Web development roadmap

Master's Thesis
Jan Bolmeson

Examiner
Per Runeson, LTH

Supervisor
Daniel Karlström, LTH

Department of Communication Systems

LUND INSTITUTE OF TECHNOLOGY
Lund University

CODEN:LUTEDX(TETS-5531)/1-72/(2005) & local 12

Web development roadmap

- a methodology for structured development of web-based systems

Jan Bolmeson
Department of Communication Systems
Lund University

Supervisors: Prof. Per Runeson, Dr. Daniel Karlström

Abstract

Within a short period of time, the Internet and the World Wide Web have become ubiquitous, surpassing all other technological developments in our history. The Internet and especially company websites has rapidly evolved in their scope and extent of use, from being a little more than fixed advertising material, i.e. a "web presences", which had no particular influence for the company's business, to being one of the most essential parts of the company's core business. At the same time, the development and maintenance processes of web application have not progressed at a sufficiently rapid rate to meet the challenges and demands. Web application development is still in its infancy and the likelihood of a similar crisis as the "software crisis" in the 1960s is very high.

Traditional software engineering approaches with process models such as CMM and Waterfall models do not work very well since web system development differs from traditional development. Several methodologies and processes have been developed and tried, all with the aim to increase the development organization's flexibility and responsiveness to changes in the process, minimizing the development overhead as well as creating and carrying out repeatable and measurable development projects. Despite the proposed solutions the problems remain. The proposed methodologies are not used and an alarmingly large amount of web projects fail.

This report's aim is to give an introduction the web development roadmap and a new way of developing web based systems. In difference to the other proposed methodologies, the web development roadmap is a compilation of hands-on experiences combined with an adaptation of different traditional and modern software engineering and web engineering principles. The web development roadmap takes a slightly different approach since the main idea behind the roadmap is to let it serve as a toolbox with activities which can be used when the situation requires it.

Table of Contents

1 Introduction

Within a short period of time, the Internet and the World Wide Web have become ubiquitous, surpassing all other technological developments in our history[1]. The Internet and especially company websites has rapidly evolved in their scope and extent of use, from being a little more than fixed advertising material, i.e. a "web presences", which had no particular influence for the company's business[2], to being one of the most essential parts of the company's core business.

At the same time, the development and maintenance processes of web application have not progressed at a sufficiently rapid rate to meet the challenges and demands[18]. Web application development is still in its infancy and the likelihood of a similar crisis as the "software crisis" in the 1960s is very high[16]. Web engineering, in its current form, is an early attempt to identify the significant issues and problems and their solutions in developing Web-based applications. The aim is to, by using Web Engineering involve scientific, engineering and management principles and systematic approaches, be able to successfully develop, deploy and maintain high-quality web based systems and applications[3].

Many of the problems in web application development today descend from the notion that web development has been viewed in terms of "publishing" or "brand building or reinforcement". Business critical applications are developed much in the same way as web sites are, namely ad hoc.[18] The difference is that business critical systems such as e.g. online banks or large web stores have different needs to perform well, be delivered on-time, be reliable, be maintainable, and be secure. In order to meet these demands, a sound, reliable, systematic, measurable and repeatable development process is needed.

Traditional software engineering approaches with process models such as, for example, CMM and Waterfall models, do not work very well since web system development differs from traditional development. The development differs in several ways, for example, there is a large gap between traditional software engineering designs and concepts and the low-level implementation model[4], many of the web based system development activities are business-oriented (e.g. web sites are sales-oriented, web sites and intranets are content-oriented) and not engineering-oriented.

Software organizations often assign or outsource web application development to small teams of highly qualified but often very young developers. Frequently the developers' attitudes toward software engineering principles are less than positive - especially towards process improvement and metrics collection[5]. Part of the problem is the business context: development budgets are small, applications demand a faster time-to-market, continual change of requirements and the developers are often inexperienced in the principles of software engineering[5].

Several methodologies and processes have been developed and tried during the first years of 21st century. Agile processes like extreme programming (XP), Scrum, Crystal, Adaptive software development, Hypermedia models (HDM) all aim to increase the development organization's flexibility and responsiveness to changes in the process as well as minimizing the development overhead.

Despite the proposed solutions the problems remain. The proposed methodologies are not used and an alarmingly large amount of web projects still fail. One reason is that many of the methodologies are over-intellectualized. Another may be that they are written by engineers to engineers – the chance of marketing or other non-engineer personnel using these methodologies is less than slim.

This report's aim is to give an introduction to area of web engineering and the most accepted methodologies but at the same time summarize the findings and advantages of the proposed solutions so that they can be learned from. These findings together with the experience of project

managers, developers, designers and evaluators and end-users from the companies Ivtaco[56], Fernebrant Media[57], Quality System AB[58], Users View[59] and Uninet[60] is used to create a roadmap for web development. The main idea behind the roadmap is for it to be a hands-on and usable supporting tool during a web development process. The roadmap is supposed to be the tool for creating sound, easy-to-use, acceptable, structured and repeatable processes which over time can reduce the cost and development time of web projects.

