of job loss
- Accomplishment: the opportunity to contribute to society/to the team
- Belonging: identification with the group
- Growth: personal development
- Intimacy: the ability to understand and to be understood by others

Economics. Software development has to create business value. The sooner money is earned, the better. Pay-per use (cf. Paragraph 3.3.3) is a way of realizing revenue soon after a feature is completed.

Mutual Benefit. This is the most important XP principle [15]. Every activity should benefit all persons involved. For example, creating test cases now not only benefits the current developer, but also all succeeding ones.

Self-Similarity. Solutions that work for one situation are likely to work in similar ones, even on a different scale.

Improvement. In XP, there is no *perfect* solution. XP strives to create excellence through continuous improvement and reflection.

Diversity. Diversity is an expression for a team where members are not alike but add different skills to the group.

Reflection. Reflection is about constantly thinking about *how* the team is working and *why* things are, or are not working.

Flow. Instead of cutting the software development process into distinct cycles, XP favors a steady flow of valuable output created by all activities.

Opportunity. Problems are seen as opportunities for change. In order to achieve excellence, it is not sufficient to just *survive* but to take problems as starting points for evolution.

Redundancy. Redundancy should be avoided unless it serves a valid purpose.

Failure. Failure is accepted as a consequence to constant improvement. If something fails, the least value that is created is knowledge.

Quality. In XP, quality is not a control variable. The trade-off between time and quality is questioned and the latter should never be sacrificed.

Baby Steps. Instead of going for the big change or big step, XP favors taking small iterations.

Accepted Responsibility. Responsibility can only be accepted, not assigned.

3.3.2 Mandatory Practices

Originally consisting of 12 practices [86], XP's current version consists of 13 *Primary Practices* and 11 *Corollary Practices* (both depicted in Figure 3.5). They represent things XP teams perform, or experience on a daily/regular basis. The practices are most effective when all are used in parallel. The primary practices are independent from other activities, and are designed to give immediate improvement. The corollary practices are somewhat more complex and it is recommended to start them after gaining experience with the primary ones [15]. This section describes all practices in brief. Information on them is taken from [15] and [87], if not referenced otherwise.

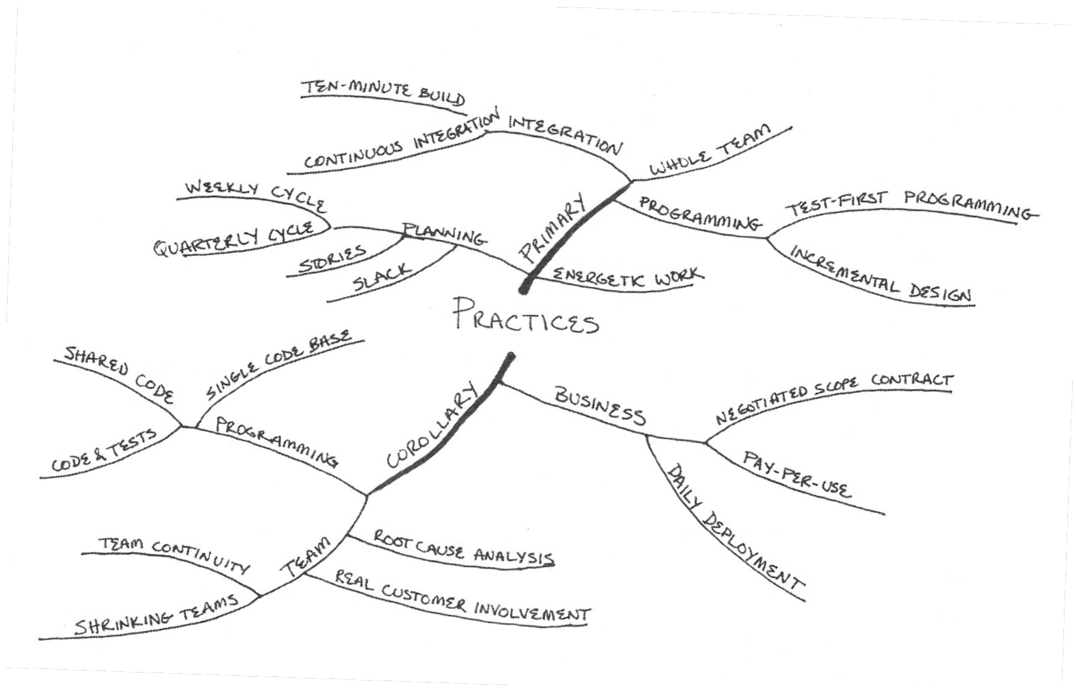


Figure 3.5: XP practices [15]

Sit Together. Sitting together means having the whole team in one space big enough for all team members. The idea behind this is to stimulate information sharing and cooperation. If this is not possible (e.g., working with multisite teams), XP encourages having as much face time as possible.

Whole Team. The whole team concept advises including enough persons in the team to cover the skills and experiences necessary for completing the project. This includes developers, testers, managers, designers, and — highly important — the customer. If a certain skill is required, somebody with this skill is taken on board. If someone is no longer necessary, this person should not continue to be part of the team. This approach is not only about having the right competences available, but is also about creating a *team spirit* to keep members committed.

Informative Workspace. The team's workspace should reflect what it is working on. Ideally, it should be possible to give a project overview to an outsider within

15 seconds. That can be supported by having large, visible charts, in the format of a physical story-board and attached story cards showing project progress. The informative workspace also includes measurements to serve human needs by, e.g., providing drinks, snacks, and a clean and friendly environment.

Energized Work. Staff should only spend those hours in the office during which they can be productive. Long office hours (more than 40 hours per week) should therefore be avoided and people encouraged to stay at home when they are ill.

Pair Programming. "In pair-programming, two programmers jointly produce one artifact (design, algorithm, code, etc.)" [88]. Both sit in front of the same monitor(s). Alternately one of them is the *driver*, physically coding or designing, using keyboard and mouse while the other (the *observer*) is able to think ahead more strategically and to check code on-the-fly. Both partners constantly review the other's output. During a work day the two partners switch roles several times. Partners also frequently change over time, usually when some relatively independent task is completed.

A controlled experiment[11] by Robert Kessler investigated the economics of pair programming. In a software engineering course, 13 advanced individuals and 14 pairs had been asked to code a class project. The programs produced by pairs passed about 14 % more test-cases than the ones created by individuals (89.13 % for pairs versus 75 % for individuals). While some individuals where not able to complete the course, all pairs finished their software. The pairs were also able to complete their assignments about 40-50 % more quickly. Also, personal satisfaction significantly rose when working in pairs.

Stories. Stories or *User Stories* are a way to capture wishes articulated by customers. They have a short name and a brief explanation of what their goal is. They commonly follow a notation like *As a <user-type> I want to <goal> so that <reason>*. User stories are limited to a few lines, so that they fit on a small paper note card. Ideally, they are written by customers themselves. Development teams estimate possible efforts early on to give customers a rough picture of each story's impact on the project.

Weekly Cycle. Weekly releases follow the natural working schedule. The week can be started by reviewing current progress, and breaking the scheduled stories

[11]For more details, read [88]

into tasks. For these tasks, automated tests can then be written and executed towards the end of the week after coding the related functionality. Weekly planning acts as a tool against over-planning. It keeps the team focused on the top priorities.

Quarterly Cycle. The quarterly cycle is a recommendation to regularly review accomplished work from a strategic position. This is done to keep the high level system structure aligned with long-term goals and priorities. A quarter is chosen as it is a natural and widely shared timescale.

Slack. *Slack* is the practice of always planning a certain number of tasks that might be dropped if time becomes critical. The reason for including slack is to honestly and transparently communicate the customer that workload estimates can never be 100 % correct. The slack can also serve as a *task reserve*, to be finished if there is some time left over.

Ten-Minute Build. XP recommends that it should always be possible to automatically build the whole system and run all of the tests within ten minutes. If the process takes longer, something has to be altered so that it again falls into the ten minute frame. The reason behind this idea is that everything taking longer than ten minutes is very likely to be either seldom, or only partially, executed.

Continuous Integration. Changes are tested and integrated after no more than a couple of hours. With distributed development, integration becomes increasingly unpredictable and complex. Building the whole system on every check-in, or on a, for example daily basis shows errors early and eases isolating their source as the single malicious change is comparatively small.

Test-First Programming. Also known as test driven development/design (TDD)[12], it means writing a test for the functionality going to be developed. This approach is chosen to solve the following problems:

- While developing, it is tempting to include code *just for the case*. Strictly writing code to pass tests prevents developers from falling into this behavior.
- Having problems with writing a test case often indicates problems with overall design. “Loosely coupled, highly cohesive code is easy to test” [15, p. 50].

[12]For test driven development, e.g. see http://www.testdriven.com/modules/news/

- Creating test cases justifies coding this functionality and creates trust.
- TDD creates a work rhythm: test, code, refactor, test, code, refactor.

Incremental Design. Design decisions are made at the last possible moment. All other techniques described in this section also serve the purpose of allowing this practice. With requirements frequently changing over the project lifetime, the design has to be incrementally adopted too. Incremental design encourages the team to primarily think about what has to be done soon.

3.3.3 Corollary Practices

This paragraph briefly lists and describes all the corollary practices.

Real Customer Involvement. Letting customers be part of the (e.g., weekly/quarterly) planning activities can benefit the supplier.

Incremental Deployment. Big deployments are risky and involve high effort. Gradually replacing legacy systems with new versions mitigates this risk, even at the cost of running two systems in parallel.

Team Continuity. Productive teams should be kept together as relations are an important factor.

Shrinking Teams. As teams become more capable and experienced, workload is kept constant but team size is reduced.

Root-Cause Analysis. When a defect is found, both the defect and the cause are eliminated.

Shared Code. Code is not *owned* by the individual programmer but by the whole team. Everybody is responsible for fixing errors wherever found.

Code and Tests. Only code and tests are stored as permanent artifacts. All other documentation is generated from these sources.

Single Code Base. Whenever possible, there should only be one actively maintained code base. Branches are only temporary measurements and are quickly merged into the main code tree.

Daily Deployment. New software is put into production on a daily/nightly basis. This practice depends on many prerequisites being met and is therefore listed as corollary.

Negotiated Scope Contract. Contracts include fixed time, costs, and quality but allow ongoing negotiation of the precise scope of the system.

Pay-Per-Use. As mentioned in [15, p. 70], "connecting money flow directly to software development provides accurate, timely information with which to drive improvement."

3.4 Scrum

The Scrum approach was developed by Ken Schwaber[13], Jeff Sutherland[14] and Mike Beedle[15]. In 2001, these developments were summarized in "Agile Software Development with Scrum" ([16]). Scrum itself is named after a game situation in rugby, where players gather in a tight formation. Being part of the Agile Alliance (cf. Section 3.1.4), Scrum shares its vision of how software development should be organized.

Scrum is an iterative software development framework. It consists of few but clear rules, and knows only three different roles: the *Product Owner*, the *Team*, and the *ScrumMaster*. An overview of how a Scrum project is run is depicted in Figure 3.6. The Product Owner is responsible for the economical goals of the project [89]. He is creator of the *Product Backlog*, a to-do list of what is going to be implemented in the future. Implementation is carried out in cycles, so called *Sprints*, with a duration of 1-4 weeks. Every sprint terminates at a fixed point in time and produces a visible, usable, deliverable product [90]. A Sprint is kicked off in a *Sprint Planning Meeting* where the tasks for the next cycle are agreed. These tasks fill the *Sprint Backlog*. The Team is responsible for developing the committed functionality. As Teams are

[13]Ken Schwaber: `http://www.scrumalliance.org/profiles/7-ken-schwaber`
[14]Jeff Sutherland: `http://www.scrumalliance.org/profiles/70-jeff-sutherland-phd`
[15]Mike Beedle: `http://www.mikebeedle.com`

self-managed, self-organized, and cross-functional, they are responsible for figuring out how to turn the Product Backlog into an increment of functionality within an iteration [91]. They have to manage their own work accordingly. The ScrumMaster acts as a coach for the Team and is responsible for everybody complying with the Scrum process and rules. At daily Standup Meetings, named *Daily Scrums*, ScrumMaster and Team get together to discuss yesterday's progress and today's tasks. At the end of the Sprint, a *Sprint Review* meeting is held [91]. At this time-boxed meeting, the Team presents the Product Owner and other potential stakeholders with what was developed during the past Sprint. Before the next Sprint Planning Meeting, the ScrumMaster and Team hold a *Sprint Retrospective* meeting. Here, the Team reflects on the finished Sprint in preparation for the next one.

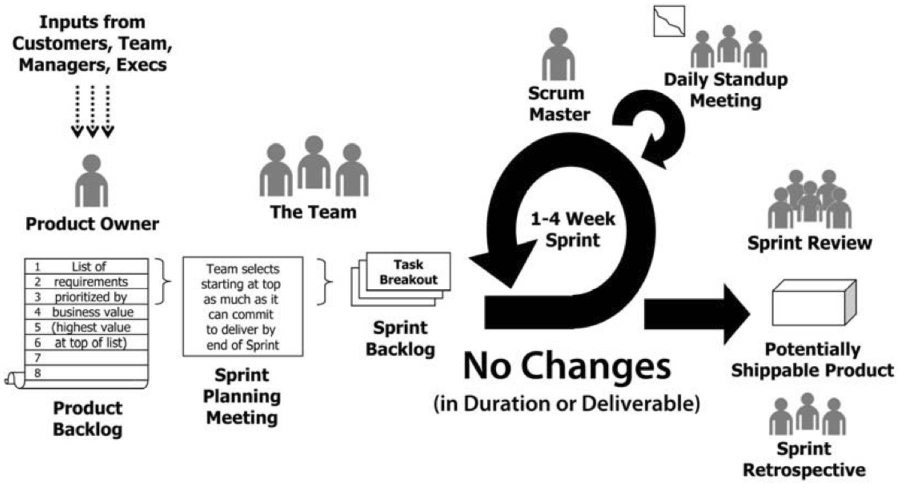


Figure 3.6: Scrum Overview [92]

3.4.1 Project Prerequisites

To run a Scrum project, the project stakeholders must answer three questions [16]:

- At the end of the project, what result can be expected by the project funders?
- Which results can be expected at the end of each Sprint?
- What makes the project a valuable investment for those who fund the project and what can increase their confidence in the abilities of those proposing the project?

To answer these questions, all stakeholders must share a common vision and have to agree on a Product Backlog (cf. Paragraph 3.4.4). The vision declares why the project is set up and what a successful completion should change. The Product Backlog is the foundation for potential Sprints and releases.

3.4.2 The Sprint

All work (such as writing code, designing, documenting) in Scrum is done within Sprints. Figure 3.6 illustrates how the Sprint fits into the overall Scrum process. Following an incremental approach, each Sprint represents one cycle in the software development process. Therefore, a Sprint could also be named an iteration [89]. The goal of each iteration is to create an increment of product functionality [91], potentially ready for delivery. All features included in this release are listed in the Sprint Backlog and defined in the Sprint Planning Meeting. At this meeting, the Sprint's duration is also defined, which may vary between one and four weeks (or up to 30 days). This timeline may not be changed during the Sprint. Also, any changes of requirements are deferred until the next Sprint. If through external influence, there appears to be a significant change in priorities, and continuing on the current work would thus mean wasting the Team's time, it is possible to terminate the Sprint early [92]. Consequently, the Team stops working on tasks from the current Sprint Backlog, and a new Sprint Planning meeting is scheduled.

The Sprint Planning Meeting. The Sprint Planning meeting is a roughly one day long meeting where the participants decide on which functionality from the Product Backlog is transferred into the Sprint Backlog for the next iteration. It is attended by the Product Owner, the ScrumMaster, and the Team. In coordination with the ScrumMaster, other stakeholders may also attend the meeting. The Product Owner proposes the goal(s) for the Sprint, and the Team identifies the related tasks, estimates effort and dependencies, and discusses what will be implemented by whom [89]. The ScrumMaster acts as a moderator. The meeting attendants start with the first requirement on the Product Backlog (that with the highest priority) to break it down into the activities needed to fulfill it. For each activity, the Team estimates its duration and self-assigns responsibility to one of its members. The meeting ends when all parties are committed to a realistic goal and a Sprint Backlog is generated.

The Daily Scrum. The Daily Scrum is a time boxed [91] meeting, lasting for 15 minutes at most. All Team members are required to attend. Each Team member answers the following three questions [91, 90]:

- Relative to the Backlog, what have you accomplished since the last Daily Scrum?
- Are any obstacles in your way, impeding you from completing this work?
- Relative to the Backlog, what do you plan to accomplish between now and the next Daily Scrum meeting?

There is no room for other discussions in this meeting. The ScrumMaster ensures that nobody breaks this rule, and records all obstacles the Team is experiencing.

The Sprint Review. At the end of the Sprint, the Team demonstrates the product increment. In addition to the Team, the Product Owner, and the ScrumMaster, additional stakeholders (such as users, customers, or other departments) may participate. Their attendance allows them to receive unfiltered and direct feedback [89]. The Product Owner checks whether the functionalities built meet the agreed requirements. Based on this, changes to the Product Backlog can be discussed.

The Sprint Retrospective. After the Sprint Review, the Team gets together to reflect on the last Sprint. This meeting is attended by the Team, the ScrumMaster, and optionally the Product Owner. As all other Scrum meetings, this one is also time boxed. In this case about three hours are scheduled. This meeting provides answers to two main questions as stated in [91]:

- "What went well during the last Sprint?"
- "What could be improved for the next Sprint?"

The ScrumMaster presents (e.g., using a whiteboard) everybody's statements for the Team to review so they can be discussed in detail. The main purpose of a Sprint Retrospective is to continuously improve performance and satisfaction.

3.4.3 Roles

There are three different roles in Scrum: the Product Owner, the Team, and the ScrumMaster. Typically one Product Owner is working with one Team, supported by one ScrumMaster. The ScrumMaster also only serves one Team.

The Product Owner. The Product Owner represents the customer inside the company and is integrated into software development. In some cases, customer and Product Owner are one and the same person, but often the latter is part of the internal organization. The Product Owner is ultimately responsible for achieving maximum business value, translating customer input into a prioritized list (the Product Backlog). According to [89], the Product Owner has the following functions:

Describing and Managing Requirements. The Product Owner is responsible for the collection of customer requirements and their descriptions, as well as input from other team members. This information is translated into a prioritized list — the Product Backlog.

Release Management and ROI. The Product Owner's main objective is maximizing the *Return on Investment (ROI)*. Therefore, the value created by the project should be as high as possible. It is the Product Owner's decision when, with which set of features, and at what cost the product is delivered. In contrast to the more traditional project/business owner, the leadership of the development project is not delegated to a project manager or team lead.

Close Collaboration with the Team. The Product Owner collaborates closely with the Team by helping to understand and express customer requirements. This includes preparing detailed requirement descriptions and tracking project progress.

Stakeholder Management. Negotiating with all project stakeholders (marketing, IT, sales, etc.) and integrating their views into the project is another major responsibility of the Product Owner.

The Team. The Scrum Team carries out the work required to realize all functionalities it has committed to deliver. The Team is multifunctional in the sense that it includes all the expertise necessary to deliver the project increment at the end of each sprint and that it possesses project management capabilities to manage itself. As with XP (cf. Paragraph 3.3.3), team continuity is regarded as very important. The recommended team size ranges from 5-9 [16] to 3-8 [93].

As communication is crucial, the Team's work facilities should be open-plan and provide space to display information on posters and whiteboards. The posters should at least show the current Sprint Backlog, and the Sprint Burndown chart [89].

The ScrumMaster. Compared to traditional roles like *project manager* or *team lead*, the ScrumMaster is much more a *coach* and *change agent* [16]. Especially when

Scrum is new to the organization, being a ScrumMaster should be a full-time job. If that is not possible, a Team member (but never the Product Owner) might serve as such [92]. The ScrumMaster is responsible for the success of the project [91]. He/she helps to ensure project success by supporting the Product Owner in selecting the most promising features to be part of the Product Backlog. The ScrumMaster's main responsibilities are:

Establish and Teach Scrum. The ScrumMaster is the role where in-depth Scrum knowledge is most critical as it is the ScrumMaster's responsibility that everybody understands and follows Scrum practices. A ScrumMaster drives Scrum adaptation within the organization and his/her duty is to try to establish a way of thinking and acting in order to make agility possible.

Serve and Support the Team. Instead of managing the Team, the ScrumMaster "serves" [92] it and protects it from outside interference. A ScrumMaster has no direct authority to instruct Team members and has no staff responsibility. He/she tries to resolve any impediments that might disturb the Team from working. The ScrumMaster's role is "similar to a personal trainer" [17, p. 117] who helps choosing exercises and performing them correctly.

Enhance Communication. The ScrumMaster constantly communicates with the Product Owner and the Team to ensure that both parties understand each other. This might include acting as a moderator between these stakeholders.

3.4.4 Artifacts

Scrum relies on a small but important number of artifacts that are created throughout the process. The following paragraphs give a brief overview.

Product Backlog. The Product Backlog is a central artifact in Scrum. Owned and maintained by the Product Owner, it contains a prioritized list of everything ever thought of as being a valuable part of the final product [16]. This prioritized feature list contains user requirements but also non-functional ones or administrative tasks. There is one Product Backlog per Scrum project. It does not contain activities, as these are only identified during the Sprint Planning meetings. The Product Backlog changes throughout the project. *User Stories* are a common way of expressing requirements in a short and simple manner. More on user stories can be found in Paragraph 3.3.2. As the Product Backlog is initially incomplete [16], new stories are added, existing ones are removed or revised, and their degree of detail increases. All entries in the Product Backlog contain an estimated effort, usually expressed in *story points* or *ideal man-days* [93, p. 11]. Both methods are used to weigh the different requirements in a project against each other.

Product Backlogs can be further broken down into *Release Backlogs*, containing all information selected for a release.

Sprint Backlog. The Sprint Backlog is created after the Sprint Planning Meeting and before the first Daily Scrum [93]. It contains all activities to which the Team has committed itself at the Sprint Planning meeting. The effort of all activities is estimated in hours. The level of detail dictates that each task should be completed in roughly four to sixteen hours [16]. No items may be added or removed from the Sprint Backlog until the Sprint ends, unless time allows the inclusion of new activities or existing ones fall out of scope. Nevertheless, it is constantly changing as Team members commence to work on their tasks and update descriptions and estimates on a daily basis.

Changes Report. At the end of the Sprint Review, the Product Owner creates a Changes Report, comparing the Product Backlog *before* and *after* the Sprint. It summarizes everything that happened during the Sprint, what was highlighted at the Sprint Review, and what adaptations have been made to the project in response to the inspection done there [91]. The Changes Report is a tool to report progress to other stakeholders, such as management or customers.

Burndown Report. The Burndown Report demonstrates the Sprint progress. On the x-axis it lists all working days and the y-axis displays the effort in, for example, story points or ideal days. Every day, the Team enters its progress into the Sprint Backlog and the amount of remaining effort ends up in a point on the Burndown Report. This can then be compared to the ideal, linear *Burndown* as depicted in Figure 3.7.

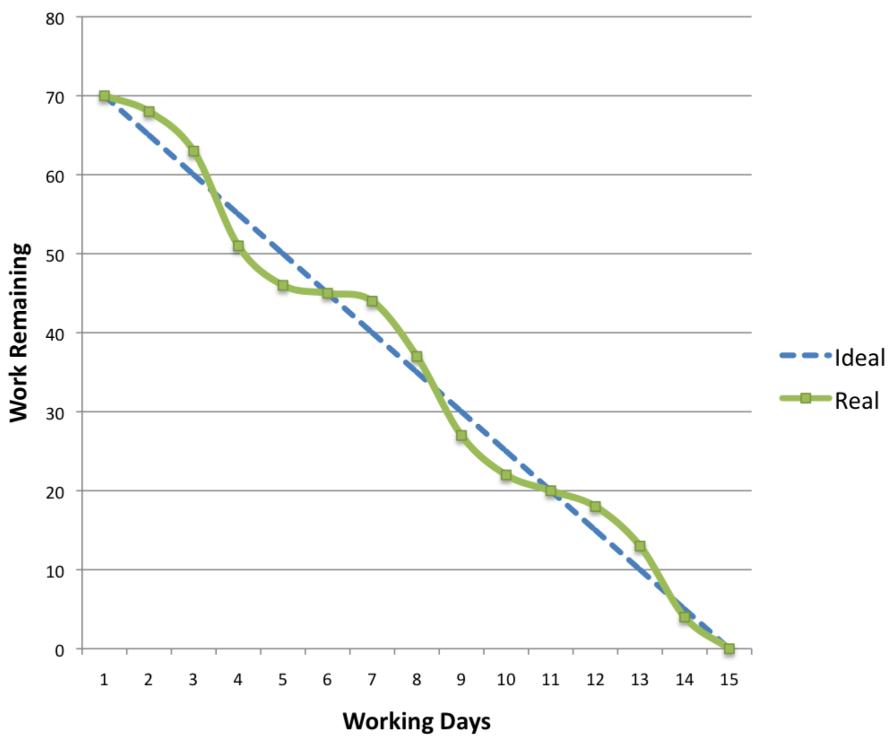


Figure 3.7: Sample Burndown Report

The Burndown Report gives a realtime overview of the Sprint progress. It therefore helps the Team with self-organization and can be used to give stakeholders an up-to-date project health status.

The Impediments Chart. This report contains a list of factors/issues impeding the project. Each impediment is formed of a short description and the date on which it was identified. Identifying impediments is part of the Daily Scrum and the ScrumMaster is responsible for recording them. When an impediment gets rectified, this is noted together with the date of remediation. The chart helps to keep track of these difficulties and shows concluding trends.

Chapter 4

Classification and Applicability

The preceding chapters describe what Web engineering is about and provide insight into a number of development approaches ranging from traditional concepts to very recent ones following agile patterns. The challenge of this chapter is to sufficiently compare these methodologies following shared comparison standards. Subsequently, a judgement about their applicability for Web engineering can be reached. It is, however, not part of this chapter to provide an in-depth analysis of similarities, differences, shortcomings, and application areas, as the existing literature [94, 19, 95, 96] provides a good overview.

4.1 Classifying Software Engineering Processes

The comparison approach used in this chapter combines components and results from [19], [97], [98], and [94]. These methods are applied on XP, Scrum, and RUP. The conclusion is based on results described in the named literature as well as on findings gathered when elaborating the descriptive chapters of the paradigms presented within this thesis. The comparison indicates which characteristics identify the methodologies and which parts of the development lifecycle are covered by each of them.

4.1.1 Describing the Characteristics

The first part of the analysis concentrates on the characteristics of the mentioned software development paradigms. Following existing, well documented examples [98, 99, 94], the methodologies can be distinguished from each other and covers a generic overview as well as a more detailed investigation.

4.1.2 Describing the Lifecycle

The second part divides the development project into nine phases, loosely following the Waterfall approach. These phases cover the full development lifecycle from concept creation to the usage of the developed system. This approach is introduced in [19]. The phases covered are: Concept Creation, Requirements Specification, Design, Code, Unit Test, Integration Test, System Test, Acceptance Test, and System in Use.

Support for each of these phases is divided into three different categories. This allows differentiating to which degree these phases are covered. The criteria are:

1. Does it include project management practices?
2. Is an adequate process proposed and described?
3. Does it offer concrete guidance for handling scenarios in a certain phase?

The results of this review are depicted in Figure 4.1.

4.2 Classifying Methodologies

This section shows the results of evaluating XP, Scrum, and the RUP following the described approach. The results are based on descriptive (books and papers presenting an approach) and on investigative (comparing and judging methodologies) approaches.

4.2.1 Characteristics

The first part of the analysis concentrates on the differences between agile and non-agile approaches in general and between the individual methodologies in particular. Table 4.1 presents a generic overview on where light- and heavyweight solutions put their emphasis.

Characteristics	**Agile methods**	**Plan-driven methods**
Approach	Adaptive	Predictive
Primary objective	Early value	High assurance
Return on investment	Early in project	End of project
Emphasis	People oriented	Process oriented
Size & complexity	Smaller teams and products	Larger teams and products
Developers	Agile, knowledgeable, co-located, and collaborative	Plan-oriented; adequate skills; access to external knowledge
Customers	Dedicated, knowledgeable, co-located, collaborative, representative, and empowered	Access to knowledgeable, collaborative, representative, and empowered customers
Requirements	Largely emergent, rapid change	Knowable early; largely stable
Architecture	Designed for current requirements	Designed for current and foreseeable requirements
Refactoring	Inexpensive	Expensive

Table 4.1: Generic Characteristics [98, 99, slightly adapted]

These findings to a large degree match the results in [95], where common characteristics of agile methods are summarized. All of these methodologies have been published between 1995 and 2002 in the USA and UK. Based on the assumption that projects undergo constant change, they resemble objectivist methods providing technical solutions to address business problems. None of them have been developed with much academic background, but are practitioner based instead. Typically, agile paradigms distinguish between a project manager and a developer perspective. They enforce incremental development with iteration cycles of about 1 month. Development is done in small, *empowered* teams of about 3-10 programmers and involves active user/stakeholder participation, feedback, and learning. As communication between all project participants is critical, frequent meetings are a common characteristic. The ultimate goal of agile methodologies is to create working software as the main product of development whilst modeling techniques are not mandated and documentation is minimized.

In contrast to Table 4.1 which introduces a generic view on the matter, Table 4.2

focuses on the individual paradigms. Despite XP and Scrum sharing many fundamental characteristics, they are still two distinct approaches and value specific properties differently.

Characteristics	**XP**	**Scrum**	**RUP**
Iteration length	Short	Short	Long
Importance of planning	Low	Medium	High
Importance of documentation	Low	Low	High
Collaborative	High	High	Medium
Refactoring enforced	High	Medium	Low
Unit Testing enforced	High	Medium	Medium
Functional Testing enforced	Low	Low	High
Integration of changes	High	Medium	Medium
Knowledge sharing	High	Medium	Low
Interaction with end-users	Medium	High	Medium
Team member interaction	High	Medium	Low
Team self organization	High	High	Low

Table 4.2: Detailed Characteristics [94, modified, extensively adapted]

Table 4.2 indicates that XP is the most *radically* agile of the methodologies listed, whereas the RUP clearly represents a more planned approach. The result also suggests that Scrum and XP share a lot of characteristics.

4.2.2 Lifecycle Coverage

In this section, the methodologies are compared based on their differing lifecycle support. Thus, for each method, the software lifecycle coverage — including project management support, process guidance, and practical support — is evaluated. Figure 4.1 shows that there are three different aspects to be considered when inspecting the software development lifecycle.

Project Management. As Figure 4.1 shows, XP does not address the project management perspective. In contrast, one of the main focuses of Scrum is to support this role with tools and a theoretical background. The RUP also provides the necessary coverage, even extending that of Scrum.

Process Description. XP describes in detail how the software development process should be conducted and concentrates on concrete guidance. Scrum also covers most parts of the process but on a more abstract level. Again, the RUP gives the most complete picture.

Practical Support. This is the main strength of XP. Its 24 practices provide detailed descriptions of how to handle the different tasks in software development. Scrum follows a more abstract approach, focusing on requirements description and collection, and customer integration. RUP, being a fully featured methodology, covers all these aspects.

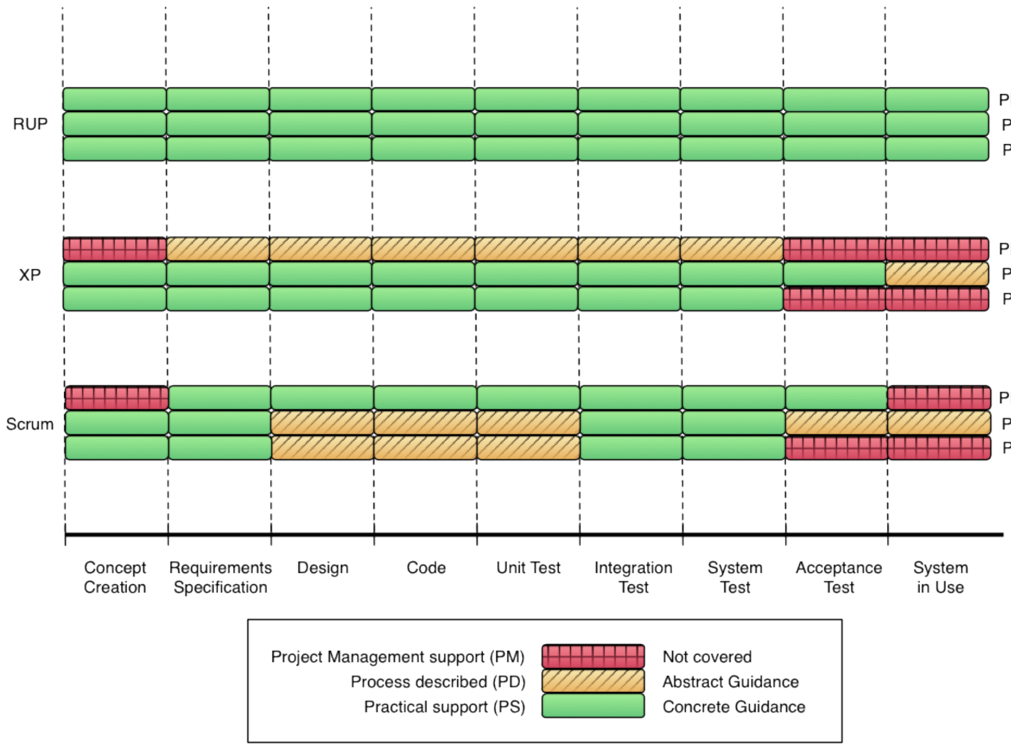


Figure 4.1: Lifecycle Coverage [19, modified]

4.3 Applicability for Web Engineering

Having characterized Web engineering and software development processes, which kind of methodology should be chosen? Traditional software development practices — such as the Waterfall model (cf. Section 3.1.2) — were originally proposed in the 1970s. As a part of a then very young industry, they reflect an environment much different from what is reality today. Dominated by substantial organizations such as the military or big corporations, developers wrote large-scale, centralized systems, usually to achieve automation [100]. User interfaces — if they existed — were command line based. Many of the approaches that historically seemed natural, merely translate into modern application development or Web engineering. The following paragraphs aim to discover which pieces of the more modern approaches described in Chapter 3 are applicable in these scenarios. Based on the information collected in previous chapters, this part of the thesis looks into the applicability of these software development paradigms for Web engineering.

The following sections are based on the information given in this thesis and on a method described in [20], where the authors propose a way of evaluating the kind of development method best suited to a project. The suitability of a project for agile development is dependent on a number of factors. These are described in the following paragraphs.

4.3.1 Domain Factors

Not all domains are appropriate for applying agile management principles. The Internet application domain is stated as being particularly [95] suitable. Other factors relate to technological requirements, such as automated testing and the utilization of object-oriented development.

Chapter 2 of this thesis provides a detailed list of peculiarities that distinguish Web engineering from other disciplines. The book “Extreme Programming for Web Projects” [6] provides a similar but shorter list, which in parts corresponds to that mentioned in this thesis. The subsequent paragraphs summarize the most important points and investigate whether the found specifics influence the choice between conducting a project in an agile or traditional manner.

Teams. Described in Sections 2.4 and 2.6.2, Web engineering consists of the disciplines Media Design, Project Management, Software Engineering, and Content Creation. Two of the participating entities — Project Management and Software Engineering — are common participants in almost every software development project. The other two, however, are particular to Web engineering and deserve separate attention as these roles are typically not covered by theories centering around software development. In [6] this is referred to as a multidisciplinary team covering a “myriad of new disciplines”.

Common to all agile methods is the strong emphasis on personal communication and co-located teams. Both factors help with breaking down the psychological and professional walls between people of different specializations. The strong interdisciplinarity that is required by Web projects make this kind of interaction highly important and may often replace formal in-detail documentation.

Support for Multiple User Environments. The background has been described in Sections 2.5.3 and 2.6.3. It means that one version of a Website must be able to support multiple types of devices and a heterogenous user base. This can be seen as an additional technical constraint that has to be tackled but it should not affect the development process itself significantly. In any case, customers might not be aware of this device heterogeneity so the targeted user audience (and their technical equipment) must be clearly specified.

Usability & Testing. Described in Sections 2.4.2 and 2.6.3, this point explains that Web interfaces are different from desktop GUIs as they may be designed quite freely, following only loose patterns. This makes usability testing a very important component of the software development cycle. Additionally, the interface is usually not laid out by a backend programmer but by people that combine artistry with specific coding knowledge such as HTML, CSS, and Javascript. As these creatives usually do not come from a software development background, their acceptance of strict processes is not very sound. This opens the field for agile practices as they tend to give the individual more freedom. Integrating design, usability testing, and customer approvals early in the process is necessary as testing at the end “doesn’t work” [17]. Scrum for instance makes testing a central practice and part of the development process.

Rapid Deployment. As mentioned in Section 2.6.4, Web applications quickly become obsolete. Updated versions have to be deployed rapidly. Integration of new code or designs is a regular task that happens frequently. This reflects agile approaches. In XP, team programming is a divide, conquer, and integrate problem [15]. Some of this integration effort can be automized by taking advantage of unit testing and build automation. As rapidly changing requirements lead to constantly changing software (or Websites), agility is certainly an important factor here.

Customers. Web engineering is often positioned at the interface between marketing and IT. These two worlds frequently do not have much in common and also might not have a deep understanding of what a good Web product is about. As customers are less experienced with Web applications than with traditional software or marketing channels, it is vital to find a common language to manage and understand expectations. This requires a lot of personal interaction which reflects what agile approaches suggest. Prototyping and releasing software early are agile ways of transporting information in a way that is comprehensible for all participants.

4.3.2 Organization Factors

In oder to leverage the effects of agile methodologies, an organization has to meet certain criteria. This largely concentrates on social factors, such as trusting, collaborative, and competent interaction, face-to-face communication and a general informality in daily interaction. The organization should be willing to empower people, following a management style which combines leadership with collaboration, which in turn requires an environment that is flexible, encourages participation and social interaction.

4.3.3 Project Factors

The next factor relates to the project itself. Research [101] shows that theoretical entitlement seems to differ from reality. According to this analysis, projects with stable requirements and a well-established architecture are reported as especially suitable for agile methods. Small, co-located teams, qualitative project control, and a project manager acting as a facilitator are further factors to be named.

4.3.4 People Factors

Agile methods put heavy emphasis on people[1]. Therefore, the participating characters have to meet certain conditions. Developers should be experienced and capable of forming self-organizing teams. On the other side of the spectrum, agility needs customers that are collaborative, representative, authorized, committed, and knowledgeable. In the best scenario, the customer would be an on-site usage expert, intensively collaborating with the development team.

[1]See the Agile Manifesto at Section 3.1.4

4.4 Critical Remarks on Applicability

The described factors clearly suggest that agile practices are favorable for the described purpose. The more strict types do not provide the flexibility desired for this fast moving environment. The different paradigms provide a varying level of coverage for the software lifecycle aspects. Depending on the focus of a project, a method based on any of the discussed frameworks can be chosen whilst keeping in mind that a fully fledged RUP rollout will cripple agility. The next few paragraphs summarize general thoughts on software development methods and on their applicability for Web engineering.

RUP. There are many reasons why the RUP has become so popular and in many areas a de-facto industry standard. According to [102],

> "it combines recognized best practices such as adaptive, iterative, and risk-driven development; it has been developed by world-class leaders with experience in both small and large systems development; it is flexible in its application and extension; and it has been coherently documented in both print and the online RUP product".

Despite this heritage and whilst containing agile elements [20], the RUP is still said to be "overly complex" [103]. The methodology itself tries to cover almost every aspect of the software development lifecycle. This makes the RUP very hard to understand completely. The inherent complexity and sheer volume of information make it very difficult to actually find the pieces of the process that are actually valuable for a specific implementation. Not all pieces of the process follow the same standard of quality. In [103], the author states that "it takes a very intimate knowledge of the process framework and some very real experience using it to determine what to use and what to completely ignore". RUP, therefore leaves a lot of tailoring to the user entirely, which raises the question how much of the RUP can be left out, with the resulting process still being *RUP* [19].

Documentation and — if the process is strictly followed — the approach itself add a lot of complexity to the RUP. For example, strict execution requires the completion of around a 100 work products [76]. An example of the consequences this might lead to can be found in [95], where one of the survey participants mentioned that, over five years, "using RUP was prohibitive due to the time and cost of generating the specified documentation".

Generally, the RUP does not seem to be very suitable for smaller projects where delivering the product is often more valuable than any strategic planning. It is very hard to imagine the average Web agency follow this strict framework. The possible benefits would not sacrifice the overhead. Only very complex and/or high-risk projects can possibly require such strict processes, especially as the latter may often be conducted in sensitive (high security) environments.

Agile Processes. In the past years many papers, books, and journals have been published that describe the adoption of agile methods. However, most of these articles on adoption are experience reports, often in the form of anecdotal success stories or lessons-learned from organizations that have adopted one of the processes for a project [104]. These reports often lack empirical evidence for efficacy, benefits, and problems, therefore information about the most appropriate project environment for agile methods is scarce [95].

A study carried out by D. Reifer [105, 101] in 2002 revealed that projects utilizing agile methods are small (10 participants or less), often pilot projects and most were Web-based, quick-to-market applications. This study involved 31 projects, 14 claiming to use agile methods. To identify costs and benefits, a small number of measurements was taken in some of the projects, assisted by further qualitative assessment. The measurements showed productivity improvements of 15% to 23% and cost reduction of 5% to 7%. The small team size noted coincides with other sources [106] that state the conclusions of agile practitioners and other researchers that the agile value set and practices best suit co-located teams of about 50 people or fewer who have easy access to user and business experts and are developing projects that are not life-critical. Although the agile value set might be adopted (and may be adapted) under other circumstances, the boundary conditions for truly agile behavior seem to be fairly set. This makes scalability, or the lack of, a real issue. The strong focus on face-to-face communication in agile processes makes it especially difficult to scale up to larger projects. Even XP founder Kent Beck admits that size clearly matters and that it is probably not possible to run an XP project with a lot more than twenty programmers, suggesting a maximum number of ten [15]. As shown in a 50-person XP case study [20], larger agile projects need to adopt traditional plans and specifications in order to handle the increasingly complex, multidimensional interactions among the project's elements.

Another point negatively affecting scalability is that current experience [20] indicates that there are limits to low-cost refactoring as projects scale up. The most serious problems that can arise when a software architecture is not sufficiently planned are problems known as "architecture breakers" [20]. These potentially highly expensive problems can occur when early, simple design failures result in changes either later within the same project or a follow-up project, that cause breakage in design or functionality beyond the ability of refactoring to handle. Naturally, a larger codebase means more risk for architecture breakers and more effort to fix them.

For the average Web project, agile approaches provide a lot of valuable input as their core competence is about managing the flexibility often needed in Web engineering. The close collaboration with the customer guarantees the feedback that is especially critical for situations where visual design is a major factor. Also technical recommendations such as continuous integration make much sense in order to stay in touch with customer requirements.

Even for more standard solutions without much development involved even following agile approaches can be unnecessary and standard project management is sufficient. When launching the finished product, errors can be immediately visible to a very large audience. This makes the launch phase very important. However, this phase is missing in most agile methodologies, including Scrum and XP (cf. Figure 4.1).

4.5 Deciding in Uncertainty — a Risk based Approach

In [20], B. Boehm and R. Turner propose a system for deciding whether to follow an agile or more "plan-driven" route for a project. This solution uses project factors, such as size (number of involved people), criticality, personnel skill level, dynamism (requirements change per month), the organization's culture (chaos versus order), and especially project risk to determine the most suitable type of method to use for a particular project [95]. It tries to answer the question "How much planning and architecting is enough?" to find the right balance.

The approach categorizes the possible risks regarding three different categories [20]:

> "Environmental risk
>
> - E-Tech. Technology uncertainties
> - E-Coord. Many diverse stakeholders to coordinate
> - E-Cmplx. Complex system of systems
>
> Agile risks: risks that are specific to the use of agile methods
>
> - A-Scale. Scalability and criticality
> - A-YAGNI. Use of simple design or YAGNI (*You Ain't Gonna Need It*)
> - A-Churn. Personnel turnover or churn
> - A-Skill. Not enough people skilled in agile methods

Plan-driven risks: risks that are specific to the use of plan-driven methods

- P-Change. Rapid change
- P-Speed. Need for rapid results
- P-Emerge. Emergent requirements
- P-Skill. Not enough people skilled in plan-driven method"

Figure 4.2 depicts the process model behind it.

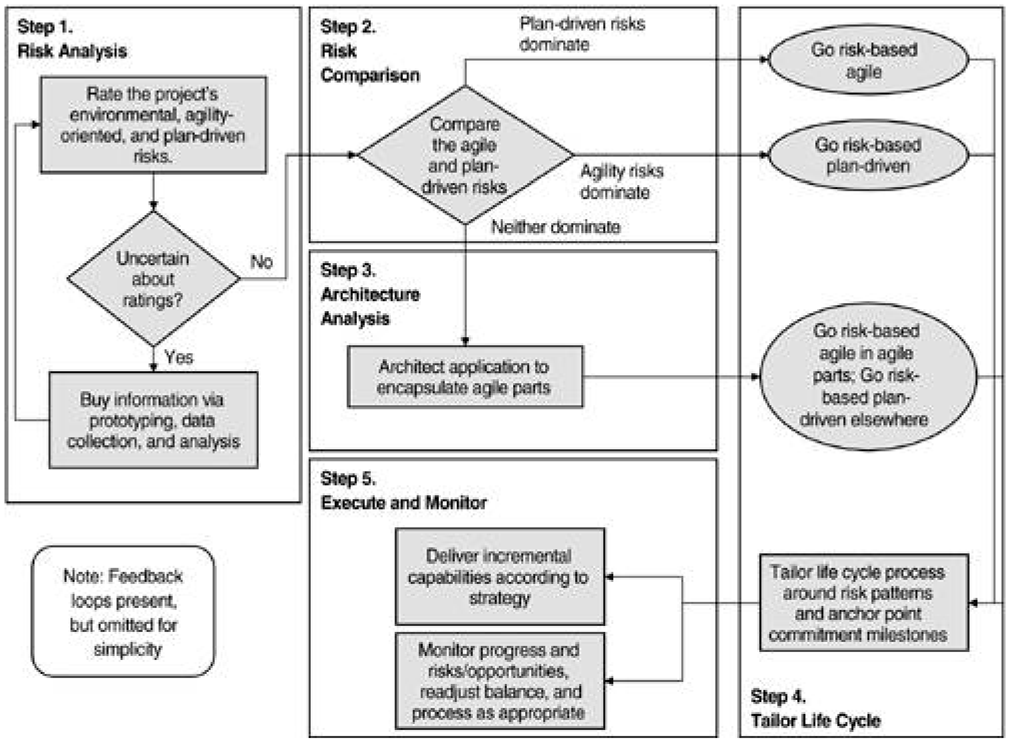

Step 1.
Risk Analysis
Rate the project's environmental, agility-oriented, and plan-driven risks.
Uncertain about ratings?
No
Yes
Buy information via prototyping, data collection, and analysis
Step 2.
Risk Comparison
Compare the agile and plan-driven risks
Plan-driven risks dominate
Agility risks dominate
Neither dominate
Go risk-based agile
Go risk-based plan-driven
Step 3.
Architecture Analysis
Architect application to encapsulate agile parts
Go risk-based agile in agile parts; Go risk-based plan-driven elsewhere
Note: Feedback loops present, but omitted for simplicity
Step 5.
Execute and Monitor
Deliver incremental capabilities according to strategy
Monitor progress and risks/opportunities, readjust balance, and process as appropriate
Tailor life cycle process around risk patterns and anchor point commitment milestones
Step 4.
Tailor Life Cycle

Figure 4.2: Summary of the Risk-Based Method - Model [20]

In total, five steps are part of the method. Each of the steps requires that the executor collects information or finalizes a decision. Every subsequent step depends on the outcome of its predecessor.

1. Rate the project's environmental, agile, and plan-driven risks. If uncertain about ratings, buy information via prototyping, data collection, and analysis.
2. Make a decision.

   a) If agility risks outweigh plan-driven risks, go risk-based plan-driven.

   b) If plan-driven risks outweigh agility risks, go risk-based agile.

3. If parts of the application satisfy 2a and others 2b, architect the application to encapsulate the agile parts. Go risk-based agile in the agile parts and risk-based plan-driven elsewhere.

4. Establish an overall project strategy by integrating individual risk mitigation plans.

5. Monitor progress and risks/opportunities, readjust balance and process as appropriate.

Often the findings of Step 1 will lead to a final decision without the need of following the whole process. Basing the decision on risk applies when the environment itself is only vaguely determining the outcome. That may apply to completely new organizations or to teams put together for one big project.

Risk and the organization executing the project are just one side of the coin. The other side is made up of the different methodologies themselves. Their main characteristics have already been described in Section 4.2. The developers of the utilized decision model also categorize these. The classification given in Table 4.3 covers two dimensions: *Levels of Concern* defining the organizational scope for which the methodology provides guidance and *Sources of Constraint* which summarizes all the constraints the method puts on the project team. A third category *Life Cycle Activities* has already been covered in Figure 4.1. On a scale from 1 to 5, where 1 represents *very low* and 5 *very high*', [20] assigns Scrum the least number of constraints (1), XP classifies as 2-3 and the RUP as 2-4.

	Levels of Concern					**Sources of Constraint**				
Method	Business Enterprise	Business System	Multi-Team Project	Single-Team Project	Individual	Management Processes	Technical Practices	Risk/Opportunity	Measurement Practices	Customer Interface
Scrum	-	o	o	o	o	+	-	o	-	o
XP	-	o	-	+	+	o	+	o	o	+
RUP	-	o	+	+	-	o	o	+	o	-

+ ... fully supported
o ... partially supported
- ... not supported

Table 4.3: Risk-Based Method - Method Comparison [20, shortened]

4.5.1 Assessment

The risk based approach described above gives a sound and generic schema for classifying projects. Taking this classification method as a basis, it is possible to feed the typical attributes of Web engineering into this model. The basis for that is given in Section 2.2 “Categorization” as part of this thesis. There, Web engineering is subdivided into categories, taking into account novelty and complexity. For the intended purpose here, only complexity is relevant and is taken as factor `A` in this calculation. `A` follows a linear curve with 1 being the lowest number and 8 the highest. The other factor taken into account is risk (factor `B`). Divided into 8 levels, it is weighed higher than complexity as risk is the major source for making a decision according to [20]. This importance is reflected in the value assignment for these 8 levels. Here the Fibonacci sequence is chosen, starting with the third number in this row. These two factors are then multiplied and the result represents the level of planning that should be involved. The higher the value, the more planning is going to be needed. The results of this calculation can be found in Table 4.4.

As Figure 4.3 shows, higher complexity and, more prominently, higher risk lead to higher planning demands. Taking into account that the more complex variations of

			B: Risk							
			low							high
		*	1	2	3	5	8	13	21	34
A: Complexity	Informational	1	1	2	3	5	8	13	21	34
	Interactive	2	2	4	6	10	16	26	42	68
	Transactional	3	3	6	9	15	24	39	63	102
	Service Oriented	4	4	8	12	20	32	52	84	136
	Workflow Oriented	5	5	10	15	25	40	65	105	170
	Collaborative	6	6	12	18	30	48	78	126	204
	Social	7	7	14	21	35	56	91	147	238
	Cloud Based	8	8	16	24	40	64	104	168	272

Table 4.4: Risk-Based Method - Web engineering - Results

Web engineering are usually the youngest types, two possible conclusions can be drawn.

The first one stipulates that Web engineering by trend more and more evolves towards classical software engineering. Still, the factors described in the opening of this chapter apply, but are possibly more and more merged into standard, but flexible, software development approaches. However, these influences require practices which are currently not sufficiently covered by the methodologies described.

The second conclusion from this practice is that *planned* and *agile* are not formal descriptions, but synonyms which rather vaguely describe approaches differentiating from each other in various dimensions. The participants of a huge and critical banking software project might have a totally different view on what these terms mean than a group of university graduates working for a local startup. This makes Step 1 of the risk-based method crucial. Here the domain factors listed in Section 4.3.1 clearly indicate that organizations in the Web environment tend to lean towards agile approaches.

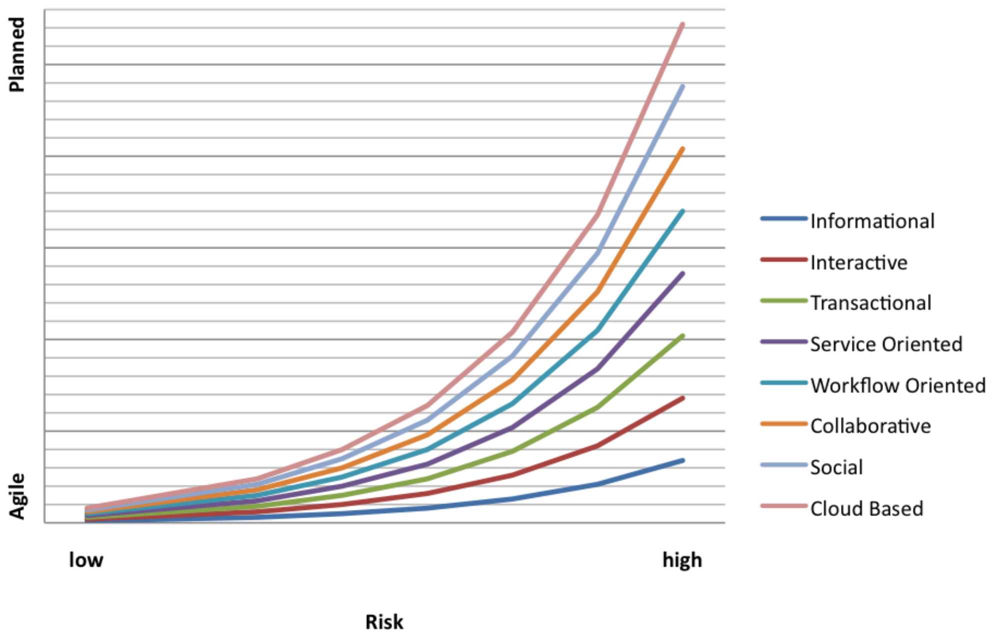


Figure 4.3: Risk-Based Method - Web engineering - Overview

4.6 Combining Methodologies

As the different parts of this thesis show, the various methods focus on different aspects of the software development lifecycle. Here, XP is clearly more focused on practical and descriptive areas, whereas Scrum gives more attention to project management and processes. RUP provides an extensive and detailed coverage of all aspects. Therefore, combining two methods seems to be a valuable idea.

XP and Scrum. As described in [107], Scrum is a product development methodology consisting of practices and rules to be used by all project stakeholders to maximize the productivity and value of a development effort, whereas Extreme programming (XP) is an engineering methodology mainly consisting of practices that support focusing on creating top-quality code. This also reflects the lifecycle related findings in Figure 4.1. As XP lacks management components [19], its practices may be “wrapped” [16] by Scrum to provide a management and process framework. This combines the strengths of both systems.

XP and RUP. Two articles [108, 103] relate to this combination as utilizing RUP to provide a proven framework whose practices are enriched with agile techniques. In summary, this advocates mixing agile with more structured methodologies like the Rational Unified Process as a good way to combine energetic, creative and real-time approaches with structured, more methodical frameworks that can help organizing large-scale projects [103]. The same author emphasizes the complexity of the RUP and how he uses XP to lighten it up.

RUP and Scrum. A combination of these two management focused approaches can possibly be used to free a Scrum Team within a RUP structure from some of the efforts of creating non-code artifacts. Since Scrum teams work very independently, all the artifacts that could be produced by them should be declared optional to reflect the control that teams have in Scrum environments [109].

Chapter 5

Supporting the Web engineering Process

This chapter of the thesis is about adding real world experience to the theory already covered. All content is based on the author's work in a department responsible for e-marketing projects in one of the 10 biggest pharmaceutical companies[1] in the world — the Vienna eMarketing Center (VEC) — and on the insights gained while elaborating the theoretical background. Both parts constantly influence each other, thus the outcome is truly a consequence of both.

5.1 Work Environment

This first paragraphs describe the work environment in which the processes and tools to be described in later chapters are part of daily routine. The VEC was established in 2008. It started with about 10 people in 2008 and consists of about 60 at the time of writing.

Core Business. The VEC's core business is to produce electronic marketing assets for company affiliates around the globe. These assets mainly consist of: websites build with .NET and Java; *eDetails* which are Flash or HTML5 based presentations delivered via the Web, tablets, or iPads; and mobile applications for iOS and Android devices. The target audience is primarily health care professionals (primarily physicians) with supplemental offers for patients and a general audience. Marketing in the pharmaceutical industry is strictly regulated [110], which is especially true for communication targeted at patients. This prohibits product related information from being publicly accessible. In this environment, the VEC covers the full product lifecycle (concept, design, implementation, project management, maintenance, hosting, etc.).

[1]Specific information may be given on request

Department Structure. The VEC consists of a number of specialized departments as illustrated in Figure 5.1. General Management leads the department, Project & Program Management (PM) guides the projects until they launch and also does concept work, Operations (Ops) deals with the hardware and software infrastructure, Project Support (ProS) is responsible for customer support and supportive tasks, and Project Quality (ProQ) performs general and usability testing.

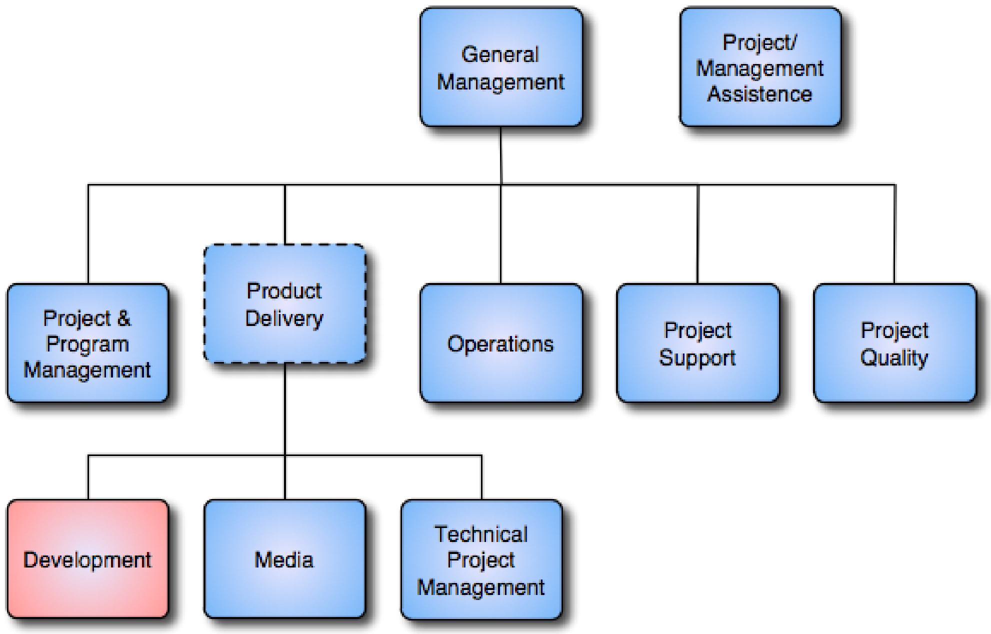


Figure 5.1: eMarketing Center - Structure (simplified)

This thesis focuses on the group marked in red on the chart — Development (DEV). The author of this thesis leads this group. Together with the Media department which does design and its implementation (HTML, CSS, Javascript, video cutting, Flash), the Development team is part of the Product Delivery unit. It is accountable for the implementation and also the maintenance of VEC assets. Thus it covers application architecture design, project management, programming in C#/ASP.NET, Java, Javascript, AS3 (Adobe Flex), Objective-C (Apple iOS[2]), and MSSQL[3]. The maintenance part includes updating Websites with new runtime versions, and about pushing website updates (content, design, etc.) from the Staging to the Production environment. The team consists of nine developers, a senior developer, a team coordinator, and a team lead.

[2]About iOS: http://www.apple.com/iphone/ios4/
[3]MS SQL Server (http://www.microsoft.com/sqlserver/en/us/default.aspx)

5.2 Process Evaluation

Section 4.5 describes a risk based process for choosing the right development process or the ideal level of planning. As outlined, application and environment factors (cf. Section 4.3) of a project have great influence on the decision process, which is especially true in a domain where risk is not the major factor. The next paragraphs discuss the factors that apply to the VEC work environment.

5.2.1 Risk

In most cases, the VEC does not develop business critical applications. However, during 2010 the average project tended to become both more complicated and time critical. The differences between the individual projects did not justify substantially differing processes yet but first steps are taken into this direction.

5.2.2 Application Categorization

Section 2.2 in this thesis shows that there is not one discipline of Web engineering but that it can be further divided into sub-disciplines depending on the application context. Table 5.1 lists how projects completed by the VEC are approximately distributed between the different categories. The list clearly shows that *simple*

Application Type	**%**
Informational	55
Interactive	25
Transactional	0
Workflow Oriented	5
Collaborative	5
Service Oriented	5
Social	5
Cloud Based	0

Table 5.1: VEC Project Distribution

projects still dominate daily business. EDetails fall into the *Informational* class. The rest, making up 20% of all projects reflect the growing demand for more sophisticated solutions.

5.2.3 Environment Factors

Domain Factors. The VEC is clearly doing Web engineering. With almost no exception, all software produced is either distributed by, or exposed to, the Web, or it is built with Web technologies. Therefore all of the peculiarities described in Sections 2.4, 2.5, and 2.6 apply.

Organization Factors. Communication between VEC members is often face to face, supported by the office facilities as the vast majority of staff are working in one open plan office. These kind of offices are also advocated by proponents of agile methodologies [17, 6, 15], as they clearly encourage personal conversation. Also, hierarchies are rather flat and do not hinder communication.

Project Factors. The VEC undertakes a lot of projects for many different customers. The customers themselves heavily influence how the project is conducted as he/she defines and changes requirements. Often, quick turnaround times are required leaving little time for architecture and documentation. Also development projects are usually very short. Most of them concern the adding, or changing of little pieces of functionality for one of the used *Content Management System* implementations. Long-term projects are often triggered internally and usually do not see more than 2 developers working on them simultaneously. Therefore even agile release schedules (or sprints) of one to four weeks are too long for the quick turnaround times.

People Factors. Overall the VEC team is quite young. Most staff range between 25 and 35 years of age. This influences flexibility in a positive way.

5.3 Resulting Processes & Activities

Considering the set preconditions, this chapter shows how these influence the project and development process implemented in the VEC. It also relates this to the methodology-related findings of previous sections. It describes both the overall project process (cf. Section 5.3.1) and the internal development processes (cf. Section 5.3.2) as both depend on each other and need to mutually complement each other.

Due to the given environmental factors (cf. Section 5.2.3), a few aspects must be considered:

- Quick requirements definition and agreement between stakeholders
- Independent work of developers
- Common quality and coding standards
- External quality checks
- Fast or continuous integration of new developments
- Keeping track of individual tasks

For this thesis, the last item of this list is central, as all of the practical work (cf. Section 5.4) is based on it.

5.3.1 Project Process

This section explains the processes and parties involved in running a project with the VEC. Figure 5.2 shows project phases and milestones, provides information about the roles and responsibilities of the project team members, and describes document deliverables from a customer perspective. The image does not show the delivering units like Media or Development, as these usually have no, or only supportive, direct customer contact. This might not at first glance look very agile at all, but the project manager represents the customer on site, being a constant peer for communication. This means that developers do not have to consider arranging timelines or discussing sign offs, as these activities are transparently handled by the project manager.

Only if the project contains major development efforts, real development practices come into place. These parts of the project are then overseen and planned by the development leadership team. Here, the development lead in some aspects relates to the Scrum Master. Besides only paving the way for the Team this role also contains activities like resource management, project management, and task assignment. This is especially true of very technology-driven projects, which to a large extent rely on development resources and know-how.

The project process itself relies on close collaboration between all departments. The three *delivery* teams — Development, Media, and ProS — need to work closely together. Still, the procedures differ from department to department. Often for simple sites, no or only minor backend development is needed. The remaining configuration tasks can be easily distributed amongst developers. On the other had, bigger projects often need more than one core developer, not only to deliver in time, but also to mitigate the contingency risk.

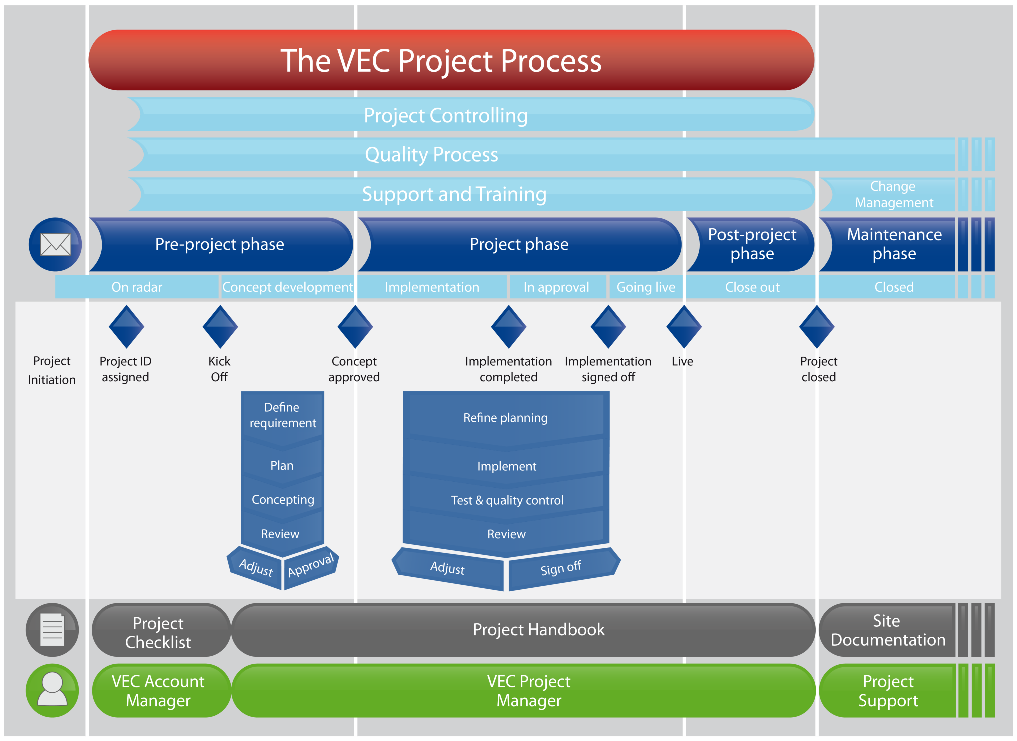


Figure 5.2: Summary of the VEC Project Process

5.3.2 Development Process

What the project process does for overall project handling and customer communication, the software development process does for internally managing the development efforts. Figure 5.3 shows the standard VEC development process defined for software creation. The smaller, orange boxes contain all the artifacts created throughout the process. The included decision nodes provide the possibility for review and iteration. In detail, they have the following functions:

1. **Development Feasible?** The requirements have been checked against the given timelines and available systems. If it is feasible, this triggers an initial effort estimation for creating an offer and to improve planning. If not, some preconditions will have to be changed. Usually the PM creates a TFS Task (cf. Section 5.4.4) in order to initiate development after this circle is finished.

2. **Specification OK?** A member of the development leadership team checks the elaborated specification and corrects it together with the PM if necessary. If it is OK, efforts are estimated by developer and lead.

3. **Effort OK?** After the efforts have been discussed between project management

and development, the former (either alone or in coordination with the customer) decides if the efforts meet given preconditions. If yes, implementation starts, if not, requirements might be changed.

4. **Prototype or Final Release?** During implementation, prototypes are constantly checked by project management (and the customer) and the development leadership team. Before forwarding the final solution to formal testing, a code review is done, acting as a quality gateway.
5. **Review Successful?** Only after a successful review and after all issues are fixed, are ProQ asked to perform tests.
6. **Test Successful?** If the ProQ tests have been performed without finding any bugs or other issues, the final version is checked into the source control trunk and the final deployment is done.

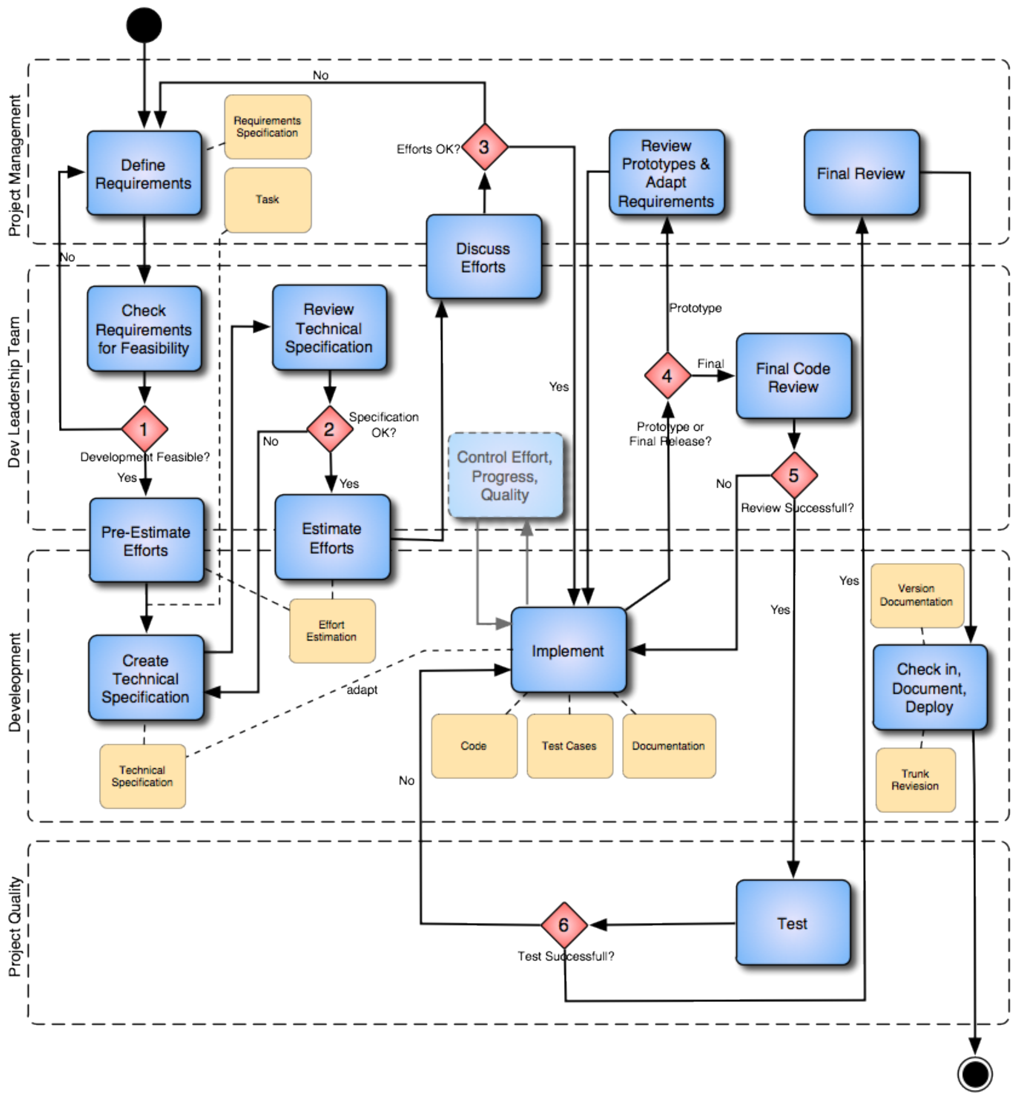


Figure 5.3: The VEC Development Process

As the activities around the *Implement* node show, the developer is in constant communication with the customer, represented by the project manager and the development lead. To achieve this, especially in bigger developments, prototypes are constantly integrated on development or staging servers for testing and review. For smaller work packages, some steps — such as creating a technical specification — can be skipped while still keeping crucial ones, like quality testing by ProQ or code review.

5.3.3 Comparing Practice and Theory

There is one fundamental thing that differentiates almost any VEC project from all the described theoretical approaches. Only very rarely is there the possibility to fully dedicate a person or even a team to just one specific project. This starkly contrasts with core claims of both XP and Scrum. Scrum especially emphasizes the need of having a dedicated, multi-disciplinary, and self-organized [89, p. 15] team for each project. Even if small teams can be established, proper planning (as required for a Sprint) is often deterred, as unexpected, important tasks with short timelines may present themselves. In fact, this requires more agility than even the named approaches provide. It might sound like a paradox, but in the end this increases planning overhead, as planners always have to take this uncertainty into account. This somewhat chaotic precondition hampers predictability and potentially risks agreed timelines. However, this just describes the status quo. As described in Section 5.2.2, the average complexity of the developed solutions increases. This automatically entails the demand for more resources which creates team-like structures if the project is big and important enough. Also applying the risk-driven decision methods as in Section 4.5 to this development indicates that this will lead to a more structured environment in the future.

Even if the VEC does not strictly follow any specific methodology, agile aspects found in Scrum (cf. Section 3.4) and XP (cf. Section 3.4) do play an important role as the following list shows.

Co-located Team. All VEC staff works in one big open plan office. This significantly eases face-to-face communication and speeds up the information flow.

Whole Team. Though this is not realized in the manner XP suggests (one project, one team, all disciplines, c.f. Paragraph 3.3.2), the VEC has all knowledge available to run projects.

Informative Workspace. The VEC uses flip-charts and whiteboards throughout the office to support discussions.

Short Iteration Cycles. When working on bigger projects, weekly iteration cycles are the goal. If this proves to be too short, usually they are extended to have bi-weekly releases. This falls into cycle durations both recommended in Scrum and XP.

Continuous Integration. Both for .NET and Java development, continuous builds are done, both on check-in and scheduled. New features are incrementally integrated on designated environments. Continuously updating staging and development systems for review aligns the deliverables with what the customer expects. Therefore, there are no big-bang deployments and associated surprises at the end of a project. A more heavyweight approach would certainly not comply with the very flexible nature of the business.

Customer on Board. The customer is represented by the project manager. This person sits in the same office as the development team and is always available.

Shared Code. Nobody owns code personally. Often extensions or fixes of existing software are deliberately assigned to developers who have not been working on the code in the past. Doing this, not only one person knows the code, thereby mitigating the contingency risk and potentially increasing quality.

Limited Paperwork. The development team tries not to produce unnecessary paper. The focus clearly lies on creating software instead. Still, documentation is required and must be extended during development. Often a concept paper becomes the technical documentation as it grows.

Stand-Ups. Bigger projects are usually accompanied by daily stand-up meetings.

Lessons-learned. After every project, all project members participate in a *lessons-learned* meeting, following the model of the Scrum Sprint Retrospective.

From a process perspective, the RUP (cf. Section 3.2) does not apply to the VEC as it is more targeted on classical, large scale software development. Still, UML is used to design and document bigger architectures as it is a de-facto industry standard and therefore is understood throughout the development team. Also the development process as outlined in Figure 5.3 follows the RUP terminology of distinguishing between Roles (cf. Paragraph 3.2.2), Activities (cf. Paragraph 3.2.2), and Artifacts (cf. Paragraph 3.2.2). However, the artifacts described are not the main focus of VEC development processes.

To summarize, the VEC development process is an organically grown procedure, leaving much space for ad-hoc activities. It accepts that it is not possible to enforce one single methodology to cover both the occasional very ad-hoc developments often not lasting longer than two days, whilst at the same time applies to projects with development efforts of a few months. The process also only applies to the development department. Media work is not included in these processes, but form part of the standard project procedures as described in Section 5.3.1. This process can only be sustained by keeping communication lines very open, whilst not overstraining the individual through the high number of different projects. Canalizing and documenting this information flow is therefore a key activity and can only be realized with appropriate tool support. This is what the following pages focus on.

5.4 Implementation

This section represents the practical work done in terms of implementing specialized Work Item Types 5.4.3 and related components in Microsoft[4] Team Foundation Server (TFS) to fit the VEC's requirements. As TFS is the main internal form of creating and assigning work requests, it is a key tool for managing everyday work. The actual implementation was preceded by a phase of collecting requirements from all department leads. After a first draft, further requirements had been implemented and further adaptations have since been made on an irregular/on-request basis.

5.4.1 Problem Description

The very agile manner of how projects are done in the VEC relies on direct communication lines between all stakeholders within the organization. Without a regulatory factor, this can lead to a rather chaotic situation in which everybody directly assigns tasks to a specific person without keeping the *big picture* in mind. This basic problem had been solved with the introduction of team leads. They act as central hubs that receive and dispatch tasks. This is only possible by canalizing existing communication streams through one central tool (c.f. Figure 5.4). With an increasing number of requests, not relying on a centralized solution can easily overstrain the cognitive capacity of individuals, increasing the risk of forgetting or missing something.

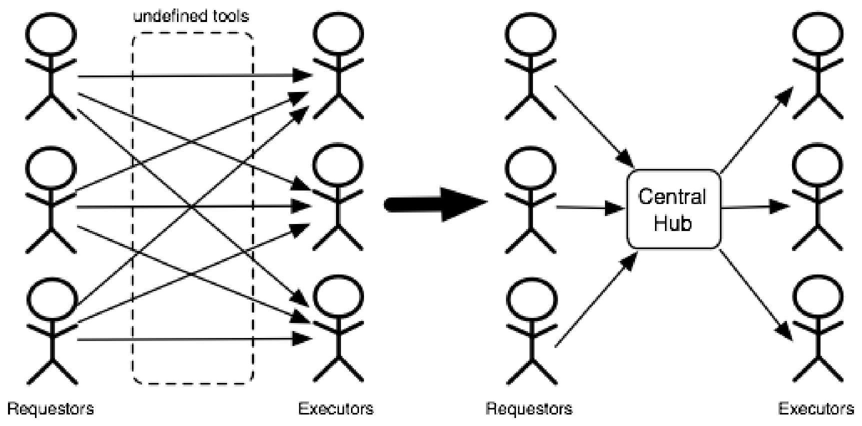


Figure 5.4: Overcoming the Communication Chaos

[4]http://www.microsoft.com

5.4.2 Problem Solution

In order to solve the problem described, a tool was needed that could assist in task management and would integrate into the existing infrastructure. As it was already planned to upgrade to Visual Studio 2008, the TFS was an obvious choice.

Microsoft Team Foundation Server. The Microsoft Team Foundation Server's (TFS) primary purpose is to enable collaboration within a team to support building software products, and to complete projects [111]. The VEC has used the TFS since 2009 throughout the whole application lifecycle as it provides integration into Microsoft Visual Studio, source control, automated build management, team member alerts, role and task management, iterations, and reporting. Some of these features can be customized, for example by using different Process Guidance templates such as the one described in Paragraph 5.4.2. All information in this thesis is related to Microsoft Visual Studio Team System 2008. Figure 5.5 gives an overview on how TFS integrates into existing software development processes.

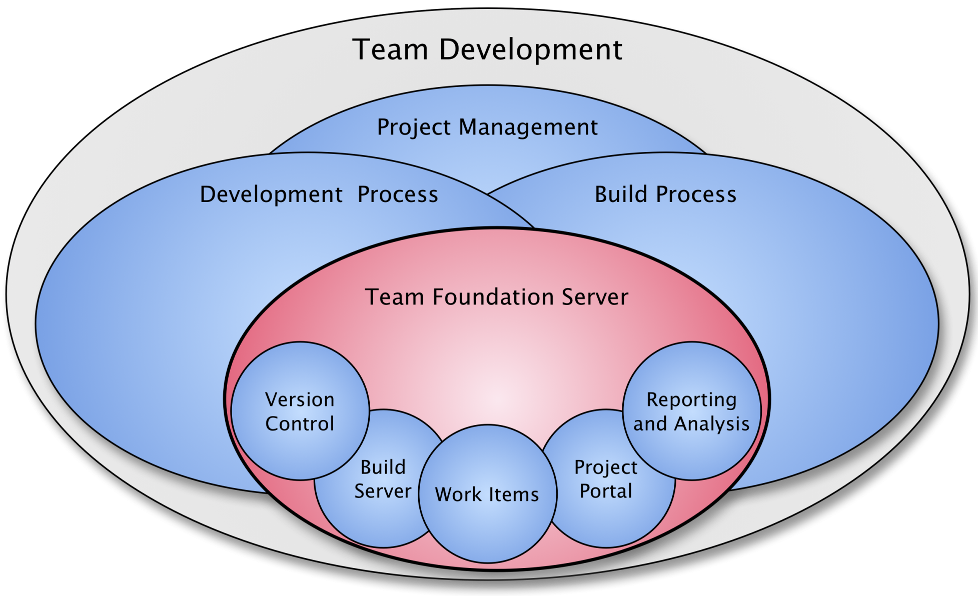


Figure 5.5: TFS Integration into Software Development Processes [111]

MSF for Agile Template. The Microsoft Solution Framework (MSF) in the 2008 version is a software development process framework. It was established in 1994 as a collection of best practices from Microsoft's product development efforts and consulting engagements. Today, MSF is developed by a dedicated development team within Microsoft. The MSF does not only concentrate on software application development, but also covers operations management perspectives. Figure 5.6 presents the components of MSF version 4. The red boxes are the ones on which this thesis focuses. *Infrastructure Development* is only connected with a dashed line as Microsoft does not offer a product supporting it yet.

The MSF can be utilized as a *Process Template* for TFS, providing a collection of files that together define various process elements of a team project in Team Foundation Server [112]. All further customization (cf. Section 5.4) is based on this template. Version 5 has been released as a part of *Visual Studio Team System* in 2010 and is based on Scrum. However, all practical work has been completed using the 2008 version.

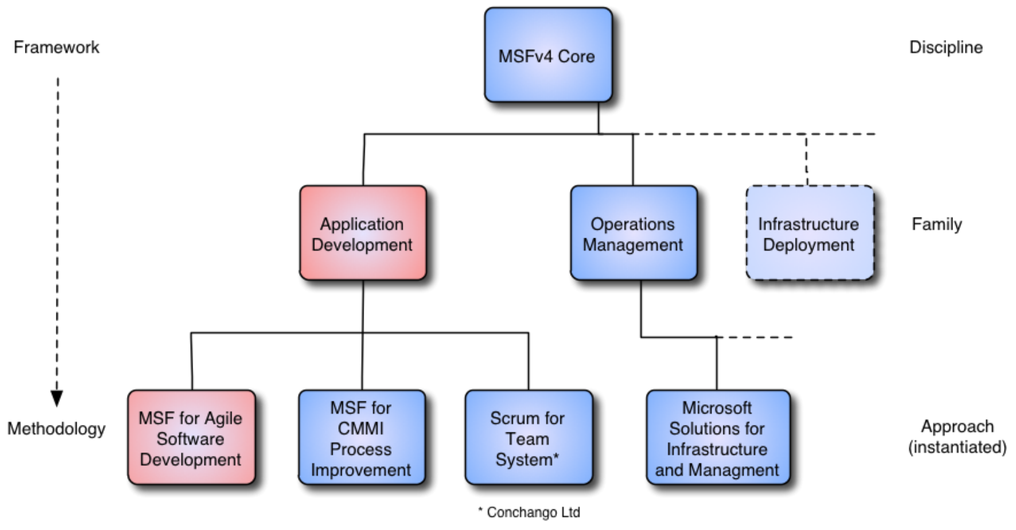


Figure 5.6: MSF Components Overview [113]

5.4.3 Adaptations

During the first months of running TFS it became clear that the standard MSF for Agile task management capabilities were not covering all the team's requirements. As the introduction of another tool was not in scope, it was decided to configure TFS appropriately. The subsequently implemented artifacts require both coding and configuration. This part of the thesis describes how this was performed and what it required. All required knowledge was acquired via self study. Most resources needed are freely available on blogs, forums, the Microsoft Developer Network[5], etc. on the Internet.

The Work Item implementation is done in Microsoft Visual Studio Team System 2008 Development Edition (VS) using the TFS Power Tools[6] add on. Writing the Deadline Reminder (cf. Section 5.4.9) in C# additionally requires the Team Foundation Server SDK for Visual Studio 2008. The Power Tools in particular are more or less inevitable for administering the TFS.

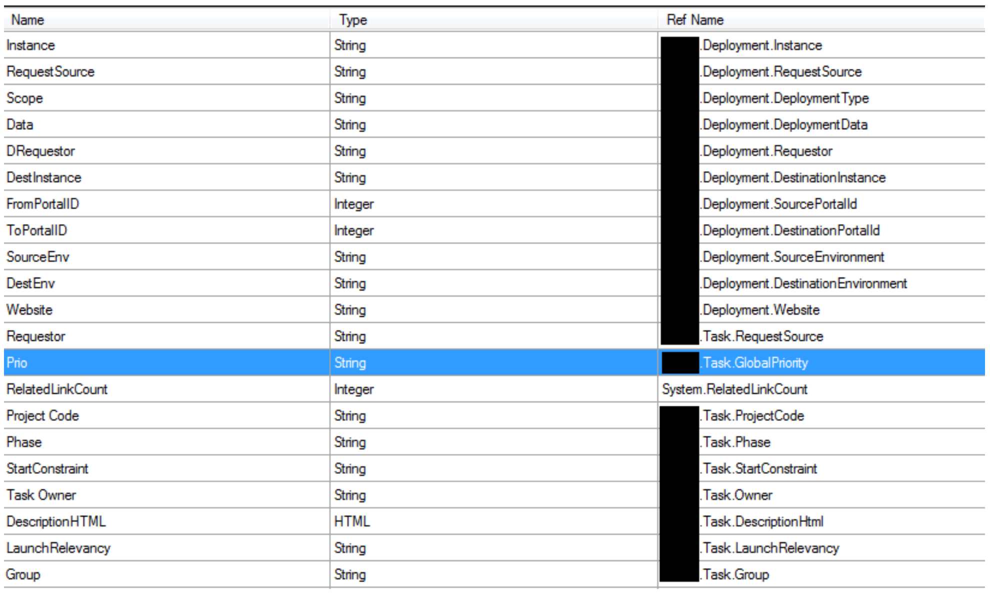
Work Item Type (WIT) is the generic TFS notation for all different kinds of requests/tasks that can be created. One single instance of a created WIT is then called *Work Item* (WI). MSF for Agile contains some predefined WITs (Bug, Risk, Scenario, Task, Quality of Service Requirement) but these do not fit the VEC's needs. Reviewing the requirements made clear that two main Work Items are necessary for daily work: *Task* and *Deployment Request*. As the TFS is the central task management solution for the VEC, *Personal Todo* has been added to provide an interface that is capable of managing both tasks that are assigned to somebody else as well as personal ones. All WITs have an underlying workflow defined, which means that their state can change. These states are critical as they are the main source of filtering.

WITs are represented by XML files, also called *Work Item Type Definition*. Examples can be found in Appendix A. These files can be imported into and exported from TFS and can be modified manually.

Fields. Each WIT consists of a number of fields that in turn can be used to build the user interface. Each process template comes with a set of standard fields. However, these are not sufficient for more extensive adaptation. Screenshot 5.7 shows a number of custom additions needed for the desired customizations.

[5]MSDN: http://msdn.microsoft.com/en-us/default.aspx

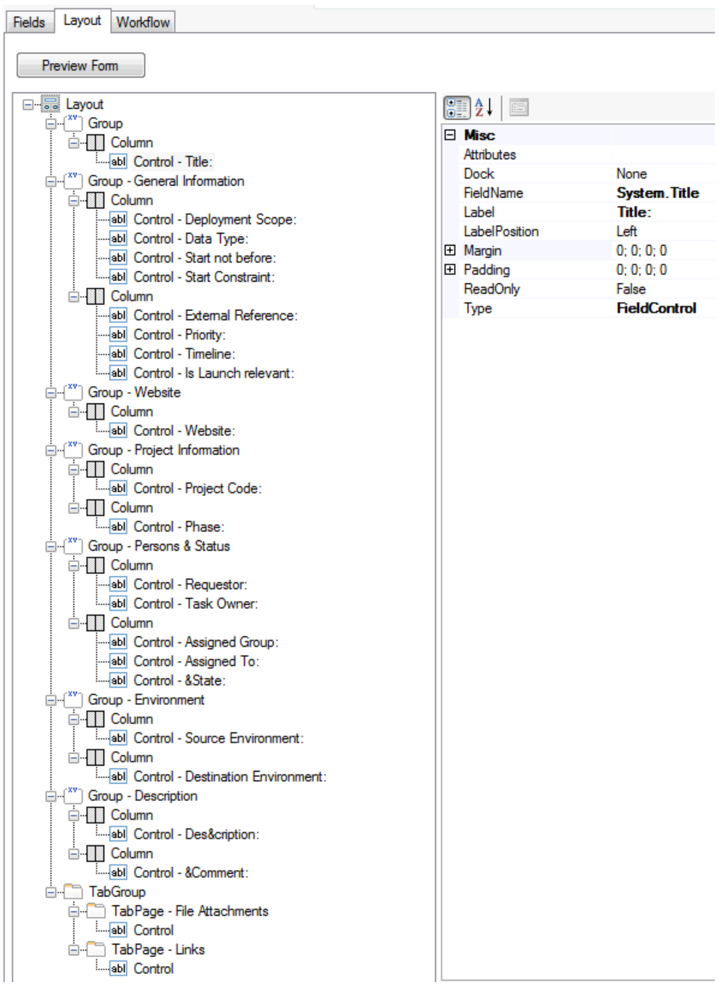
[6]Download: http://www.microsoft.com/downloads/en/details.aspx?FamilyID=fbd14eea-781f-45a1-8c46-9f6ba2f68bf0&DisplayLang=en



Name	Type	Ref Name
Instance	String	[redacted].Deployment.Instance
RequestSource	String	[redacted].Deployment.RequestSource
Scope	String	[redacted].Deployment.DeploymentType
Data	String	[redacted].Deployment.DeploymentData
DRequestor	String	[redacted].Deployment.Requestor
DestInstance	String	[redacted].Deployment.DestinationInstance
FromPortalID	Integer	[redacted].Deployment.SourcePortalId
ToPortalID	Integer	[redacted].Deployment.DestinationPortalId
SourceEnv	String	[redacted].Deployment.SourceEnvironment
DestEnv	String	[redacted].Deployment.DestinationEnvironment
Website	String	[redacted].Deployment.Website
Requestor	String	[redacted].Task.RequestSource
Prio	String	[redacted].Task.GlobalPriority
RelatedLinkCount	Integer	System.RelatedLinkCount
Project Code	String	[redacted].Task.ProjectCode
Phase	String	[redacted].Task.Phase
StartConstraint	String	[redacted].Task.StartConstraint
Task Owner	String	[redacted].Task.Owner
DescriptionHTML	HTML	[redacted].Task.DescriptionHtml
LaunchRelevancy	String	[redacted].Task.LaunchRelevancy
Group	String	[redacted].Task.Group

Figure 5.7: TFS - Custom Fields

Layout The user interface is built in the Layout section (cf. Figure 5.8) of the WIT creation interface. All elements defined can be used as controls here. The design itself follows a strict table layout approach with rows and columns.



```
Layout
    Group
        Column
            Control - Title:
    Group - General Information
        Column
            Control - Deployment Scope:
            Control - Data Type:
            Control - Start not before:
            Control - Start Constraint:
        Column
            Control - External Reference:
            Control - Priority:
            Control - Timeline:
            Control - Is Launch relevant:
    Group - Website
        Column
            Control - Website:
    Group - Project Information
        Column
            Control - Project Code:
        Column
            Control - Phase:
    Group - Persons & Status
        Column
            Control - Requestor:
            Control - Task Owner:
        Column
            Control - Assigned Group:
            Control - Assigned To:
            Control - &State:
    Group - Environment
        Column
            Control - Source Environment:
        Column
            Control - Destination Environment:
    Group - Description
        Column
            Control - Des&cription:
        Column
            Control - &Comment:
    TabGroup
        TabPage - File Attachments
            Control
        TabPage - Links
            Control
```

Misc	
Attributes	
Dock	None
FieldName	**System.Title**
Label	**Title:**
LabelPosition	Left
Margin	0; 0; 0; 0
Padding	0; 0; 0; 0
ReadOnly	False
Type	**FieldControl**

Figure 5.8: TFS - WIT Layout

Workflow. Every WIT has a workflow assigned. A workflow defines the *States* that can be assigned to a WIT. Examples would be open, in progress, or closed. Workflows are modeled in the respective Workflow section of the WIT creation interface. A small extract is shown in Figure 5.9. The connection between two States is called *Transition* with each of them forming a 1:1 connection. Each Transition can have n numbers of *Reasons* assigned that may be chosen when changing the State of a Work Item.

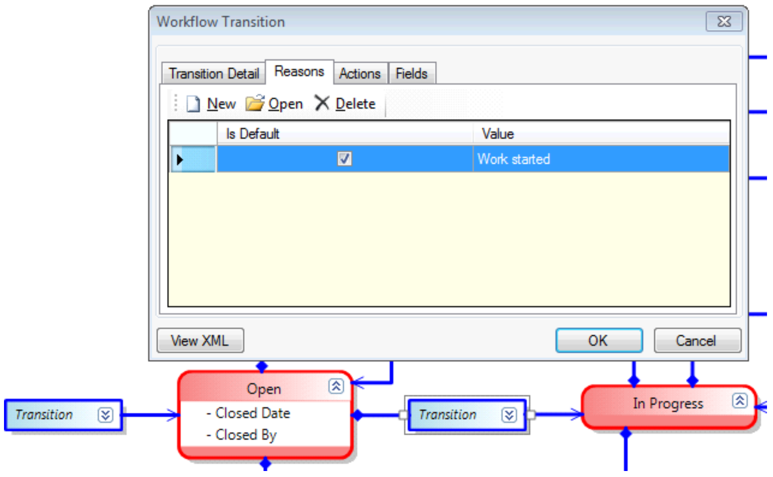


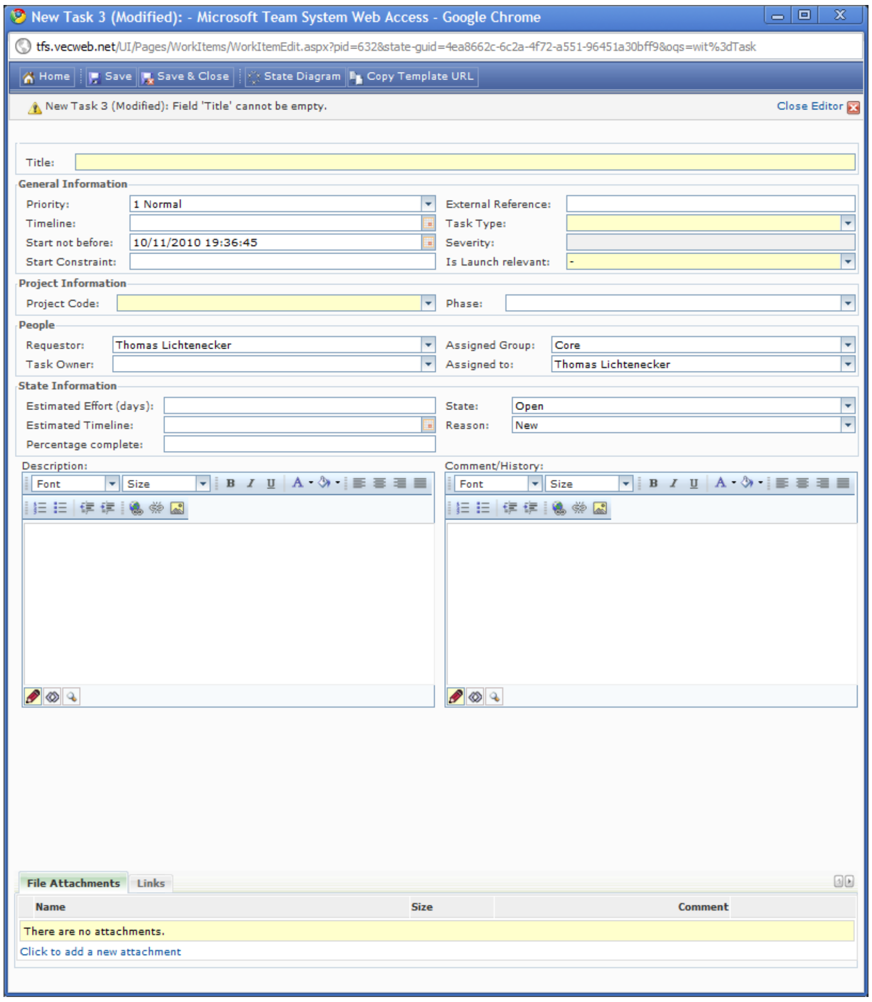
Figure 5.9: TFS - Transition and Reason

5.4.4 Work Item Type *Task*

The Task is the central WIT in the solution. It can cover almost everything that somebody might request from another team member. Currently the following Task Types are featured:

- Bug: something is broken somewhere and should be fixed
- Feature Request: a new feature is required
- Support Request: request that something has to be configured, evaluated, investigated, etc.
- Test: something has to be tested

The user interface is shown in Figure 5.10. It divides the content into six groups to add some structure. The yellow boxes in the screenshot represent required fields. The related fields are listed in Table 5.2. The XML representation is part of Appendix A.1.


New Task 3 (Modified): - Microsoft Team System Web Access - Google Chrome

tfs.vecweb.net/UI/Pages/WorkItems/WorkItemEdit.aspx?pid=632&state-guid=4ea8662c-6c2a-4f72-a551-96451a30bff9&qs=wit%3dTask

New Task 3 (Modified): Field 'Title' cannot be empty.

Title:

General Information

Priority:	1 Normal	External Reference:	
Timeline:		Task Type:	
Start not before:	10/11/2010 19:36:45	Severity:	
Start Constraint:		Is Launch relevant:	-

Project Information

Project Code:		Phase:	

People

Requestor:	Thomas Lichtenecker	Assigned Group:	Core
Task Owner:		Assigned to:	Thomas Lichtenecker

State Information

Estimated Effort (days):		State:	Open
Estimated Timeline:		Reason:	New
Percentage complete:			

Description:

Comment/History:

Name	Size	Comment
There are no attachments.		

Click to add a new attachment

Figure 5.10: TFS Task - User Interface

Name	**Description**	**Allowed Content**
Title	Title of Task	String
Priority	Priority of Task	0 Low 1 Normal 2 Important 3 Emergency
Timeline	Desired finish date	Datetime
Start not before	Constraint date	Datetime
Start Constraint	External start constraint	String
External Reference	e.g. ticket number	String
Task Type	Type of Task	Bug, Feature Request, Test, Support Request
Severity	In case of Bug	1 - Cosmetic 2 - Functionality affected 3 - Showstopper
Is Launch relevant	Launch relevant?	yes, no
Project Code	VEC project code	Imported list
Phase	Project Phase	Project, Maintenance
Requestor	Task requestor	Active Directory entities
Task Owner	Task owner	Active Directory entities
Assigned Group	Internal groups	Core, SMA
Assigned To	Assigned person or team	Active Directory entities
Estimated Effort	Planned effort	String
Estimated Timeline	Planned timeline	Datetime
Percentage complete	Rate of completeness	String
State	State of task	Section 5.11
Reason	Reason for state change	Dropdown
Description	Task description	HTML
Comment/History	Change description	HTML
File Attachments	Attached files	File
Links	Links to other WIs	Link

Table 5.2: TFS Task - User Interface Fields

The Task workflow is kept as generic as possible to reflect the various task types that can be associated with it. As depicted in Figure 5.11, it makes use of descriptive information associated with each state transition. Those displayed in the diagram are the default values given when no other choice is made.

The basic workflow is Open →In Progress →Executed →Closed. In Progress may be disregarded for short timed tasks. All closed tasks can be reopened again. From every state it is possible to set a Task to Cancelled. Tasks can also be put On Hold if their execution is paused for any reason. The See History transition comment refers to the History functionality of all WITs, which stores every change. Here, comments can be added.

The different colors in the diagram refer to the people/departments who are most closely associated to a certain step where red represents ProQ, green is related to the Task creator, and blue represents the person responsible for task execution. Only ProQ users can set Tasks from Executed to Closed to ensure that no Task can be finished without going through proper quality control.

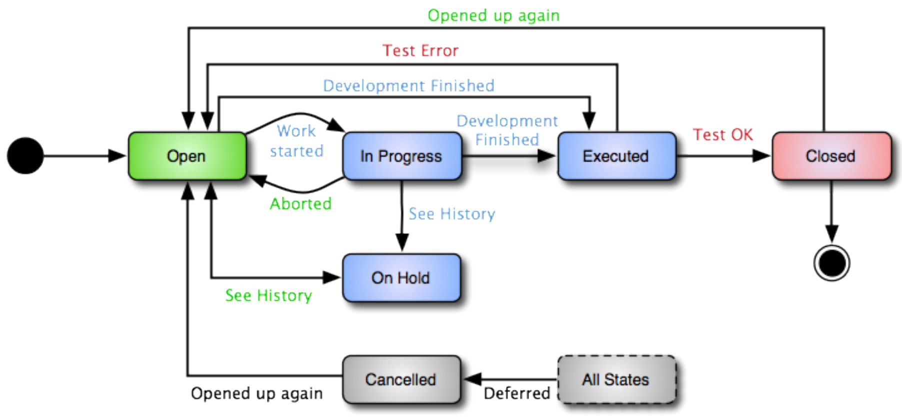


Figure 5.11: Task State Diagram

5.4.5 Work Item Type *Deployment Request*

A Deployment Request (DR) is a WIT specific to jobs that require the synchronization of entities between the development, staging, and production environments. The DR workflow is based on a queue concept that automatically splits created tasks into distinct queues for DEV and for ProS. Therefore, no classical task assignment is needed. In contrast, the person opening a DR to work on it self-assigns it to take it out of the queue. The workflow depicted in Figure 5.12 shows three roles

— again represented by corresponding colors as in the state diagram. The Requestor represents the person requesting a deployment to be executed, the Executor refers to the person/team executing the deployment — DEV or ProS and ProQ stands for the quality assurance team. On first glance the DR workflow might look more complicated than that associated with the Task. However, the DR has no reasons defined for state transitions as all information needed is part of the Work Item's state name.

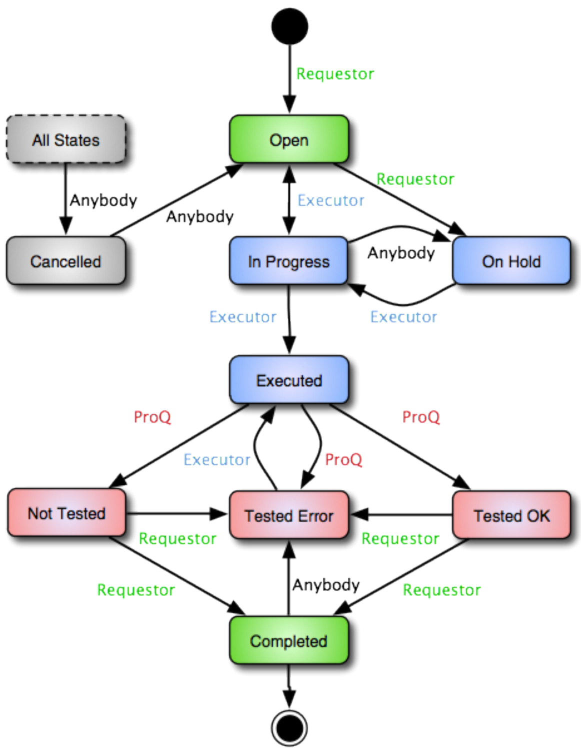


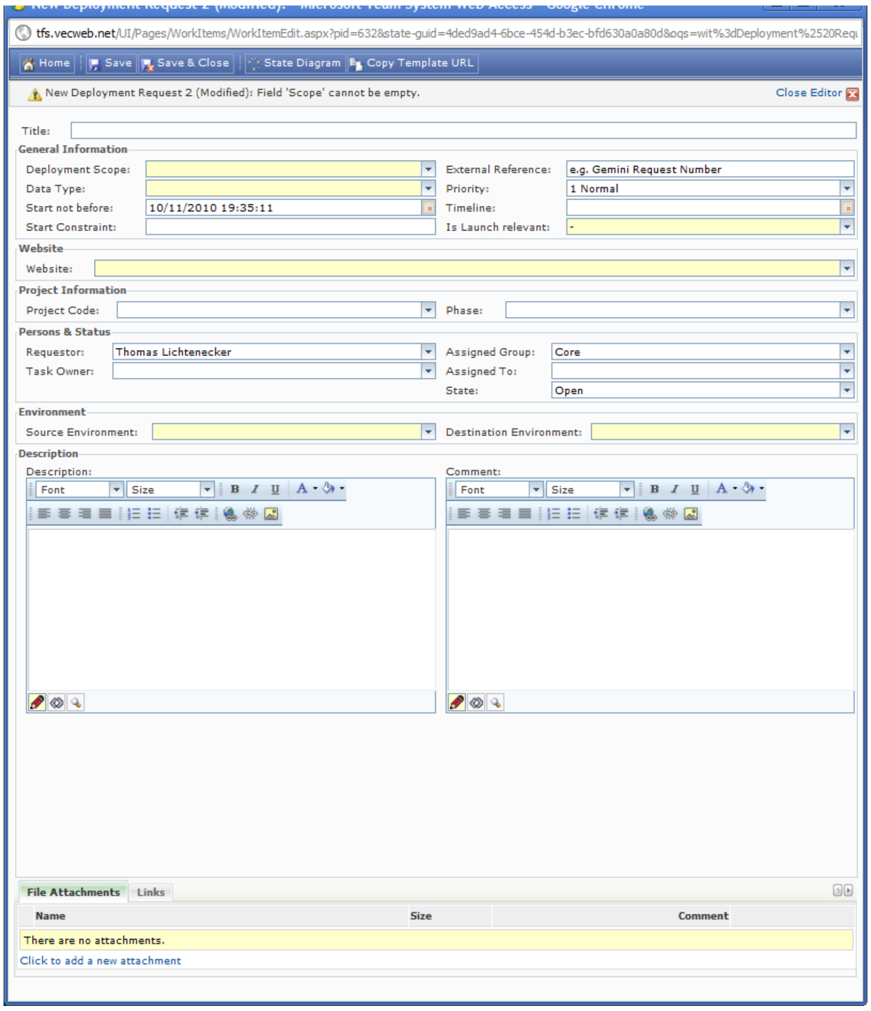
Figure 5.12: Deployment Request State Diagram

The states that can be chosen are:

- Open: The task is newly created. Both ProS and DEV only see their subset of deployment tasks.
- Executed: ProS or DEV have executed the deployment, so it is finished from their perspective.
- On Hold: Work Item execution has been paused for some reason.
- Not Tested: ProQ decides that the deployment does not have to be tested. Work Item completion still has to be confirmed by the Requestor.

- Tested Error: ProQ has tested the deployment and an error has been found. Work Item completion still has to be confirmed by the Requestor. Also Work Items which are Tested OK or Not Tested can be set to Tested Error if a defect was found.
- Tested OK: ProQ has tested the deployment and everything is OK. Work Item completion still has to be confirmed by the Requestor.
- Completed: If the Requestor has reviewed the Work Item and everything is OK, it can be set to Completed. Alternatively, if the review was not successful, the Work Item can be set to Tested Error.
- Cancelled: Task is deferred for whatever reason.

The user interface is shown in Figure 5.10. It divides the content into eight groups to add some structure. As for the Task, yellow boxes represent required fields. The related fields are listed in Table 5.2. The XML representation is part of Appendix A.2.


tfs.vecweb.net/UI/Pages/WorkItems/WorkItemEdit.aspx?pid=632&state-guid=4ded9ad4-6bce-454d-b3ec-bfd630a0a80d&oqs=wit%3dDeployment%2520Requ

New Deployment Request 2 (Modified): Field 'Scope' cannot be empty.

Title:

General Information

Deployment Scope:		External Reference:	e.g. Gemini Request Number
Data Type:		Priority:	1 Normal
Start not before:	10/11/2010 19:35:11	Timeline:	
Start Constraint:		Is Launch relevant:	-

Website

Website:

Project Information

Project Code:		Phase:	

Persons & Status

Requestor:	Thomas Lichtenecker	Assigned Group:	Core
Task Owner:		Assigned To:	
		State:	Open

Environment

Source Environment:		Destination Environment:	

Description

Description:

Comment:

Name	Size	Comment
There are no attachments.		

Click to add a new attachment

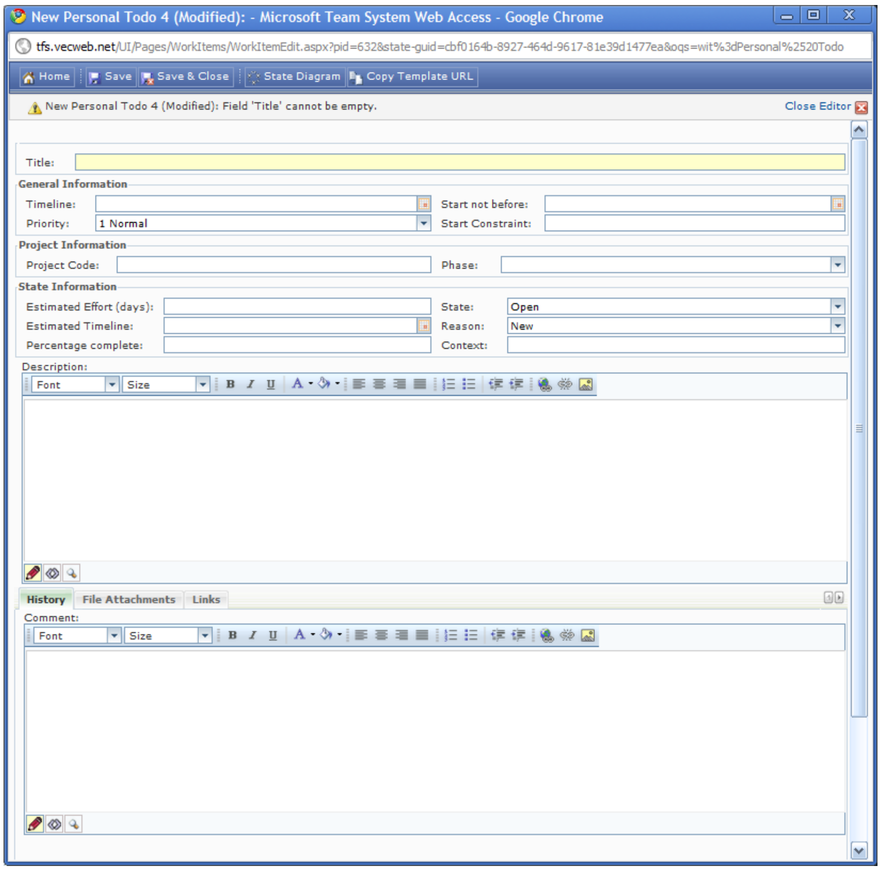
Figure 5.13: TFS Deployment Request - User Interface

Name	**Description**	**Allowed Content**
Title	Title of DR	String
Deployment Scope	The scope	1 Content 2 Design/Template 3 Binary 4 Whole Portal 5 Whole Instance 6 Launch 7 New Website
Data Type	The *what*	1 Files, 2 DB, 3 Files & DB
Start not before	Constraint date	Datetime
Start Constraint	External constraint	String
External Reference	e.g. ticket number	String
Timeline	Desired finish date	Datetime
Priority	Priority of DR	0 Low 1 Normal 2 Important 3 Emergency
Is Launch relevant	Launch relevant?	yes, no
Website	Affected Web	Imported list
Project Code	VEC project code	Imported list
Phase	Project phase	Project, Maintenance
Requestor	DR requestor	Active Directory entities
Task Owner	DR owner	Active Directory entities
Assigned Group	Internal groups	Core, SMA
Assigned To	Assigned person or department	Active Directory entities
State	State of DR	Section 5.12
Source Environment	The source	1 DEV, 2 STA, 3 PRD
Destination Environment	The target	1 DEV, 2 STA, 3 PRD
Description	DR description	HTML
Comment/History	Change description	HTML
File Attachments	Attached files	File
Links	Links to other WIs	Link

Table 5.3: TFS Deployment Request - User Interface Fields

5.4.6 Work Item Type *Personal Todo*

The WIT Personal Todo (PTD) is more or less a simplified version of a Task, providing an easy-to-use interface (cf. Figure 5.14). It is automatically assigned to the person that creates it. The workflow is Open →In Progress →Closed. The PTD uses fields already defined for Task and DR. Its XML representation is part of Appendix A.3.



New Personal Todo 4 (Modified): - Microsoft Team System Web Access - Google Chrome

tfs.vecweb.net/UI/Pages/WorkItems/WorkItemEdit.aspx?pid=632&state-guid=cbf0164b-8927-464d-9617-81e39d1477ea&oqs=wit%3dPersonal%2520Todo

New Personal Todo 4 (Modified): Field 'Title' cannot be empty.

Title:

General Information

Timeline:		Start not before:	
Priority:	1 Normal	Start Constraint:	

Project Information

Project Code:		Phase:	

State Information

Estimated Effort (days):		State:	Open
Estimated Timeline:		Reason:	New
Percentage complete:		Context:	

Description:

Comment:

Figure 5.14: TFS Personal Todo - User Interface

5.4.7 TFS Queries

In order to extract information from TFS, the system provides the possibility to create queries. Queries allow filtering for all used WIT fields by using operators depending on their data type. This operators include: >, <, <>, ≥, ≤, In (e.g. certain group), Was Ever (e.g. was ever assigned to somebody), Contains (e.g. part of project name), and Does Not Contain. The underlying language is based on a SQL-like syntax and is called *Work Item Query Language* (WIQL). WIQL allows the combination of filters with AND/OR operators. This allows creating fairly complex queries. A list of all 27 custom VEC queries created is available in Section B.1 as well as an example (cf. Appendix B.1) that returns all Tasks that have been assigned to the Development department but are not assigned to a developer.

5.4.8 User Alerts

TFS is capable of automatically sending e-mails when fields change their content. Unfortunately, this feature is limited in a way that only permits the application of some important query parameters (such as *changes to*) to core field types. Also, alerts are always stored in the context of the user that created them, which makes administration unnecessarily burdensome as one must create a central admin user if configuration should be shared between users. Nevertheless, the application currently stores 102 different alerts for VEC members. Their main function is to inform users of changes to Work Items that they have either created or that have been assigned to them. Also, department alerts exist that allow team leads to keep track of the progress of tasks assigned to one of their team members. The reminders themselves are stored as XPath queries in the TFS database such as these shown in two examples in the Appendices B.2 and B.3.

5.4.9 Deadline Reminder

The deadline reminder is basically a workaround to cover functionality that is not available through default TFS mechanisms. It is a small C# program (source code: Appendix B.3) that queries TFS for Work Items whose finish date is today+1 and adds this information into the History field of the respective Work Items. As this changes the content of a Work Item field, this triggers the standard user alert functionality as in Section 5.4.8. The tool runs on the same server as TFS and is triggered nightly via a Windows Service.

5.5 Critical Discussion

The implemented solution has definitely changed the way in which the VEC works. TFS is now the main means of assigning tasks and the only accepted way of sending requests between teams, or even within them. The next paragraphs discuss the advantages and disadvantages of the chosen system.

5.5.1 Strengths of the Solution

The TFS is well accepted and from an end-user perspective is relatively easy to work with. Thanks to the made adaptations, the system covers all the standard use cases in the VEC process workflow. The query mechanism provides flexible access to all information held by Work Items. This information can be accessed via the standard Web interface, which presents all options in a well structured manner. All queries can be stored on a personal or team level and the output produced can be filtered and ordered by columns.

All information held by TFS is stored centrally in a few MSSQL databases. On top of this, the system creates a cube that allows extracting information in tools like Microsoft Excel. Even without this, the simple fact of running all major communication via one channel makes it easy to retrieve historical data. Before the tool was introduced in its current form, there was no real way of measuring the workload. Now it is at least possible to count the number of accomplished Tasks, Bugs, or Deployment Requests. This makes work more transparent and gives developers a better understanding of what they have achieved.

Another benefit is that all staff have a personalized overview of what is of individual interest: an executor only sees a list of tasks assigned to him/her, ordered by priority and timeline. The same interface also gives valuable information like the affected project (for correctly booking the hours spent on a task) and the requestor. The latter sees a similar screen with all requests he/she sent, in which state they are, and who is working on them.

The automated notification system makes it harder to neglect timelines and outstanding tasks. For team coordinators this reminder is a good indicator of resource shortages and facilitates acting in a timely manner. The C# code providing this functionality makes use of the rich TFS API, which nicely integrates into Visual Studio. For a developer who knows the system, it allows quite extensive modifications.

Sharepoint services integration makes it possible to totally integrate TFS into an existing Sharepoint architecture. That includes Microsoft Office connectivity.

5.5.2 Weaknesses of the Solution

With the TFS, Microsoft tries to combine various different components supporting the software developing cycle into one product. Though this provides a good opportunity to integrate otherwise only loosely coupled systems, version 2008 seems like it tries to do a lot and in the end does nothing really right. This impression is created when realizing that it requires a third party plugin (TFS Power Tools) to actually be able to do any customization without touching XML files and the command line. Even with the help of this tool, the interface feels unfinished and unnecessarily cumbersome to work with. Also, the included source control component is quite awkward to use and falls short when compared to free solutions.

The end-user interface for creating/updating Work Items only allows table-like layouts and does not provide the full wealth of HTML controls (e.g., checkboxes are not available). It is also not possible to properly bulk-edit Work Items or to edit them directly where they are listed. Linking Work Items to each other is also a problem in TFS 2008, as the interface for this purpose is very unfriendly to the user and because it is not possible to create hierarchies (Work Item A is a child of Work Item B, possible with TFS 2010). This is, in fact, a major drawback as it renders splitting bigger tasks into small, functional entities impossible.

Work Item Types and Work Items always exist in the context of a project. This behavior makes sense if a team only works on a few projects over the year. However, for creating tasks, this always requires switching between contexts, which is not very practicable. Therefore, the current solution uses only one generic project named *VEC Tasks*. Unfortunately, the actual codebase that forms the shared base for all Web projects is not part of this project but spread over several others. This makes it impossible to connect Work Items with commits to source control repository. For some reason, this separation is only partly true for the fields used to build the Work Item Types, as their names are unique globally even though a field with the same namespace can be created again for each Work Item Type.

Though the TFS provides an almost complete API, all the restrictions for closed source software apply. For instance, using all TFS 2010 features in Visual Studio requires updating the latter to version 2010 as well. Of course, the price tag that comes with this upgrade can not be overlooked. Also, TFS integration is limited to Visual Studio and support for other systems like Eclipse[7] is only available through another commercial product[8].

[7] `http://www.eclipse.org`
[8] `http://www.teamprise.com/`

5.5.3 Result

In a pure .NET environment, task management via TFS makes sense. However, the tool's inflexibility, incompleteness, and closed character make it hard to justify a recommendation. At least from the experience with version 2008, it makes more sense to sacrifice a tiny bit of integration by using tools that are really focused on their use case, and often free of charge. Open source examples include Redmine[9] (issue management, wiki, file repository, project management, forum, etc.), Jenkins[10] (continuous integration), Bitten[11] (build server, metrics), and SVN[12] (source code version control). There are also commercial competitors such as Atlassian[13] (Jira and other tools) and JetBrains[14] (YouTrack, TeamCity).

[9] `http://www.redmine.org/`
[10] `http://jenkins-ci.org/`
[11] `http://bitten.edgewall.org/`
[12] `http://subversion.apache.org/`
[13] `http://www.atlassian.com/`
[14] `http://www.jetbrains.com/index.html`

Chapter 6

Conclusion

This thesis discussed a topic named Web engineering — a field which experiences rapid change and at the same time receives little academic attention. With Web applications becoming more and more sophisticated and complex, a thorough review based on existing literature seemed necessary. This includes literature on software development in order to verify its applicability to the Web environment. The thesis therefore covers three major theoretical areas.

First, it gave an overview of Web engineering and presented a method to categorize it. This approach separates the domain into subtypes based on novelty and complexity. This leads to a total of eight different subcategories. The other important categorization step was to work out the peculiarities of Web engineering in comparison with a more generic understanding of software development. In the process of writing the thesis, this has been proven to be indispensable for judging the applicability of standard software development processes for Web engineering. These characteristics have been elaborated by looking at the discipline from the three distinct perspectives of (end-)users, developers, and managers. Especially for customer facing Web applications with medium complexity this showed that the contrast with classical software engineering are reasoned by the important role of media asset and content creation, in usually tight timelines, fast update cycles, and limited customer know-how.

The second theoretical part introduced the concept of software engineering methodologies and the distinction between lightweight and heavyweight types. Subsequently, three modern and popular proponents have been presented, with the Rational Unified Process (RUP) representing the more traditional/heavyweight faction and Scrum and Extreme Programming (XP) covering the agile/lightweight world. These types were described in a manner that is understandable for non-professionals in this area, whilst providing enough background to make the findings in the consecutive chapters reproducible.

Classifying the three discussed software development paradigms was then subject of the last theoretical part of the thesis. This has been computed based on the processes' characteristics and on the coverage they provide for the software development cycle. This showed that the RUP is clearly the most complete of all processes with XP and Scrum only partly covering the application lifecycle. On the other hand, the RUP entails significant complexity and process-overhead. It also requires a deep level of understanding to adapt existing processes in order to make them RUP compliant. This information made it possible to further discuss the methods' applicability for Web engineering. The main finding was that in virtue of the environment's ever changing and flexible nature, agile approaches seem to be more suitable for the majority of use cases. This is also reflected by the strong influence of domain factors on the chosen methodology. Here, the literature sees the Internet application domain as being particularly appropriate for agile approaches. Other influential factors include the nature of the project itself and the organization and people executing it. The focus of heavyweight processes on producing non-source-code artifacts contrasts starkly with the typical Web specifics. Only under special circumstances would it be necessary to opt for more structured approaches. This partly relates to the incompleteness of the agile methodologies that were analyzed. It has been shown that compared to the RUP, XP and Scrum only concentrate on specific parts of the development cycle. However, combining them fills most of the gaps and poses a reasonable option.

For cases of uncertainty, the thesis then presented a solution which bases the choice between different software development methods mainly on the overall project risk. This attempt was then combined with the categorization of Web engineering elaborated in the first part of the thesis, producing a rough guidance for choosing the correct development paradigm.

The practical part of the thesis described what had been accomplished in a real-world scenario in order to cope with the specific challenges of Web engineering. It gave an overview on the work environment of a team (the Vienna eMarketing Center - VEC), realizing Web projects for one of the biggest pharmaceutical companies in the world. Here, it focused on the processes found to be useful which in turn were partly based on the theoretical findings in this thesis. It further concentrated on aspects of communication and time management, as these are important for agile processes in general and in particular for how work is done in the VEC. The main challenge to be solved in the VEC was to streamline existing communication and task management activities by introducing a new solution. Due to the software development environment used, the Microsoft Team Foundation Server (TFS) was a natural choice.

As the TFS standard configuration did not suit the existing processes, the main task was to implement a solution reflecting the team's specific needs. Therefore a number of custom Work Item Definitions had to be created together with underlying workflows and a little C# program for user notification in case a task in the system is about to become overdue.

Though the product of this work has proven itself to be stable and useful, critically challenging it showed a number of shortcomings mostly rooting from TFS' inadequacies and the complex project structure given. Still, it is used on a daily basis and definitely represents a step in the right direction. It shows that a central communication platform benefits the quality and execution speed of diversified teams.

Bibliography

[1] Gellersen HW, Wicke R, Gaedke M. WebComposition: An Object-Oriented Support System for the Web engineering Lifecycle. Computer Networks and ISDN Systems. 1997;29:865–1553.

[2] (editor) WS. Web engineering: Principles and Techniques. Idea Group Publishing; 2005.

[3] Murugesan S, Ginige A. Web engineering: Introduction and Perspectives. In: Web engineering: Principles and Techniques. Idea Group Publishing; 2005. .

[4] Murugesan S, Deshpande Y, Hansen S, Ginige A. Web engineering: A New Discipline for Development of Web-Based Systems; 2001. Department of Computing and Information Systems, University of Western Sydney.

[5] Kappel G, Pröll B, Reich S, Retschitzegger W. Web Engineering - the Discipline of Systematic Development of Web Applications. Vienna: John Wiley and Sons, Ltd; 2006.

[6] Wallace D, Raggett I, Aufgang J. Extreme Programming for Web Projects. The XP Series. Pearson Education Limited; 2003.

[7] Dumke R, Lother M, Wille C, Zbrog F. Web Engieering. Pearson Studium; 2003.

[8] McIlroy MD. Software Engineering: Report on a conference sponsored by the NATO Science Committee. In: NATO Software Engineering Conference. NATO Scientific Affairs Division; 1968. p. 138–155.

[9] Benington HD. Production of Large Computer Programs (Reprint). In: Proceedings of the 9th international conference on Software Engineering. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press; 1987. p. 299–310.

[10] Hosier WA. Pitfalls and Safeguards in Real-Time Digital Systems with Emphasis on Programming. Engineering Management, IRE Transactions on. 1961 June;EM-8(2):99–115.

[11] Royce WW. Managing the Development of Large Software Systems:

Concepts and Techniques. Proceedings of the IEEE WESTCON, Los Angeles. 1970 August;p. 1–9. Reprinted in *Proceedings* of the Ninth International Conference on Software Engineering, March 1987, pp. 328–338. Available from: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf.

[12] Boehm BW. A Spiral Model of Software Development and Enhancement. Computer. 1988 May;21(5):61–72. Available from: http://portal.acm.org/citation.cfm?id=45797.45801.

[13] Highsmith J. Agile Software Development Ecosystems. 1st ed. The Agile Software Development Series. Addison-Wesley Longman; 2002.

[14] Pressman RS, Lowe D. Web engineering: A Practitioners Approach. vol. 1. McGraw-Hill Higher Education; 2009.

[15] Beck K. Extreme Programming Explained - Embrace Change. 2nd ed. O'Hagan D, editor. The XP Series. Addison-Wesley; 2005.

[16] Schwaber K, Beedle M. Agile Software Development with Scrum. Prentice Hall; 2002.

[17] Cohn M. Succeeding with Agile - Software Development using Scrum. Guzikowski C, editor. Signature Series. Addison-Wesley; 2009.

[18] Kruchten P. The Rational Unified Process. An Introduction. 2nd ed. Erickson K, editor. Object Technology Series. Pearson Education Limited; 2000.

[19] Abrahamsson P, Salo O, Ronkainen J, Warsta J. Agile Software Development Methods - Review and Analysis. Kettunen M, editor. VTT Technical Research Centre of Finland; 2002.

[20] Boehm B, Turner R. Balancing Agility and Discipline: A Guide for the Perplexed. Addison Wesley; 2003.

[21] Berners-Lee T. Information Management: A Proposal; 1989. http://www.w3.org/History/1989/proposal.html.

[22] Berners-Lee T. WorldWideWeb: Proposal for a HyperText Project; 1990. http://www.w3.org/Proposal.

[23] ICANN; 2009. http://www.icann.org/.

[24] Internetworldstats.com; 2009. http://www.internetworldstats.com/stats.htm. Available from: http://www.internetworldstats.com/stats.htm [cited 29-Jan-2009].

[25] July 2010 Web Server Survey; 2010.
http://news.netcraft.com/archives/2010/07/16/july-2010-web-server-survey-16.html.

[26] Schneider G. Electronic Commerce. 7th ed. Course Technology Inc.; 2006.

[27] Hypertext Transfer Protocol – HTTP/1.1; 1999. Network Working Group.

[28] van Hengstum E, Pras A, Hazewinkel H. Management of the World-Wide Web; 1997. Centre for Telematics and Information Technology, University of Twente.

[29] Postel JB. Simple Mail Transfer Protocol, Internet RFC-821; 1982.

[30] Myers J, Mellon C, Rose M. Post Office Protocol - Version 3; 1996. Available from: http://www.ietf.org/rfc/rfc1939.txt [cited 28-Dec-2009].

[31] Crispin M. Internet Message Access Protocol - Version 4.rev1; 2003. The Internet Society - Network Working Group.

[32] Raggett D, Hors AL, Jacobs I. HTML 4.01 Specification. W3C; 1999.

[33] McCarty W. A Serious Beginner's Guide to Hypertext Research; 2001. http://www.allc.org/reports/map/hyperbib.html. Available from: http://www.allc.org/reports/map/hyperbib.html [cited 11-Jun-2009].

[34] Nickul D. Service Oriented Architecture (SOA) and Specialized Messaging Patterns; 2007. Adobe technical paper; http://www.adobe.com.

[35] Hinchcliffe D. Comparing Amazon's and Google's Platform-as-a-Service (PaaS) Offerings; 2008.
http://blogs.zdnet.com/Hinchcliffe/?p=166&tag=btxcsim.

[36] Duhl J. Rich Internet Applications; 2003. Adobe technical paper; http://www.adobe.com.

[37] Kappel G, Michlmayr E, Pröll B, Reich S, Retschitzegger W. Web Engineering - Old Wine in New Bottles? In: Proceedings of the 4th International Conference on Web Engineering. LNCS 3140; 2004. p. 6–12.

[38] Nielsen J. Differences Between Print Design and Web Design; 1999.
http://www.useit.com/alertbox/990124.html.

[39] ISO9126; 1991. International Organization for Standardization.

[40] Nielsen J. Designing Web Usability: The Practice of Simplicity. 1st ed. Thousand Oaks, CA, USA: New Riders Publishing; 1999.

[41] Lauesen S. User Interface Design - A Software Engineering Perspective. Pearson Education Limited; 2005.

[42] Introduction to Web Accessibility; 2009. http://www.w3.org/WAI/intro/accessibility.php. Available from: http://www.w3.org/WAI/intro/accessibility.php [cited 28-Dec-2009].

[43] W3C - All Standards and Drafts; 2009. http://www.w3.org/TR/.

[44] Shubin H, Meehan MM. Designing applications for the Web vs for the desktop. Interaction Design, Inc.; 1997. http://www.user.com/webapps/webapps.htm.

[45] Farkas DK. Hypertext and Hypermedia. Berkshire Publishing; 2004.

[46] Gaedke M, Meinecke J. The Web as an Application Platform. In: Web engineering - Modelling and Implementing Web applications. Springer; 2008. .

[47] Rossi G, Schwabe D, Olsina L, Pastor O. Overview of Design Issues for Web Applications Development. Springer; 2008.

[48] Conklin J. Computer-supported cooperative work: a book of readings. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1988. p. 423–475. Available from: http://dl.acm.org/citation.cfm?id=49504.49520.

[49] Preciado JC, Linaje M, Comai S, Sanchez-Figueroa F. Designing Rich Internet Applications with Web Engineering Methodologies; 2007. Quercus Software Engineering group, Universidad de Extremadura.

[50] Preciado JC, Linaje M, Sanchez F, Comai S. Necessity of methodologies to model Rich Internet Applications. In: Proceedings of the Seventh IEEE International Symposium on Web Site Evolution. Washington, DC, USA: IEEE Computer Society; 2005. p. 7–13. Available from: http://dl.acm.org/citation.cfm?id=1092361.1092692.

[51] The Web Standards Project - About; 2009. http://www.webstandards.org/about/. Available from: http://www.webstandards.org/about/ [cited 08-Dec-2009].

[52] The Web Standards Project - ACID Tests; 2009. http://www.acidtests.org/. Available from: http://www.acidtests.org/ [cited 08-Dec-2009].

[53] CHAOS Summary 2009. The Standish Group; 2009.

[54] Emam KE, Koru AG. A Replicated Survey of IT Software Project Failures. IEEE Software; 2008.

[55] Royce W. Improving Software Development Economics Part II: Reducing Software Product Complexity and Improving Software Processes. The Rational Edge. 2001;.

[56] Mendes E. Cost Estimation Techniques for Web Projects. IGI Publishing; 2007.

[57] Reifer D. Ten Deadly Risks in Internet and Intranet Software Development. IEEE Software. 2002;19(2):12–14.

[58] Andrews K. Information Architecture and Web Usability; 2009. Graz University of Technology.

[59] Bevan N. Usability Issues in Web Site Design; 1998. National Physical Laboratory, Usability Services.

[60] Scacchi W. In: Process Models in Software Engineering. 2nd ed. John Wiley and Sons, Ltd; 2001. .

[61] Petersen K, Wohlin C, Baca D. The Waterfall Model in Large-Scale Development. In: 10th International Conference on Product-Focused Software Process Improvement. Blekinge Institute of Technology. Springer; 2009. p. 386–400.

[62] Brooks FP Jr. No Silver Bullet — Essence and Accidents of Software Engineering. vol. 20. IEEE Computer Society Press; 1987. p. 10–19.

[63] Pressman RS. Software Engineering: A Practitioner's Approach. 5th ed. McGraw-Hill Higher Education; 2001.

[64] Leffingwell D. Scaling Software Agility. Addison-Wesley Professional Agile Software Development Series. Addison-Wesley Professional; 2007.

[65] Cockburn A. Agile Software Development. 1st ed. The Agile Software Development Series. Addison-Wesley Professional; 2001.

[66] Consortium D. Introduction to DSDM - Public Version; [cited 2010-09-26]. Available from: http://www.dsdm.org/version4/2/public/.

[67] Cockburn A. Crystal Clear a Human-Powered Methodology for Small Teams. Addison-Wesley Professional; 2004. Available from: http://portal.acm.org/ft_gateway.cfm?id=1406822&type=safari&coll=GUIDE&dl=GUIDE&CFID=106127749&CFTOKEN=45871543.

[68] Palmer SR, Felsing M. A Practical Guide to Feature-Driven Development. Pearson Education; 2001.

[69] Poppendieck M, Poppendieck T. Lean Software Development: An Agile Toolkit. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2003. Available from: http://portal.acm.org/ft_gateway.cfm?id=829556&type=safari&coll=GUIDE&dl=GUIDE&CFID=106127749&CFTOKEN=45871543.

[70] Highsmith J. Agile Project Management: Creating Innovative Products. 2nd ed. Addison-Wesley Professional; 2009.

[71] Ambler SW. The Agile Unified Process (AUP); 2009. http://www.ambysoft.com/unifiedprocess/agileUP.html.

[72] Manifesto for Agile Software Development; 2001. http://agilemanifesto.org/.

[73] Principles behind the Agile Manifesto; 2001. http://agilemanifesto.org/principles.html.

[74] Kroll P, Kruchten P. The Rational Unified Process Made Easy. Pearson Education Limited; 2003. Available from: http://books.google.at/books?id=7FSf5661dfMC&printsec=frontcover#v=onepage&q&f=false.

[75] Iacob I. Extreme Programming and Rational Unified Process – Contrasts or Synonyms?; 2004. Faculty of Computer Science for Business Management, Romanian – American University, Bucharest, Romania.

[76] Hirsch M. Making RUP Agile;. Zühlke Engineering AG, Department of Computer and Information Science, Linköping University.

[77] Ambler SW. A Manager's Introduction to The Rational Unified Process (RUP); 2004. Ambysoft.

[78] IBM Rational Unified Process Reference and Certification Guide: Solution Designer. 1st ed. IBM Press; 2007.

[79] Rational Unified Process - Best Practices for Software Development Teams; 1998. Rational Software Corporation.

[80] Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Shanklin JC, editor. Addison-Wesley object technology series. Addison-Wesley Longman; 1999.

[81] Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language

Reference Manual. Shanklin JC, editor. The Addison-Wesley object technology series. Addison-Wesley; 1999.

[82] Kruchten P, Royce W. A Rational Development Process. CrossTalk - The Journal of Defense Software Engineering. 1996;p. 11–16.

[83] Paulk MC. Extreme Programming from a CMM Perspective; 2001. XP Universe, Raleigh, NC, 23-25 July 2001.

[84] Wells D. Extreme Programming: A Gentle Introduction; 2009. http://www.extremeprogramming.org/.

[85] Jarvis B, Gristock S. JPMorgan Chase Case Study: Extreme Programming (XP), Six Sigma, CMMI - How they can work together;. http://www.sei.cmu.edu/library/assets/jarvis-gristock.pdf.

[86] Beck K. Extreme Programming Explained - Embrace Change. 1st ed. The XP Series. Addison-Wesley; 2000.

[87] Marchenko A. XP Practices; 2007. http://agilesoftwaredevelopment.com/xp/practices.

[88] Williams L, Kessler RR, Cunningham W, Jeffries R. Strengthening the Case for Pair-Programming; 1999. University of Utah.

[89] Pichler R. Scrum - Agiles Projectmanagement erfolgreich einsetzen. 1st ed. Preisendanz C, editor. dpunkt.verlag; 2008.

[90] Rising L, Janoff NS. The Scrum Software Development Process for Small Teams. IEEE Softw. 2000 July;17:26–32. Available from: http://dl.acm.org/citation.cfm?id=624638.626150.

[91] Schwaber K. Agile Project Management with Scrum. 2nd ed. Atkins K, editor. Microsoft Press; 2004.

[92] Deemer P, Benefield G. The Scrum Primer - An Introduction to Agile Project Management with Scrum; 2007. goodagile.

[93] Kniberg H. Scrum and XP from the Trenches - How We Do Scrum. C4Media; 2007.

[94] Iacovelli A, Souveyet C. Framework for Agile Methods Classification. In: Ebersold S, Front A, Lopistéguy P, Nurcan S, Franch X, Hunt E, et al., editors. MoDISE-EUS. vol. 341 of CEUR Workshop Proceedings. CEUR-WS.org CEUR-WS.org; 2008. p. 91–102.

[95] Strode DE. The Agile Methods : An Analytical Comparison of Five Agile

Methods and an Investigation of Their Target Environment. Massey University. Department of Information Systems; 2007.

[96] Robert DT, France R, Rumpe B. Limitations of Agile Software Processes. In: In Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002). Springer; 2000. p. 43–46.

[97] Abrahamsson P, Warsta J, Siponen MT, Ronkainen J. New Directions on Agile Methods: A Comparative Analysis. In: ICSE. IEEE Computer Society; 2003. p. 244–254.

[98] Boehm BW. Get Ready for Agile Methods, with Care. IEEE Computer. 2002;35(1):64–69.

[99] Awad MA. A Comparison between Agile and Traditional Software Development Methodologies. University of Western Australia; 2005.

[100] Harrison W. Is Software Engineering as We Know it over the Hill? IEEE Software. 2003 May/June;p. 5–7.

[101] Reifer DJ. How Good are Agile Methods? IEEE Software. 2002;19(4):16–18.

[102] Larman C, Bittner K. How to Fail with the Rational Unified Process: Seven Steps to Pain and Suffering. Valtech Technologies and Rational Software; 2001.

[103] Anderson J. Agile over RUP (Part 2); 2009. `http://agileconsulting.blogspot.com/2009/02/agile-over-rup-part-2.html`.

[104] Layman L, Williams L, Cunningham L. Exploring Extreme Programming in Context: An Industrial Case Study. In: IEEE Agile Development Conference; 2004. p. 32–41.

[105] Reifer DJ. How to Get the Most out of Extreme Programming/Agile Methods. In: Wells D, Williams LA, editors. XP/Agile Universe. vol. 2418 of Lecture Notes in Computer Science. Springer; 2002. p. 185–196.

[106] Williams L, Cockburn A. Guest Editors' Introduction: Agile Software Development: It's about Feedback and Change. Computer. 2003;36:39–43.

[107] Mar K, Schwaber K. Scrum with XP; 2002. `http://www.informit.com/articles/article.aspx?p=26057`.

[108] Pollice G. RUP and XP Part I: Finding Common Ground. The Rational

Edge. 2001;Available from: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/archives/rup.html.

[109] Krebs J. RUP in the Dialogue with Scrum. The Rational Edge. 2005;Available from: http://www.ibm.com/developerworks/rational/library/feb05/krebs/index.html.

[110] Dennis A, Drouault-Gardrat P, Kupchyk A, Smith R. In: Marketing, E-Commerce and Advertising in the Pharmaceutical Industry: France, the Uk and the Us. Practical Law Company; 2007. .

[111] Meier JD, Taylor J, Bansode P, Mackman A, Jones K. Team Development with Visual Studio Team Foundation Server - patterns & practices. Microsoft Corporation; 2007.

[112] Microsoft. Process Templates and Tools; 2010. http://msdn.microsoft.com/en-us/vstudio/aa718795.aspx. Available from: http://msdn.microsoft.com/en-us/vstudio/aa718795.aspx.

[113] Turner MSV. Micrisoft Solutions Framework Essentials. Finnel L, editor. Microsoft Press; 2006.

Appendix A

TFS Work Items XML

```xml
<WITD application="Work item type editor" version="1.0" xmlns:witd="http://schemas.microsoft.com/VisualStudio/2005/workitemtracking/typedef">
  <WORKITEMTYPE name="Task">
    <DESCRIPTION>General WorkItem to potentially cover all tasks to be worked on within the VEC</DESCRIPTION>
    <FIELDS>
      <FIELD name="Title" refname="System.Title" type="String" reportable="dimension">
        <HELPTEXT>Short description of the task used to differentiate it in a list or report</HELPTEXT>
        <REQUIRED />
      </FIELD>
      <FIELD name="State" refname="System.State" type="String" reportable="dimension">
        <HELPTEXT>The workflow state of the task</HELPTEXT>
      </FIELD>
      <FIELD name="Rev" refname="System.Rev" type="Integer" reportable="dimension" />
      <FIELD name="Changed By" refname="System.ChangedBy" type="String" reportable="dimension">
        <VALIDUSER />
      </FIELD>
      <FIELD name="Issue" refname="Microsoft.VSTS.Common.Issue" type="String" reportable="dimension">
        <HELPTEXT>Used to highlight a task, e.g., to mark it as an issue</HELPTEXT>
        <REQUIRED />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Yes" />
          <LISTITEM value="No" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="No" />
    </FIELD>
    <FIELD name="State Change Date" refname="Microsoft.VSTS.Common.StateChangeDate" type="DateTime">
      <WHENCHANGED field="System.State">
        <SERVERDEFAULT from="clock" />
      </WHENCHANGED>
      <WHENNOTCHANGED field="System.State">
        <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Activated Date" refname="Microsoft.VSTS.Common.ActivatedDate" type="DateTime" reportable="dimension">
      <WHENNOTCHANGED field="System.State">
        <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Activated By" refname="Microsoft.VSTS.Common.ActivatedBy" type="String" reportable="dimension">
      <WHENNOTCHANGED field="System.State">
        <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Reason" refname="System.Reason" type="String" reportable="dimension">
      <HELPTEXT>The reason why the task is in the current state</HELPTEXT>
    </FIELD>
    <FIELD name="Assigned To" refname="System.AssignedTo" type="String" reportable="dimension">
      <ALLOWEXISTINGVALUE />
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="[project]\Contributors" />
      </ALLOWEDVALUES>
      <REQUIRED for="[global]\Team Foundation Valid Users" />
    </FIELD>
    <FIELD name="Work Item Type" refname="System.WorkItemType" type="String" reportable="dimension" />
    <FIELD name="Closed By" refname="Microsoft.VSTS.Common.ClosedBy" type="String" reportable="dimension">
    <WHENNOTCHANGED field="System.State">
     <READONLY />
    </WHENNOTCHANGED>
   </FIELD>
   <FIELD name="Closed Date" refname="Microsoft.VSTS.Common.ClosedDate" type="DateTime" reportable="dimension">
    <WHENNOTCHANGED field="System.State">
     <READONLY />
    </WHENNOTCHANGED>
   </FIELD>
   <FIELD name="Rank" refname="Microsoft.VSTS.Common.Rank" type="String" reportable="dimension">
    <HELPTEXT>Stack rank to prioritize work</HELPTEXT>
   </FIELD>
   <FIELD name="Created Date" refname="System.CreatedDate" type="DateTime" reportable="dimension" />
   <FIELD name="Created By" refname="System.CreatedBy" type="String" reportable="dimension" />
   <FIELD name="Exit Criteria" refname="Microsoft.VSTS.Common.ExitCriteria" type="String" reportable="dimension">
    <HELPTEXT>Flag to determine if this scenario should be tracked as an exit criteria for the iteration</HELPTEXT>
    <REQUIRED />
    <ALLOWEDVALUES expanditems="true">
     <LISTITEM value="Yes" />
     <LISTITEM value="No" />
    </ALLOWEDVALUES>
    <DEFAULT from="value" value="No" />
   </FIELD>
   <FIELD name="Integration Build" refname="Microsoft.VSTS.Build.IntegrationBuild" type="String" reportable="dimension">
    <HELPTEXT>The build in which the task was completed</HELPTEXT>
    <SUGGESTEDVALUES expanditems="true">
     <LISTITEM value="&lt;None&gt;" />
    </SUGGESTEDVALUES>
    <SUGGESTEDVALUES expanditems="true" filteritems="excludegroups">
     <GLOBALLIST name="Builds - DNN.Template.4.9.1" />
```

```
      </SUGGESTEDVALUES>
    </FIELD>
    <FIELD name="Discipline" refname="Microsoft.VSTS.
       Common.Discipline" type="String" reportable="
       dimension">
      <HELPTEXT>The discipline to which the task belongs</
         HELPTEXT>
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="Development" />
        <LISTITEM value="Test" />
        <LISTITEM value="Deployment" />
        <LISTITEM value="Launch Management" />
      </ALLOWEDVALUES>
    </FIELD>
    <FIELD name="Baseline Work" refname="Microsoft.VSTS.
       Scheduling.BaselineWork" type="Double" reportable="
       measure" formula="sum">
      <HELPTEXT>The number of hours of work from the
         baseline plan</HELPTEXT>
    </FIELD>
    <FIELD name="Start Date" refname="Microsoft.VSTS.
       Scheduling.StartDate" type="DateTime" reportable="
       dimension">
      <HELPTEXT>The date to start the task</HELPTEXT>
      <DEFAULT from="clock" />
    </FIELD>
    <FIELD name="Finish Date" refname="Microsoft.VSTS.
       Scheduling.FinishDate" type="DateTime" reportable="
       dimension">
      <HELPTEXT>The date to finish the task</HELPTEXT>
      <WHEN field="COMPANY.Task.GlobalPriority" value="2
         Important">
        <REQUIRED for="[global]\Team Foundation Valid
           Users" />
      </WHEN>
    </FIELD>
    <FIELD name="Task Hierarchy" refname="Microsoft.VSTS.
       Scheduling.TaskHierarchy" type="String" reportable=
       "dimension">
      <HELPTEXT>A string representing MS-Project context
         for the given task</HELPTEXT>
    </FIELD>
    <FIELD name="Completed Work" refname="Microsoft.VSTS.
```

```
       Scheduling.CompletedWork" type="Double" reportable=
       "measure" formula="sum">
      <HELPTEXT>The number of hours that have been
         completed for this task</HELPTEXT>
    </FIELD>
    <FIELD name="Remaining Work" refname="Microsoft.VSTS.
       Scheduling.RemainingWork" type="Double" reportable=
       "measure" formula="sum">
      <HELPTEXT>An estimate of the number of hours
         remaining to complete the task</HELPTEXT>
    </FIELD>
    <FIELD name="Description" refname="System.Description"
        type="PlainText" />
    <FIELD name="Module" refname="COMPANY.Module" type="
       String">
      <HELPTEXT>COMPANY.Module</HELPTEXT>
    </FIELD>
    <FIELD name="History" refname="System.History" type="
       History">
      <HELPTEXT>Discussion thread and other historical
         information</HELPTEXT>
    </FIELD>
    <FIELD name="PlanningState" refname="COMPANY.
       PlanningState" type="String">
      <HELPTEXT>PlanningState</HELPTEXT>
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="Unplanned" />
        <LISTITEM value="Planned" />
      </ALLOWEDVALUES>
    </FIELD>
    <FIELD name="Project" refname="COMPANY.Project" type="
       String" reportable="dimension">
      <HELPTEXT>Project</HELPTEXT>
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="SMA" />
        <LISTITEM value="WeaningPlanner" />
        <LISTITEM value="GrowthChart" />
      </ALLOWEDVALUES>
    </FIELD>
    <FIELD name="RequestSource" refname="COMPANY.
       Deployment.RequestSource" type="String" reportable=
       "detail" />
```

```xml
      <FIELD name="Website" refname="COMPANY.Deployment.Website" type="String" reportable="dimension">
        <HELPTEXT>The website concerned (including URLs, PortalID, ...)</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <GLOBALLIST name="VEC_Websites_Instances" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="DevTimeline" refname="COMPANY.Task.DevTimeline" type="DateTime">
        <HELPTEXT>Timeline set by executing person</HELPTEXT>
      </FIELD>
      <FIELD name="Completion" refname="COMPANY.Task.Completion" type="Integer">
        <HELPTEXT>Completion Percentage</HELPTEXT>
      </FIELD>
      <FIELD name="Requestor" refname="COMPANY.Task.RequestSource" type="String" reportable="dimension">
        <HELPTEXT>Who requested this task?</HELPTEXT>
        <ALLOWEXISTINGVALUE />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="[project]\Contributors" />
        </ALLOWEDVALUES>
        <DEFAULT from="currentuser" />
      </FIELD>
      <FIELD name="Prio" refname="COMPANY.Task.GlobalPriority" type="String" reportable="dimension">
        <HELPTEXT>Global Task Importance</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1 Normal" />
          <LISTITEM value="2 Important" />
          <LISTITEM value="3 Emergency" />
          <LISTITEM value="0 Low" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="1 Normal" />
      </FIELD>
      <FIELD name="Task Type" refname="COMPANY.Task.TaskType" type="String" reportable="dimension">
        <HELPTEXT>Defines the type (Bug, Feature Request,...) of a task</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Bug" />
          <LISTITEM value="Test" />
          <LISTITEM value="Feature Request" />
          <LISTITEM value="Support Request" />
        </ALLOWEDVALUES>
        <REQUIRED for="[global]\Team Foundation Valid Users"
            />
      </FIELD>
      <FIELD name="RelatedLinkCount" refname="System.
         RelatedLinkCount" type="Integer" />
      <FIELD name="Project Code" refname="COMPANY.Task.
         ProjectCode" type="String" reportable="dimension">
        <HELPTEXT>The VEC project code</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <GLOBALLIST name="VEC_Projects" />
        </ALLOWEDVALUES>
        <REQUIRED for="[global]\Team Foundation Valid Users"
            />
      </FIELD>
      <FIELD name="Bug Severity" refname="COMPANY.Task.
         Severity" type="String" reportable="dimension">
        <HELPTEXT>Severity of a bug</HELPTEXT>
        <SUGGESTEDVALUES expanditems="true">
          <LISTITEM value="1 - Cosmetic" />
          <LISTITEM value="2 - Functionality affected" />
          <LISTITEM value="3 - Showstopper" />
        </SUGGESTEDVALUES>
        <WHENNOT field="COMPANY.Task.TaskType" value="Bug">
          <READONLY for="[global]\Team Foundation Valid
             Users" />
        </WHENNOT>
        <WHEN field="COMPANY.Task.TaskType" value="Bug">
          <REQUIRED for="[global]\Team Foundation Valid
             Users" />
        </WHEN>
      </FIELD>
      <FIELD name="Estimated Effort" refname="COMPANY.Task.
         EstimatedEffort" type="Double" reportable="
         dimension">
        <HELPTEXT>The estimated effort in days</HELPTEXT>
      </FIELD>
```

```xml
      <FIELD name="Phase" refname="COMPANY.Task.Phase" type="String" reportable="detail">
        <HELPTEXT>Phase in which the Task falls into (e.g. Maintenance, Project)</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Project" />
          <LISTITEM value="Maintenance" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="StartConstraint" refname="COMPANY.Task.StartConstraint" type="String" reportable="detail">
        <HELPTEXT>Constraint for Start Date of a WorkItem</HELPTEXT>
      </FIELD>
      <FIELD name="Task Owner" refname="COMPANY.Task.Owner" type="String" reportable="detail">
        <HELPTEXT>Person responsible for the Work Item</HELPTEXT>
        <ALLOWEXISTINGVALUE />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="[project]\Contributors" />
        </ALLOWEDVALUES>
        <DEFAULT from="field" field="System.CreatedBy" />
      </FIELD>
      <FIELD name="DescriptionHTML" refname="COMPANY.Task.DescriptionHtml" type="HTML" />
      <FIELD name="LaunchRelevancy" refname="COMPANY.Task.LaunchRelevancy" type="String" reportable="dimension">
        <HELPTEXT>Is this WI launch relevant?</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Yes" />
          <LISTITEM value="No" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="-" />
        <REQUIRED for="[global]\Team Foundation Valid Users" />
      </FIELD>
      <FIELD name="Group" refname="COMPANY.Task.Group" type="String" reportable="dimension">
        <HELPTEXT>To which group does the task belong to (e.g. SMA, Core, ...)</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
```

```xml
          <LISTITEM value="SMA" />
          <LISTITEM value="Core" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="Core" />
      </FIELD>
      <FIELD name="Iteration Path" refname="System.IterationPath" type="TreePath" reportable="dimension">
        <HELPTEXT>The iteration of the product with which this task is associated</HELPTEXT>
      </FIELD>
      <FIELD name="IterationID" refname="System.IterationId" type="Integer" />
      <FIELD name="ExternalLinkCount" refname="System.ExternalLinkCount" type="Integer" />
      <FIELD name="Team Project" refname="System.TeamProject" type="String" reportable="dimension" />
      <FIELD name="HyperLinkCount" refname="System.HyperLinkCount" type="Integer" />
      <FIELD name="AttachedFileCount" refname="System.AttachedFileCount" type="Integer" />
      <FIELD name="Node Name" refname="System.NodeName" type="String" />
      <FIELD name="Area Path" refname="System.AreaPath" type="TreePath" reportable="dimension">
        <HELPTEXT>The area of the product with which this task is associated</HELPTEXT>
      </FIELD>
      <FIELD name="Revised Date" refname="System.RevisedDate" type="DateTime" />
      <FIELD name="Changed Date" refname="System.ChangedDate" type="DateTime" reportable="dimension" />
      <FIELD name="ID" refname="System.Id" type="Integer" reportable="dimension" />
      <FIELD name="AreaID" refname="System.AreaId" type="Integer" />
      <FIELD name="Authorized As" refname="System.AuthorizedAs" type="String" />
    </FIELDS>
    <WORKFLOW>
      <STATES>
        <STATE value="Open">
          <FIELDS>
```

```
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate
               ">
              <EMPTY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <EMPTY />
            </FIELD>
          </FIELDS>
        </STATE>
        <STATE value="Closed" />
        <STATE value="In Progress" />
        <STATE value="Executed" />
        <STATE value="Cancelled" />
        <STATE value="On Hold" />
      </STATES>
      <TRANSITIONS>
        <TRANSITION from="" to="Open">
          <REASONS>
            <DEFAULTREASON value="New" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="System.AssignedTo">
              <DEFAULT from="currentuser" />
            </FIELD>
          </FIELDS>
        </TRANSITION>
        <TRANSITION from="Closed" to="Open">
          <REASONS>
            <DEFAULTREASON value="Problem found" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedBy">
```

```xml
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="System.AssignedTo">
              <COPY from="field" field="Microsoft.VSTS.
                 Common.ClosedBy" />
            </FIELD>
          </FIELDS>
        </TRANSITION>
        <TRANSITION from="Open" to="In Progress">
          <REASONS>
            <DEFAULTREASON value="Work started" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="Open">
          <REASONS>
            <DEFAULTREASON value="Other Task Opened" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Open" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
            <REASON value="Covered by other Task" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Cancelled" to="Open">
          <REASONS>
            <DEFAULTREASON value="Opened up again" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
            <REASON value="Covered by other Task" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Executed" to="Closed" for="[global
           ]\VEC ProQ Users">
          <REASONS>
            <DEFAULTREASON value="Tested OK" />
            <REASON value="Not Tested" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate
               ">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedBy">
              <READONLY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedDate">
              <READONLY />
            </FIELD>
          </FIELDS>
          <ACTIONS>
            <ACTION value="Microsoft.VSTS.Actions.Checkin" /
               >
          </ACTIONS>
        </TRANSITION>
        <TRANSITION from="Open" to="Executed">
          <REASONS>
            <DEFAULTREASON value="Development finished" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate
               ">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedBy">
              <READONLY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedDate">
              <READONLY />
            </FIELD>
          </FIELDS>
          <ACTIONS>
            <ACTION value="Microsoft.VSTS.Actions.Checkin" />
          </ACTIONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="Executed">
          <REASONS>
            <DEFAULTREASON value="Development finished" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Executed" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
            <REASON value="Covered by other Task" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Executed" to="Open">
          <REASONS>
            <DEFAULTREASON value="Tested Error" />
            <REASON value="Not finished" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="On Hold">
          <REASONS>
            <DEFAULTREASON value="See History" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Open" to="On Hold">
          <REASONS>
            <DEFAULTREASON value="See History" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="On Hold" to="Cancelled">
          <REASONS>
```

```
            <DEFAULTREASON value="Deferred" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="On Hold" to="Open">
          <REASONS>
            <DEFAULTREASON value="Opened up again" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="On Hold" to="In Progress">
          <REASONS>
            <DEFAULTREASON value="Opened up again" />
          </REASONS>
        </TRANSITION>
      </TRANSITIONS>
    </WORKFLOW>
    <FORM>
      <Layout>
        <Group Label=" ">
          <Column PercentWidth="100">
            <Control FieldName="System.Title" Type="
               FieldControl" Label="&amp;Title:"
               LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="General Information">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.GlobalPriority"
                Type="FieldControl" Label="Priority:"
               LabelPosition="Left" />
            <Control FieldName="Microsoft.VSTS.Scheduling.
               FinishDate" Type="DateTimeControl" Label="
               Timeline:" LabelPosition="Left" />
            <Control FieldName="Microsoft.VSTS.Scheduling.
               StartDate" Type="DateTimeControl" Label="
               Start not before:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.StartConstraint
               " Type="FieldControl" Label="Start
               Constraint:" LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Deployment.
               RequestSource" Type="FieldControl" Label="
               External Reference:" LabelPosition="Left" />
```

```xml
            <Control FieldName="COMPANY.Task.TaskType" Type=
               "FieldControl" Label="Task Type:"
               LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.Severity" Type=
               "FieldControl" Label="Severity:"
               LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.LaunchRelevancy
               " Type="FieldControl" Label="Is Launch
               relevant:" LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="Project Information">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.ProjectCode"
               Type="FieldControl" Label="Project Code:"
               LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.Phase" Type="
               FieldControl" Label="Phase:" LabelPosition="
               Left" />
          </Column>
        </Group>
        <Group Label="People">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.RequestSource"
               Type="FieldControl" Label="Requestor:"
               LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.Owner" Type="
               FieldControl" Label="Task Owner:"
               LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.Group" Type="
               FieldControl" Label="Assigned Group:"
               LabelPosition="Left" />
            <Control FieldName="System.AssignedTo" Type="
               FieldControl" Label="Assi&amp;gned to:"
               LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="State Information">
          <Column PercentWidth="50">
```

```xml
      <Control FieldName="COMPANY.Task.EstimatedEffort" Type="FieldControl" Label="Estimated Effort (days):" LabelPosition="Left" />
      <Control FieldName="COMPANY.Task.DevTimeline" Type="DateTimeControl" Label="Estimated Timeline:" LabelPosition="Left" />
      <Control FieldName="COMPANY.Task.Completion" Type="FieldControl" Label="Percentage complete:" LabelPosition="Left" />
     </Column>
     <Column PercentWidth="50">
      <Control FieldName="System.State" Type="FieldControl" Label="&amp;State:" LabelPosition="Left" />
      <Control FieldName="System.Reason" Type="FieldControl" Label="&amp;Reason:" LabelPosition="Left" />
     </Column>
    </Group>
    <Group>
     <Column PercentWidth="50">
      <Control FieldName="COMPANY.Task.DescriptionHtml" Type="HtmlFieldControl" Label="Des&amp;cription:" LabelPosition="Top" Dock="Fill" />
     </Column>
     <Column PercentWidth="50">
      <Control FieldName="System.History" Type="WorkItemLogControl" Label="Comment/History:" LabelPosition="Top" Dock="Fill" />
     </Column>
    </Group>
    <TabGroup>
     <Tab Label="File Attachments">
      <Control Type="AttachmentsControl" LabelPosition="Top" Dock="Fill" />
     </Tab>
     <Tab Label="Links">
      <Control Type="LinksControl" LabelPosition="Top" />
     </Tab>
    </TabGroup>
   </Layout>
  </FORM>
  </WORKITEMTYPE>
</WITD>
```

Listing A.1: XML Structure - Work Item Type “Task”

```
<WITD application="Work item type editor" version="1.0"
   xmlns:witd="http://schemas.microsoft.com/VisualStudio
   /2005/workitemtracking/typedef">
  <WORKITEMTYPE name="Deployment Request">
    <DESCRIPTION>WorkItem to describe the deployment from
       and to websites</DESCRIPTION>
    <FIELDS>
      <FIELD name="Title" refname="System.Title" type="
         String" reportable="dimension">
        <HELPTEXT>Short description of the task used to
           differentiate it in a list or report</HELPTEXT>
      </FIELD>
      <FIELD name="State" refname="System.State" type="
         String" reportable="dimension">
        <HELPTEXT>The workflow state of the task</HELPTEXT>
      </FIELD>
      <FIELD name="Rev" refname="System.Rev" type="Integer"
         reportable="dimension" />
      <FIELD name="Changed By" refname="System.ChangedBy"
         type="String" reportable="dimension">
        <VALIDUSER />
      </FIELD>
      <FIELD name="Issue" refname="Microsoft.VSTS.Common.
         Issue" type="String" reportable="dimension">
        <HELPTEXT>Used to highlight a task, e.g., to mark it
            as an issue</HELPTEXT>
        <REQUIRED />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Yes" />
          <LISTITEM value="No" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="No" />
      </FIELD>
      <FIELD name="State Change Date" refname="Microsoft.
         VSTS.Common.StateChangeDate" type="DateTime">
        <WHENCHANGED field="System.State">
          <SERVERDEFAULT from="clock" />
        </WHENCHANGED>
        <WHENNOTCHANGED field="System.State">
```

```xml
          <READONLY />
        </WHENNOTCHANGED>
      </FIELD>
      <FIELD name="Activated Date" refname="Microsoft.VSTS.Common.ActivatedDate" type="DateTime" reportable="dimension">
        <WHENNOTCHANGED field="System.State">
          <READONLY />
        </WHENNOTCHANGED>
      </FIELD>
      <FIELD name="Activated By" refname="Microsoft.VSTS.Common.ActivatedBy" type="String" reportable="dimension">
        <WHENNOTCHANGED field="System.State">
          <READONLY />
        </WHENNOTCHANGED>
      </FIELD>
      <FIELD name="Reason" refname="System.Reason" type="String" reportable="dimension">
        <HELPTEXT>The reason why the task is in the current state</HELPTEXT>
      </FIELD>
      <FIELD name="Assigned To" refname="System.AssignedTo" type="String" reportable="dimension">
        <ALLOWEXISTINGVALUE />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="[project]\Contributors" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="Work Item Type" refname="System.WorkItemType" type="String" reportable="dimension" />
      <FIELD name="Closed By" refname="Microsoft.VSTS.Common.ClosedBy" type="String" reportable="dimension">
        <WHENNOTCHANGED field="System.State">
          <READONLY />
        </WHENNOTCHANGED>
      </FIELD>
      <FIELD name="Closed Date" refname="Microsoft.VSTS.Common.ClosedDate" type="DateTime" reportable="dimension">
        <WHENNOTCHANGED field="System.State">
          <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Rank" refname="Microsoft.VSTS.Common.Rank" type="String" reportable="dimension">
      <HELPTEXT>Stack rank to prioritize work</HELPTEXT>
    </FIELD>
    <FIELD name="Created Date" refname="System.CreatedDate" type="DateTime" reportable="dimension" />
    <FIELD name="Created By" refname="System.CreatedBy" type="String" reportable="dimension" />
    <FIELD name="Exit Criteria" refname="Microsoft.VSTS.Common.ExitCriteria" type="String" reportable="dimension">
      <HELPTEXT>Flag to determine if this scenario should be tracked as an exit criteria for the iteration</HELPTEXT>
      <REQUIRED />
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="Yes" />
        <LISTITEM value="No" />
      </ALLOWEDVALUES>
      <DEFAULT from="value" value="No" />
    </FIELD>
    <FIELD name="Discipline" refname="Microsoft.VSTS.Common.Discipline" type="String" reportable="dimension">
      <HELPTEXT>The discipline to which the task belongs</HELPTEXT>
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="Development" />
        <LISTITEM value="Test" />
        <LISTITEM value="Project Management" />
        <LISTITEM value="Requirements" />
        <LISTITEM value="Architecture" />
        <LISTITEM value="Release Management" />
      </ALLOWEDVALUES>
    </FIELD>
    <FIELD name="Integration Build" refname="Microsoft.VSTS.Build.IntegrationBuild" type="String" reportable="dimension">
      <HELPTEXT>The build in which the task was completed</HELPTEXT>
      <SUGGESTEDVALUES expanditems="true">
```

```xml
        <LISTITEM value="&lt;None&gt;" />
      </SUGGESTEDVALUES>
    </FIELD>
    <FIELD name="Completed Work" refname="Microsoft.VSTS.Scheduling.CompletedWork" type="Double" reportable="measure" formula="sum">
      <HELPTEXT>The number of hours that have been completed for this task</HELPTEXT>
    </FIELD>
    <FIELD name="Start Date" refname="Microsoft.VSTS.Scheduling.StartDate" type="DateTime" reportable="dimension">
      <HELPTEXT>The date to start the task</HELPTEXT>
      <DEFAULT from="clock" />
    </FIELD>
    <FIELD name="Finish Date" refname="Microsoft.VSTS.Scheduling.FinishDate" type="DateTime" reportable="dimension">
      <HELPTEXT>The date to finish the task</HELPTEXT>
      <WHEN field="COMPANY.Task.GlobalPriority" value="2 Important">
        <REQUIRED for="[global]\Team Foundation Valid Users" />
      </WHEN>
    </FIELD>
    <FIELD name="Task Hierarchy" refname="Microsoft.VSTS.Scheduling.TaskHierarchy" type="String" reportable="dimension">
      <HELPTEXT>A string representing MS-Project context for the given task</HELPTEXT>
    </FIELD>
    <FIELD name="Baseline Work" refname="Microsoft.VSTS.Scheduling.BaselineWork" type="Double" reportable="measure" formula="sum">
      <HELPTEXT>The number of hours of work from the baseline plan</HELPTEXT>
    </FIELD>
    <FIELD name="Remaining Work" refname="Microsoft.VSTS.Scheduling.RemainingWork" type="Double" reportable="measure" formula="sum">
      <HELPTEXT>An estimate of the number of hours remaining to complete the task</HELPTEXT>
    </FIELD>
```

```xml
      <FIELD name="Description" refname="System.Description" type="PlainText" />
      <FIELD name="Project" refname="COMPANY.Project" type="String" reportable="dimension" />
      <FIELD name="History" refname="System.History" type="History">
        <HELPTEXT>Discussion thread and other historical information</HELPTEXT>
      </FIELD>
      <FIELD name="Instance" refname="COMPANY.Deployment.Instance" type="String">
        <HELPTEXT>COMPANY.Deployment.Instance</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <GLOBALLIST name="GlobalList_Test" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="RequestSource" refname="COMPANY.Deployment.RequestSource" type="String" reportable="detail">
        <HELPTEXT>COMPANY.Deployment.RequestSource</HELPTEXT>
      </FIELD>
      <FIELD name="Scope" refname="COMPANY.Deployment.DeploymentType" type="String">
        <HELPTEXT>COMPANY.Deployment.DeploymentType</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1 Content" />
          <LISTITEM value="2 Design/Template" />
          <LISTITEM value="3 Binary" />
          <LISTITEM value="4 Whole Portal" />
          <LISTITEM value="5 Whole Instance" />
          <LISTITEM value="6 Launch" />
          <LISTITEM value="7 New Website" />
        </ALLOWEDVALUES>
        <REQUIRED for="[global]\Team Foundation Valid Users" />
      </FIELD>
      <FIELD name="Data" refname="COMPANY.Deployment.DeploymentData" type="String">
        <HELPTEXT>COMPANY.Deployment.DeploymentData</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
```

```
          <LISTITEM value="1 Files" />
          <LISTITEM value="2 DB" />
          <LISTITEM value="3 Files &amp; DB" />
        </ALLOWEDVALUES>
        <REQUIRED for="[global]\Team Foundation Valid Users"
            />
      </FIELD>
      <FIELD name="DRequestor" refname="COMPANY.Deployment.
         Requestor" type="String">
        <HELPTEXT>COMPANY.Deployment.Requestor</HELPTEXT>
      </FIELD>
      <FIELD name="DestInstance" refname="COMPANY.Deployment
         .DestinationInstance" type="String">
        <HELPTEXT>COMPANY.Deployment.DestinationInstance</
           HELPTEXT>
        <DEFAULT from="field" field="COMPANY.Deployment.
           Instance" />
      </FIELD>
      <FIELD name="FromPortalID" refname="COMPANY.Deployment
         .SourcePortalId" type="Integer">
        <HELPTEXT>COMPANY.Deployment.SourcePortalId</
           HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1" />
          <LISTITEM value="2" />
          <LISTITEM value="3" />
          <LISTITEM value="0" />
          <LISTITEM value="4" />
          <LISTITEM value="5" />
          <LISTITEM value="6" />
          <LISTITEM value="7" />
          <LISTITEM value="8" />
          <LISTITEM value="9" />
          <LISTITEM value="10" />
          <LISTITEM value="11" />
          <LISTITEM value="12" />
          <LISTITEM value="13" />
          <LISTITEM value="14" />
          <LISTITEM value="15" />
          <LISTITEM value="16" />
          <LISTITEM value="17" />
          <LISTITEM value="18" />
          <LISTITEM value="19" />
```

```xml
          <LISTITEM value="20" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="ToPortalID" refname="COMPANY.Deployment.DestinationPortalId" type="Integer">
        <HELPTEXT>COMPANY.Deployment.DestinationPortalId</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1" />
          <LISTITEM value="2" />
          <LISTITEM value="3" />
          <LISTITEM value="0" />
          <LISTITEM value="4" />
          <LISTITEM value="5" />
          <LISTITEM value="6" />
          <LISTITEM value="7" />
          <LISTITEM value="8" />
          <LISTITEM value="9" />
          <LISTITEM value="10" />
          <LISTITEM value="11" />
          <LISTITEM value="12" />
          <LISTITEM value="13" />
          <LISTITEM value="14" />
          <LISTITEM value="15" />
          <LISTITEM value="16" />
          <LISTITEM value="17" />
          <LISTITEM value="18" />
          <LISTITEM value="19" />
          <LISTITEM value="20" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="SourceEnv" refname="COMPANY.Deployment.SourceEnvironment" type="String">
        <HELPTEXT>COMPANY.Deployment.SourceEnvironment</HELPTEXT>
        <REQUIRED for="[global]\Team Foundation Valid Users" />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1 DEV" />
          <LISTITEM value="2 STA" />
          <LISTITEM value="3 PRD" />
        </ALLOWEDVALUES>
        <WHEN field="COMPANY.Deployment.DeploymentType" value="7 New Website">
          <FROZEN for="[global]\Team Foundation Valid Users" />
        </WHEN>
      </FIELD>
      <FIELD name="DestEnv" refname="COMPANY.Deployment.DestinationEnvironment" type="String">
        <HELPTEXT>COMPANY.Deployment.DestinationEnvironment</HELPTEXT>
        <REQUIRED for="[global]\Team Foundation Valid Users" />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1 DEV" />
          <LISTITEM value="2 STA" />
          <LISTITEM value="3 PRD" />
        </ALLOWEDVALUES>
        <WHEN field="COMPANY.Deployment.DeploymentType" value="7 New Website">
          <FROZEN for="[global]\Team Foundation Valid Users" />
        </WHEN>
      </FIELD>
      <FIELD name="Website" refname="COMPANY.Deployment.Website" type="String" reportable="dimension">
        <HELPTEXT>Specifies the Website</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <GLOBALLIST name="VEC_Websites_Instances" />
        </ALLOWEDVALUES>
        <REQUIRED for="[global]\Team Foundation Valid Users" />
      </FIELD>
      <FIELD name="Requestor" refname="COMPANY.Task.RequestSource" type="String" reportable="dimension">
        <HELPTEXT>COMPANY.Task.RequestSource</HELPTEXT>
        <ALLOWEXISTINGVALUE />
        <DEFAULT from="currentuser" />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="[project]\Contributors" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="Prio" refname="COMPANY.Task.
```

```
GlobalPriority" type="String" reportable="dimension">
      <HELPTEXT>COMPANY.Task.GlobalPriority</HELPTEXT>
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="1 Normal" />
        <LISTITEM value="2 Important" />
        <LISTITEM value="3 Emergency" />
        <LISTITEM value="0 Low" />
      </ALLOWEDVALUES>
      <DEFAULT from="value" value="1 Normal" />
      <REQUIRED for="[global]\Team Foundation Valid Users" />
    </FIELD>
    <FIELD name="RelatedLinkCount" refname="System.RelatedLinkCount" type="Integer" />
    <FIELD name="Project Code" refname="COMPANY.Task.ProjectCode" type="String" reportable="dimension">
      <HELPTEXT>COMPANY.Task.ProjectCode</HELPTEXT>
      <ALLOWEDVALUES expanditems="true">
        <GLOBALLIST name="VEC_Projects" />
      </ALLOWEDVALUES>
    </FIELD>
    <FIELD name="Phase" refname="COMPANY.Task.Phase" type="String" reportable="detail">
      <HELPTEXT>The Project Phase the task falls into (e.g. Maintenance, Project)</HELPTEXT>
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="Project" />
        <LISTITEM value="Maintenance" />
      </ALLOWEDVALUES>
    </FIELD>
    <FIELD name="StartConstraint" refname="COMPANY.Task.StartConstraint" type="String" reportable="detail">
      <HELPTEXT>Constraint for starting a work item</HELPTEXT>
    </FIELD>
    <FIELD name="Task Owner" refname="COMPANY.Task.Owner" type="String" reportable="detail">
      <HELPTEXT>Person responsilbe for the Work Item</HELPTEXT>
      <ALLOWEXISTINGVALUE />
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="[project]\Contributors" />
```

```
          </ALLOWEDVALUES>
          <DEFAULT from="field" field="System.CreatedBy" />
        </FIELD>
        <FIELD name="DescriptionHTML" refname="COMPANY.Task.DescriptionHtml" type="HTML" />
        <FIELD name="LaunchRelevancy" refname="COMPANY.Task.LaunchRelevancy" type="String" reportable="dimension">
          <HELPTEXT>Is this WI launch relevant?</HELPTEXT>
          <ALLOWEDVALUES expanditems="true">
            <LISTITEM value="Yes" />
            <LISTITEM value="No" />
          </ALLOWEDVALUES>
          <REQUIRED for="[global]\Team Foundation Valid Users" />
          <DEFAULT from="value" value="-" />
        </FIELD>
        <FIELD name="Group" refname="COMPANY.Task.Group" type="String" reportable="dimension">
          <HELPTEXT>To which group does the task belong to (e.g. SMA, Core, ...)</HELPTEXT>
          <ALLOWEDVALUES expanditems="true">
            <LISTITEM value="SMA" />
            <LISTITEM value="Core" />
          </ALLOWEDVALUES>
          <DEFAULT from="value" value="Core" />
        </FIELD>
        <FIELD name="Iteration Path" refname="System.IterationPath" type="TreePath" reportable="dimension">
          <HELPTEXT>The iteration of the product with which this task is associated</HELPTEXT>
        </FIELD>
        <FIELD name="IterationID" refname="System.IterationId" type="Integer" />
        <FIELD name="ExternalLinkCount" refname="System.ExternalLinkCount" type="Integer" />
        <FIELD name="Team Project" refname="System.TeamProject" type="String" reportable="dimension" />
        <FIELD name="HyperLinkCount" refname="System.HyperLinkCount" type="Integer" />
        <FIELD name="AttachedFileCount" refname="System.AttachedFileCount" type="Integer" />
```

```xml
      <FIELD name="Node Name" refname="System.NodeName" type="String" />
      <FIELD name="Area Path" refname="System.AreaPath" type="TreePath" reportable="dimension">
        <HELPTEXT>The area of the product with which this task is associated</HELPTEXT>
      </FIELD>
      <FIELD name="Revised Date" refname="System.RevisedDate" type="DateTime" />
      <FIELD name="Changed Date" refname="System.ChangedDate" type="DateTime" reportable="dimension" />
      <FIELD name="ID" refname="System.Id" type="Integer" reportable="dimension" />
      <FIELD name="AreaID" refname="System.AreaId" type="Integer" />
      <FIELD name="Authorized As" refname="System.AuthorizedAs" type="String" />
    </FIELDS>
    <WORKFLOW>
      <STATES>
        <STATE value="Open">
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate">
              <EMPTY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <EMPTY />
            </FIELD>
          </FIELDS>
        </STATE>
        <STATE value="Completed" />
        <STATE value="In Progress" />
        <STATE value="Executed" />
        <STATE value="Cancelled" />
        <STATE value="Tested Error" />
        <STATE value="Not Tested" />
        <STATE value="Tested OK" />
        <STATE value="On Hold" />
      </STATES>
      <TRANSITIONS>
        <TRANSITION from="" to="Open">
          <REASONS>
```

```
              <DEFAULTREASON value="New" />
            </REASONS>
            <FIELDS>
              <FIELD refname="Microsoft.VSTS.Common.
                 ActivatedBy">
                <COPY from="currentuser" />
                <VALIDUSER />
                <REQUIRED />
              </FIELD>
              <FIELD refname="Microsoft.VSTS.Common.
                 ActivatedDate">
                <SERVERDEFAULT from="clock" />
              </FIELD>
              <FIELD refname="COMPANY.Deployment.Requestor">
                <DEFAULT from="value" value="e.g. Project
                   Owner" />
              </FIELD>
              <FIELD refname="COMPANY.Deployment.RequestSource
                 ">
                <DEFAULT from="value" value="e.g. Gemini
                   Request Number" />
              </FIELD>
            </FIELDS>
          </TRANSITION>
          <TRANSITION from="Executed" to="Cancelled">
            <REASONS>
              <DEFAULTREASON value="Deferred" />
            </REASONS>
          </TRANSITION>
          <TRANSITION from="Completed" to="Tested Error">
            <REASONS>
              <DEFAULTREASON value="Error found" />
            </REASONS>
          </TRANSITION>
          <TRANSITION from="Tested Error" to="Executed">
            <REASONS>
              <DEFAULTREASON value="Error fixed" />
            </REASONS>
          </TRANSITION>
          <TRANSITION from="Executed" to="Tested Error" for="[
             global]\VEC ProQ Users">
            <REASONS>
```

```xml
            <DEFAULTREASON value="ProQ review not successful" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedBy">
              <SERVERDEFAULT from="currentuser" />
            </FIELD>
          </FIELDS>
        </TRANSITION>
        <TRANSITION from="Executed" to="Not Tested" for="[global]\VEC ProQ Users">
          <REASONS>
            <DEFAULTREASON value="Testing not necessary" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Executed" to="Tested OK" for="[global]\VEC ProQ Users">
          <REASONS>
            <DEFAULTREASON value="ProQ review successful" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Tested OK" to="Completed">
          <REASONS>
            <DEFAULTREASON value="Completed and tested" />
            <REASON value="Deferred" />
            <REASON value="Obsolete" />
            <REASON value="Cut" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedBy">
              <READONLY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedDate">
              <READONLY />
            </FIELD>
          </FIELDS>
          <ACTIONS>
            <ACTION value="Microsoft.VSTS.Actions.Checkin" />
          </ACTIONS>
        </TRANSITION>
        <TRANSITION from="Tested Error" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Tested OK" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Not Tested" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Open" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Not Tested" to="Completed">
          <REASONS>
            <DEFAULTREASON value="Completed without testing" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Cancelled" to="Open">
          <REASONS>
            <DEFAULTREASON value="Opened up again" />
```

```xml
          </REASONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="Executed">
          <REASONS>
            <DEFAULTREASON value="Deployment Performed" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="System.AssignedTo">
              <COPY from="field" field="Microsoft.VSTS.Common.ClosedBy" />
            </FIELD>
          </FIELDS>
        </TRANSITION>
        <TRANSITION from="Open" to="In Progress">
          <REASONS>
            <DEFAULTREASON value="Work started" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Open" to="On Hold">
          <REASONS>
            <DEFAULTREASON value="See History" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="On Hold">
          <REASONS>
            <DEFAULTREASON value="See History" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="On Hold" to="In Progress">
          <REASONS>
            <DEFAULTREASON value="Opened up again" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="On Hold" to="Cancelled">
          <REASONS>
            <DEFAULTREASON value="Deferred" />
          </REASONS>
        </TRANSITION>
      </TRANSITIONS>
    </WORKFLOW>
    <FORM>
      <Layout>
        <Group>
          <Column PercentWidth="100">
            <Control FieldName="System.Title" Type="
               FieldControl" Label="Title:" LabelPosition="
               Left" />
          </Column>
        </Group>
        <Group Label="General Information">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Deployment.
               DeploymentType" Type="FieldControl" Label="
               Deployment Scope:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Deployment.
               DeploymentData" Type="FieldControl" Label="
               Data Type:" LabelPosition="Left" />
            <Control FieldName="Microsoft.VSTS.Scheduling.
               StartDate" Type="DateTimeControl" Label="
               Start not before:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.StartConstraint
               " Type="FieldControl" Label="Start
               Constraint:" LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Deployment.
               RequestSource" Type="FieldControl" Label="
               External Reference:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.GlobalPriority"
                Type="FieldControl" Label="Priority:"
               LabelPosition="Left" />
```

```xml
            <Control FieldName="Microsoft.VSTS.Scheduling.FinishDate" Type="DateTimeControl" Label="Timeline:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.LaunchRelevancy" Type="FieldControl" Label="Is Launch relevant:" LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="Website">
          <Column PercentWidth="90">
            <Control FieldName="COMPANY.Deployment.Website" Type="FieldControl" Label="Website:" LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="Project Information">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.ProjectCode" Type="FieldControl" Label="Project Code:" LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.Phase" Type="FieldControl" Label="Phase:" LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="Persons &amp; Status">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.RequestSource" Type="FieldControl" Label="Requestor:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.Owner" Type="FieldControl" Label="Task Owner:" LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.Group" Type="FieldControl" Label="Assigned Group:" LabelPosition="Left" />
            <Control FieldName="System.AssignedTo" Type="FieldControl" Label="Assigned To:" LabelPosition="Left" />
          <Control FieldName="System.State" Type="FieldControl" Label="&amp;State:" LabelPosition="Left" />
        </Column>
      </Group>
      <Group Label="Environment">
        <Column PercentWidth="50">
          <Control FieldName="COMPANY.Deployment.SourceEnvironment" Type="FieldControl" Label="Source Environment:" LabelPosition="Left" />
        </Column>
        <Column PercentWidth="50">
          <Control FieldName="COMPANY.Deployment.DestinationEnvironment" Type="FieldControl" Label="Destination Environment:" LabelPosition="Left" />
        </Column>
      </Group>
      <Group Label="Description">
        <Column PercentWidth="50">
          <Control FieldName="COMPANY.Task.DescriptionHtml" Type="HtmlFieldControl" Label="Des&amp;cription:" LabelPosition="Top" Dock="Top" />
        </Column>
        <Column PercentWidth="50">
          <Control FieldName="System.History" Type="WorkItemLogControl" Label="&amp;Comment:" LabelPosition="Top" Dock="Top" />
        </Column>
      </Group>
      <TabGroup>
        <Tab Label="File Attachments">
          <Control Type="AttachmentsControl" LabelPosition="Top" Dock="Fill" />
        </Tab>
        <Tab Label="Links">
          <Control Type="LinksControl" Label="" LabelPosition="Top" />
        </Tab>
      </TabGroup>
    </Layout>
  </FORM>
</WORKITEMTYPE>
```

```
</WITD>
```

Listing A.2: XML Structure - Work Item Type “Deployment Request”

```
<WITD application="Work item type editor" version="1.0" xmlns:witd="http://schemas.microsoft.com/VisualStudio/2005/workitemtracking/typedef">
  <WORKITEMTYPE name="Personal Todo">
    <DESCRIPTION>Work Item to manage your personal todo list </DESCRIPTION>
    <FIELDS>
      <FIELD name="Title" refname="System.Title" type="String" reportable="dimension">
        <HELPTEXT>Short description of the task used to differentiate it in a list or report</HELPTEXT>
        <REQUIRED />
      </FIELD>
      <FIELD name="State" refname="System.State" type="String" reportable="dimension">
        <HELPTEXT>The workflow state of the task</HELPTEXT>
      </FIELD>
      <FIELD name="Rev" refname="System.Rev" type="Integer" reportable="dimension" />
      <FIELD name="Changed By" refname="System.ChangedBy" type="String" reportable="dimension">
        <VALIDUSER />
      </FIELD>
      <FIELD name="Issue" refname="Microsoft.VSTS.Common.Issue" type="String" reportable="dimension">
        <HELPTEXT>Used to highlight a task, e.g., to mark it as an issue</HELPTEXT>
        <REQUIRED />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Yes" />
          <LISTITEM value="No" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="No" />
      </FIELD>
      <FIELD name="State Change Date" refname="Microsoft.VSTS.Common.StateChangeDate" type="DateTime">
        <WHENCHANGED field="System.State">
          <SERVERDEFAULT from="clock" />
        </WHENCHANGED>
        <WHENNOTCHANGED field="System.State">
        <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Activated Date" refname="Microsoft.VSTS.Common.ActivatedDate" type="DateTime" reportable="dimension">
      <WHENNOTCHANGED field="System.State">
        <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Activated By" refname="Microsoft.VSTS.Common.ActivatedBy" type="String" reportable="dimension">
      <WHENNOTCHANGED field="System.State">
        <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Reason" refname="System.Reason" type="String" reportable="dimension">
      <HELPTEXT>The reason why the task is in the current state</HELPTEXT>
    </FIELD>
    <FIELD name="Assigned To" refname="System.AssignedTo" type="String" reportable="dimension">
      <ALLOWEXISTINGVALUE />
      <ALLOWEDVALUES expanditems="true">
        <LISTITEM value="[project]\Contributors" />
      </ALLOWEDVALUES>
    </FIELD>
    <FIELD name="Work Item Type" refname="System.WorkItemType" type="String" reportable="dimension" />
    <FIELD name="Closed By" refname="Microsoft.VSTS.Common.ClosedBy" type="String" reportable="dimension">
      <WHENNOTCHANGED field="System.State">
        <READONLY />
      </WHENNOTCHANGED>
    </FIELD>
    <FIELD name="Closed Date" refname="Microsoft.VSTS.Common.ClosedDate" type="DateTime" reportable="dimension">
      <WHENNOTCHANGED field="System.State">
        <READONLY />
        </WHENNOTCHANGED>
      </FIELD>
      <FIELD name="Rank" refname="Microsoft.VSTS.Common.Rank
         " type="String" reportable="dimension">
        <HELPTEXT>Stack rank to prioritize work</HELPTEXT>
      </FIELD>
      <FIELD name="Created Date" refname="System.CreatedDate
         " type="DateTime" reportable="dimension" />
      <FIELD name="Created By" refname="System.CreatedBy"
         type="String" reportable="dimension" />
      <FIELD name="Exit Criteria" refname="Microsoft.VSTS.
         Common.ExitCriteria" type="String" reportable="
         dimension">
        <HELPTEXT>Flag to determine if this scenario should
           be tracked as an exit criteria for the iteration<
           /HELPTEXT>
        <REQUIRED />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Yes" />
          <LISTITEM value="No" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="No" />
      </FIELD>
      <FIELD name="Integration Build" refname="Microsoft.
         VSTS.Build.IntegrationBuild" type="String"
         reportable="dimension">
        <HELPTEXT>The build in which the task was completed<
           /HELPTEXT>
        <SUGGESTEDVALUES expanditems="true">
          <LISTITEM value="&lt;None&gt;" />
        </SUGGESTEDVALUES>
        <SUGGESTEDVALUES expanditems="true" filteritems="
           excludegroups">
          <GLOBALLIST name="Builds - DNN.Template.4.9.1" />
        </SUGGESTEDVALUES>
      </FIELD>
      <FIELD name="Remaining Work" refname="Microsoft.VSTS.
         Scheduling.RemainingWork" type="Double" reportable=
         "measure" formula="sum">
        <HELPTEXT>An estimate of the number of hours
           remaining to complete the task</HELPTEXT>
      </FIELD>
      <FIELD name="Completed Work" refname="Microsoft.VSTS.
```

```
          Scheduling.CompletedWork" type="Double" reportable=
          "measure" formula="sum">
        <HELPTEXT>The number of hours that have been
           completed for this task</HELPTEXT>
      </FIELD>
      <FIELD name="Start Date" refname="Microsoft.VSTS.
          Scheduling.StartDate" type="DateTime" reportable="
          dimension">
        <HELPTEXT>The date to start the task</HELPTEXT>
      </FIELD>
      <FIELD name="Finish Date" refname="Microsoft.VSTS.
          Scheduling.FinishDate" type="DateTime" reportable="
          dimension">
        <HELPTEXT>The date to finish the task</HELPTEXT>
      </FIELD>
      <FIELD name="Task Hierarchy" refname="Microsoft.VSTS.
          Scheduling.TaskHierarchy" type="String" reportable=
          "dimension">
        <HELPTEXT>A string representing MS-Project context
           for the given task</HELPTEXT>
      </FIELD>
      <FIELD name="Discipline" refname="Microsoft.VSTS.
          Common.Discipline" type="String" reportable="
          dimension">
        <HELPTEXT>The discipline to which the task belongs</
           HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Development" />
          <LISTITEM value="Test" />
          <LISTITEM value="Deployment" />
          <LISTITEM value="Launch Management" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="Baseline Work" refname="Microsoft.VSTS.
          Scheduling.BaselineWork" type="Double" reportable="
          measure" formula="sum">
        <HELPTEXT>The number of hours of work from the
           baseline plan</HELPTEXT>
      </FIELD>
      <FIELD name="Description" refname="System.Description"
           type="PlainText" />
      <FIELD name="Project" refname="COMPANY.Project" type="
          String" reportable="dimension">
```

```xml
        <HELPTEXT>Project</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="SMA" />
          <LISTITEM value="WeaningPlanner" />
          <LISTITEM value="GrowthChart" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="History" refname="System.History" type="
         History">
        <HELPTEXT>Discussion thread and other historical
           information</HELPTEXT>
      </FIELD>
      <FIELD name="Module" refname="COMPANY.Module" type="
         String">
        <HELPTEXT>COMPANY.Module</HELPTEXT>
      </FIELD>
      <FIELD name="PlanningState" refname="COMPANY.
         PlanningState" type="String">
        <HELPTEXT>PlanningState</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Unplanned" />
          <LISTITEM value="Planned" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="Website" refname="COMPANY.Deployment.
         Website" type="String" reportable="dimension">
        <HELPTEXT>The website concerned (including URLs,
           PortalID, ...)</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <GLOBALLIST name="VEC_Websites_Instances" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="DevTimeline" refname="COMPANY.Task.
         DevTimeline" type="DateTime">
        <HELPTEXT>Timeline set by executing person</HELPTEXT
           >
      </FIELD>
      <FIELD name="Completion" refname="COMPANY.Task.
         Completion" type="Integer">
        <HELPTEXT>Completion Percentage</HELPTEXT>
      </FIELD>
      <FIELD name="Requestor" refname="COMPANY.Task.
```

```
         RequestSource" type="String" reportable="dimension"
         >
        <HELPTEXT>Who requested this task?</HELPTEXT>
        <ALLOWEXISTINGVALUE />
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="[project]\Contributors" />
        </ALLOWEDVALUES>
        <DEFAULT from="currentuser" />
      </FIELD>
      <FIELD name="Prio" refname="COMPANY.Task.
         GlobalPriority" type="String" reportable="dimension
         ">
        <HELPTEXT>Global Task Importance</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1 Normal" />
          <LISTITEM value="2 Important" />
          <LISTITEM value="3 Emergency" />
          <LISTITEM value="0 Low" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="1 Normal" />
      </FIELD>
      <FIELD name="Task Type" refname="COMPANY.Task.TaskType
         " type="String" reportable="dimension">
        <HELPTEXT>Defines the type (Bug, Feature Request
           ,...) of a task</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Bug" />
          <LISTITEM value="Test" />
          <LISTITEM value="Feature Request" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="RelatedLinkCount" refname="System.
         RelatedLinkCount" type="Integer" />
      <FIELD name="Bug Severity" refname="COMPANY.Task.
         Severity" type="String" reportable="dimension">
        <HELPTEXT>Severity of a bug</HELPTEXT>
        <SUGGESTEDVALUES expanditems="true">
          <LISTITEM value="1 - Cosmetic" />
          <LISTITEM value="2 - Functionality affected" />
          <LISTITEM value="3 - Showstopper" />
        </SUGGESTEDVALUES>
        <WHENNOT field="COMPANY.Task.TaskType" value="Bug">
```

```
          <READONLY for="[global]\Team Foundation Valid
             Users" />
        </WHENNOT>
        <WHEN field="COMPANY.Task.TaskType" value="Bug">
          <REQUIRED for="[global]\Team Foundation Valid
             Users" />
        </WHEN>
      </FIELD>
      <FIELD name="Estimated Effort" refname="COMPANY.Task.
         EstimatedEffort" type="Double" reportable="
         dimension">
        <HELPTEXT>The estimated effort in days</HELPTEXT>
      </FIELD>
      <FIELD name="Phase" refname="COMPANY.Task.Phase" type=
         "String" reportable="detail">
        <HELPTEXT>Phase in which the Task falls into (e.g.
           Maintenance, Project)</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="Project" />
          <LISTITEM value="Maintenance" />
        </ALLOWEDVALUES>
      </FIELD>
      <FIELD name="StartConstraint" refname="COMPANY.Task.
         StartConstraint" type="String" reportable="detail">
        <HELPTEXT>Constraint for Start Date of a WorkItem</
           HELPTEXT>
      </FIELD>
      <FIELD name="Project Code" refname="COMPANY.Task.
         ProjectCode" type="String" reportable="dimension">
        <HELPTEXT>The VEC project code</HELPTEXT>
      </FIELD>
      <FIELD name="Context" refname="COMPANY.Task.Context"
         type="String" reportable="detail">
        <HELPTEXT>Context of the Work Item (People,
           Environment, ...)</HELPTEXT>
      </FIELD>
      <FIELD name="DescriptionHTML" refname="COMPANY.Task.
         DescriptionHtml" type="HTML" />
      <FIELD name="Iteration Path" refname="System.
         IterationPath" type="TreePath" reportable="
         dimension">
        <HELPTEXT>The iteration of the product with which
           this task is associated</HELPTEXT>
      </FIELD>
      <FIELD name="IterationID" refname="System.IterationId"
          type="Integer" />
      <FIELD name="ExternalLinkCount" refname="System.
         ExternalLinkCount" type="Integer" />
      <FIELD name="Team Project" refname="System.TeamProject
         " type="String" reportable="dimension" />
      <FIELD name="HyperLinkCount" refname="System.
         HyperLinkCount" type="Integer" />
      <FIELD name="AttachedFileCount" refname="System.
         AttachedFileCount" type="Integer" />
      <FIELD name="Node Name" refname="System.NodeName" type
         ="String" />
      <FIELD name="Area Path" refname="System.AreaPath" type
         ="TreePath" reportable="dimension">
        <HELPTEXT>The area of the product with which this
           task is associated</HELPTEXT>
      </FIELD>
      <FIELD name="Revised Date" refname="System.RevisedDate
         " type="DateTime" />
      <FIELD name="Changed Date" refname="System.ChangedDate
         " type="DateTime" reportable="dimension" />
      <FIELD name="ID" refname="System.Id" type="Integer"
         reportable="dimension" />
      <FIELD name="AreaID" refname="System.AreaId" type="
         Integer" />
      <FIELD name="Authorized As" refname="System.
         AuthorizedAs" type="String" />
    </FIELDS>
    <WORKFLOW>
      <STATES>
        <STATE value="Open">
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate
               ">
              <EMPTY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <EMPTY />
            </FIELD>
          </FIELDS>
        </STATE>
        <STATE value="Closed" />
        <STATE value="In Progress" />
      </STATES>
      <TRANSITIONS>
        <TRANSITION from="In Progress" to="Closed">
          <REASONS>
            <DEFAULTREASON value="Completed" />
            <REASON value="Deferred" />
            <REASON value="Obsolete" />
            <REASON value="Cut" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate
               ">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedBy">
              <READONLY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedDate">
              <READONLY />
            </FIELD>
          </FIELDS>
          <ACTIONS>
            <ACTION value="Microsoft.VSTS.Actions.Checkin" /
               >
          </ACTIONS>
        </TRANSITION>
        <TRANSITION from="" to="Open">
          <REASONS>
            <DEFAULTREASON value="New" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
```

```xml
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="System.AssignedTo">
              <DEFAULT from="currentuser" />
            </FIELD>
          </FIELDS>
        </TRANSITION>
        <TRANSITION from="Closed" to="Open">
          <REASONS>
            <DEFAULTREASON value="Reactivated" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="System.AssignedTo">
              <COPY from="field" field="Microsoft.VSTS.Common.ClosedBy" />
            </FIELD>
          </FIELDS>
        </TRANSITION>
        <TRANSITION from="Open" to="In Progress">
          <REASONS>
            <DEFAULTREASON value="Work started" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="In Progress" to="Open">
          <REASONS>
            <DEFAULTREASON value="Other Task Opened" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Open" to="Closed">
```

```
          <REASONS>
            <DEFAULTREASON value="Deferred" />
            <REASON value="Obsolete" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ClosedDate
               ">
              <SERVERDEFAULT from="clock" />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ClosedBy">
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedBy">
              <READONLY />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.
               ActivatedDate">
              <READONLY />
            </FIELD>
          </FIELDS>
          <ACTIONS>
            <ACTION value="Microsoft.VSTS.Actions.Checkin" /
               >
          </ACTIONS>
        </TRANSITION>
      </TRANSITIONS>
    </WORKFLOW>
    <FORM>
      <Layout>
        <Group Label=" ">
          <Column PercentWidth="100">
            <Control FieldName="System.Title" Type="
               FieldControl" Label="&amp;Title:"
               LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="General Information">
          <Column PercentWidth="50">
            <Control FieldName="Microsoft.VSTS.Scheduling.
```

```
               FinishDate" Type="DateTimeControl" Label="
               Timeline:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.GlobalPriority"
                Type="FieldControl" Label="Priority:"
               LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="Microsoft.VSTS.Scheduling.
               StartDate" Type="DateTimeControl" Label="
               Start not before:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.StartConstraint
               " Type="FieldControl" Label="Start
               Constraint:" LabelPosition="Left" />
          </Column>
        </Group>
        <Group Label="Project Information">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.ProjectCode"
               Type="FieldControl" Label="Project Code:"
               LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.Phase" Type="
               FieldControl" Label="Phase:" LabelPosition="
               Left" />
          </Column>
        </Group>
        <Group Label="State Information">
          <Column PercentWidth="50">
            <Control FieldName="COMPANY.Task.EstimatedEffort
               " Type="FieldControl" Label="Estimated Effort
                (days):" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.DevTimeline"
               Type="DateTimeControl" Label="Estimated
               Timeline:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.Completion"
               Type="FieldControl" Label="Percentage
               complete:" LabelPosition="Left" />
          </Column>
          <Column PercentWidth="50">
            <Control FieldName="System.State" Type="
               FieldControl" Label="&amp;State:"
               LabelPosition="Left" />
```

```xml
            <Control FieldName="System.Reason" Type="FieldControl" Label="&amp;Reason:" LabelPosition="Left" />
            <Control FieldName="COMPANY.Task.Context" Type="FieldControl" Label="Context:" LabelPosition="Left" />
          </Column>
        </Group>
        <Group>
          <Column PercentWidth="100">
            <Control FieldName="COMPANY.Task.DescriptionHtml" Type="HtmlFieldControl" Label="Des&amp;cription:" LabelPosition="Top" Dock="Fill" />
          </Column>
        </Group>
        <TabGroup>
          <Tab Label="History">
            <Control FieldName="System.History" Type="WorkItemLogControl" Label="Comment:" LabelPosition="Top" Dock="Fill" />
          </Tab>
          <Tab Label="File Attachments">
            <Control Type="AttachmentsControl" LabelPosition="Top" />
          </Tab>
          <Tab Label="Links">
            <Control Type="LinksControl" LabelPosition="Top" />
          </Tab>
        </TabGroup>
      </Layout>
    </FORM>
  </WORKITEMTYPE>
</WITD>
```

Listing A.3: XML Structure - Work Item Type “Personal Todo”

Appendix B

TFS Queries and Alerts

B.1 List of Queries

- Deployments - All - Completed
- Deployments - All - Executed
- Deployments - All - Open
- Deployments - All - Tested
- Deployments - Dev - Open
- Deployments - ProS - Open
- Deployments - SMA - Open
- My Personal Todos
- My Received WorkItems
- My Requests
- Tasks - Assigned
- Tasks - Closed
- Tasks - DEV - BIP
- Tasks - DEV - Overdue
- Tasks - DEV - Unassigned
- Tasks - Executed
- Tasks - In Progress
- Tasks - Media - Open

- Tasks - Open
- Tasks - Ops - Closed
- Tasks - Ops - Open
- Tasks - ProQ - Open
- Tasks - ProQ- Overdue
- Tasks - ProS - Closed
- Tasks - ProS - Open
- Tasks - SMA - Media - Open
- Tasks - Unassigned

B.2 Code Examples

```
<?xml version="1.0" encoding="utf-8"?>
<WorkItemQuery Version="1">
   <TeamFoundationServer>SERVERURL/</TeamFoundationServer
      >
   <TeamProject>VECTasks</TeamProject>
   <Wiql>SELECT [COMPANY.Task.ProjectCode], [System.Title
      ], [COMPANY.Task.GlobalPriority], [System.
      CreatedDate], [Microsoft.VSTS.Scheduling.FinishDate
      ], [COMPANY.Task.TaskType], [COMPANY.Task.
      RequestSource], [System.State], [System.Id] FROM
      WorkItems WHERE [System.TeamProject] = 'VECTasks'
      AND [System.WorkItemType] = 'Task' AND [System.
      State] &lt;&gt; 'Closed' AND [System.State] &lt;&gt
      ; 'On Hold' AND [System.State] &lt;&gt; 'Cancelled'
       AND ( [System.AssignedTo] = 'VEC Development Users
      ' OR [System.AssignedTo] = 'FIRSTNAME LASTNAME' OR
      [System.AssignedTo] = 'FIRSTNAME2 LASTNAME2' ) AND
      [System.State] &lt;&gt; 'Completed' ORDER BY [
      System.CreatedDate] desc, [Microsoft.VSTS.
      Scheduling.FinishDate] desc, [COMPANY.Task.
      GlobalPriority] desc
      </Wiql>
   </WorkItemQuery>
```

Listing B.1: XPath - Query for unassigned Tasks

```
"CoreFields/StringFields/Field[ReferenceName='System.
   AssignedTo']/NewValue" = 'FIRSTNAME LASTNAME' AND
"PortfolioProject" = 'VECTasks' AND
"CoreFields/StringFields/Field[ReferenceName='System.
   CreatedBy']/NewValue" <> 'FIRSTNAME LASTNAME' AND "
   CoreFields/StringFields/Field[ReferenceName='System.
   ChangedBy']/NewValue" <> 'FIRSTNAME LASTNAME'
```

Listing B.2: XPath Alert-Query for assigned Work Items Request”

```
("CoreFields/StringFields/Field[ReferenceName='System.
   CreatedBy']/NewValue" = 'FIRSTNAME LASTNAME' OR ("
   ChangedFields/StringFields/Field[ReferenceName='COMPANY.
   Task.Owner']/OldValue" <> 'FIRSTNAME LASTNAME' AND "
   ChangedFields/StringFields/Field[ReferenceName='COMPANY.
   Task.Owner']/NewValue" = 'FIRSTNAME LASTNAME')  OR ("
   ChangedFields/StringFields/Field[ReferenceName='COMPANY.
   Task.RequestSource']/OldValue" <> 'FIRSTNAME LASTNAME'
   AND "ChangedFields/StringFields/Field[ReferenceName='
   COMPANY.Task.RequestSource']/NewValue" = 'FIRSTNAME
   LASTNAME') ) AND
"PortfolioProject" = 'VECTasks' AND
"CoreFields/StringFields/Field[ReferenceName='System.
   WorkItemType']/NewValue" <> 'Personal Todo' AND "
   CoreFields/StringFields/Field[ReferenceName='System.
   ChangedBy']/NewValue" <> 'FIRSTNAME LASTNAME'
```

Listing B.3: XPath - Alert-Query for created Work Items

B.3 Deadline Reminder

```
 using  System;
using System.Collections.Generic;
using System.Text;
using Microsoft.TeamFoundation.Client;
using Microsoft.TeamFoundation.WorkItemTracking.Client;
using log4net;

namespace WorkItemReminder
{
    internal class WorkItemReminder
    {

        private static log4net.ILog log;

        public WorkItemReminder()
        {
            try
            {
                log.Info("Start");
                WorkItemStore service =
                   TeamFoundationServerFactory.GetServer(
                   Settings.Default.ServerUrl).GetService(
                   typeof(WorkItemStore)) as WorkItemStore;
                foreach (WorkItem item in service.Query(
                   Settings.Default.Query))
                {
                    log.InfoFormat("Found work item # {0}",
                       item.Id);
                    item.Open();
                    item.History = item.History + string.
                       Format(Settings.Default.ResponseText,
                        item["Finish Date"].ToString());
                    item.Save();
                }
                log.Info("End");
            }
            catch (Exception exception)
            {
                log.Error(exception.Message, exception);
                System.Console.WriteLine(exception.Message);
            }
        }

        static WorkItemReminder()
        {
            log = log4net.LogManager.GetLogger(typeof(
               WorkItemReminder));
        }

        public static void Main(string[] args)
        {
            new WorkItemReminder();
        }
    }
}
```

Listing B.4: Alert Reminder - Class WorkItemReminder

```
 using  System;
using System.Collections.Generic;
using System.Text;
using System.Configuration;
using System.Diagnostics;

namespace WorkItemReminder
{
    internal sealed class Settings : ApplicationSettingsBase
    {
        // Fields
        private static Settings defaultInstance = ((Settings
           )SettingsBase.Synchronized(new Settings()));

        // Properties
        public static Settings Default
        {
            get
            {
                return defaultInstance;
            }
        }

        [ApplicationScopedSetting, DebuggerNonUserCode,
           DefaultSettingValue("select id from WorkItems
           where [System.TeamProject] = 'VECTasks' and [
           System.WorkItemType] = 'Task' and [Finish Date] =
            @today and State <> 'Closed' and State <> '
           Executed'")]
        public string Query
        {
            get
            {
                return (string)this["Query"];
            }
        }

        [ApplicationScopedSetting, DebuggerNonUserCode,
           DefaultSettingValue("The work item finish date
           expires at {0}")]
        public string ResponseText
        {
            get
            {
                return (string)this["ResponseText"];
            }
        }

        [DebuggerNonUserCode, ApplicationScopedSetting,
           DefaultSettingValue("SERVERURL")]
        public string ServerUrl
        {
            get
            {
                return (string)this["ServerUrl"];
            }
        }
    }

}
```

Listing B.5: Alert Reminder - Class Settings

```
   <?xml version="1.0"?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings" type="System.
       Configuration.ApplicationSettingsGroup, System,
       Version=2.0.0.0, Culture=neutral, PublicKeyToken=
       b77a5c561934e089">
      <section name="WorkItemReminder.Settings" type="System
         .Configuration.ClientSettingsSection, System,
         Version=2.0.0.0, Culture=neutral, PublicKeyToken=
         b77a5c561934e089" requirePermission="false"/>
    </sectionGroup>
  </configSections>
  <applicationSettings>
    <WorkItemReminder.Settings>
      <setting name="ServerUrl" serializeAs="String">
        <value>SERVERURL</value>
      </setting>
      <setting name="Query" serializeAs="String">
        <value>select id from WorkItems where [System.
           TeamProject] = 'VECTasks' and [System.
           WorkItemType] = 'Task' and [Finish Date] = @today
            and State &lt;&gt; 'Closed' and State &lt;&gt; '
           Executed'</value>
      </setting>
      <setting name="ResponseText" serializeAs="String">
        <value>ATTENTION: The work item finish date expires
           at {0}</value>
      </setting>
    </WorkItemReminder.Settings>
  </applicationSettings>
  <startup>
    <supportedRuntime version="v2.0.50727"/>
  </startup>
</configuration>
```

Listing B.6: Alert Reminder - app.config

```
<log4net debug="false">
  <appender name="FileAppender" type="log4net.Appender.
     FileAppender">
    <param name="File" value="log4net.txt" />
    <param name="AppendToFile" value="true" />
    <layout type="log4net.Layout.PatternLayout">
      <param name="ConversionPattern" value="%d [%t] %-5p %c
          - %m%n" />
    </layout>
  </appender>
  <root>
    <appender-ref ref="FileAppender"/>
    <level value="ALL"/>"
  </root>
</log4net>
```

Listing B.7: Alert Reminder - log4net.config