1.1 Main issues at hand and the need of a methodology

The Cutter Consortium was the first company to show that more than 50% of web based projects suffered from schedule delays, budget exceeds, poor deliverable quality, poor functionality and more than 80% of the delivered systems did not meet the business requirements and needs[13].

Some of the common problems experienced in failed web development projects are described in the list below:

- Unstructured development projects which differ in structure from project to project
- Lack of testing
- Inexperienced developers with various background
- Difficulties with configuration management and documentation
- Difficulties with frequent changes in requirements
- Customers without clear goal specifications
- Budget and time-frame exceeds
- Low time-to-market frames
- No environment, programming or coding standards
- Lack of involvement of non-engineering personnel
- No maintenance or evolution planning
- Not mature web programming languages
- Lack of extensive stake-holder analysis
- High system complexity
- Difficulties with project and product measurement and evaluation

Many of these problems are not unique for web system development and were frequently occurring in early software development projects. Many of these problems have been solved in the traditional software development projects with the use of methodologies such as the Waterfall-process.

Development of simple web based systems such as a single or a couple of web pages do not necessitate a development methodology although it may draw advantages from it. But, it should be noted that even the smallest web project contains non-technical activities which are often disregarded – and hopefully can be avoided being ignored with a structured approach.

But in order to succeed with the development of an advanced web applications or a large website, a sound and structured methodology and process is crucial. A sound methodology can help understanding the different domains (requirements, objects, behaviors, business goals and rules etc)[13] of the web application as well as it addresses a certain level of quality, on-time delivery, budget constraints and risk management.

In the absence of a disciplined approach to web based systems development, we will sooner than later find out that our web based applications are not delivering the desired performance and quality , the development process becomes increasingly complex, difficult to manage and improve and at the same time very expensive and grossly behind schedule[18].

At the same time it is cost-ineffective to disregard from the knowledge elicited from years of software engineering. A structured process is repeatable and hence savings in both time and money should be possible. Similar requirements for different types of modules should be reusable, ready-to-use function and class libraries should be able to cut down development time and graphical stock-templates and stock-graphics should make graphical user interface prototyping easier.

Regardless of whether the project at hand is a new development or a rework of an existing web based system, a large number of issues need to be addressed. Several of these issues are of non-technical nature and since most development teams only consist of people with technical backgrounds, these issues are often overlooked. The process should be used as a support tool in order to avoid many of the common pitfalls.

At first glance it is easy to become overwhelmed of the scope of the project at hand, therefore it is extremely important to divide the project into smaller and more manageable pieces – phases. A phased approach gives a better project structure, makes the project more transparent and it automatically also becomes more manageable. The phased approach also allows processes to become reusable – i.e. lessons learnt from a phase in the process life-cycle can be used the next time the phase is entered and hence avoiding doing the same mistakes.

As all other software development processes a web development process must obey the four fundamental software engineering activities; software specification, software design and implementation, software verification and validation and finally software evolution.

But in order for a process to be successful in a web environment it must satisfy some additional requirements. Next to the traditional requirements such as systematic, measurable, repeatable, structural it must respond well to the rapid changes in requirements, rapid prototyping and rapid deployment and at the same time allow for extensive maintenance and evolution.

It must also at the same time regard the fact that many of the activities are of non-technical sort and conducted by non-engineers. Hence the process must be easy-to-use in order to get accepted throughout the developing organization and at the same time fulfill the needs of the engineering staff. But the single most important requirement is that the process must show increase in productivity or generate savings or increasing profit. If the process succeeds with the latter the organization will probably invest more funds and thus devote itself to continuous improvement.

To conclude, a sound approach to web development should:

- consist of a phased approach to the development life-cycle in order to make it manageable.
- consist of the fundamental SE-activities specification, design, implementation, verification and validation.
- be flexible enough in order to be able to respond to rapid changes in requirements
- be able to encompass other software engineering processes as sub-activities
- be able to decompose it self into activities of which only some are used
- be simple enough to be understood by non-engineers
- allow for extensive reuse of artifacts such as code-snippets, objects etc.

1.2 Master thesis work methodology

The author realized the problem with a structured approach to web application development for the first time during the work at Quality System AB, a Swedish company specializing in the development of web based systems. The problems which occurred during the development were in many ways similar to the problems described in Sommerville's Software Engineering[7] although Sommerville did not focus specially on web development.

The notion came around, that it should be possible to reduce development time as well as at the same time increase product quality if experience could be gained from the extensive theory of software engineering.

The master thesis started with an extensive literature analysis and Internet search. Since the phenomenon of structured web application development is fairly new there has not been very much research on the subject. Methodologies and papers were presented in academic papers with very few real-world studies or examples.

Based on the experience gained at Quality System AB the work started with developing a toolbox of described activities and a roadmap to be used as a supporting tool in future web development processes. The aim of the report was to elicit and describe process-activities based on hands-on experience and hence other companies which work with development of web applications were involved in the work with the web development roadmap.

Based on the common experience of carrying out successful web projects the proposed roadmap was reviewed several times in an iterative process. The roadmap was later also used as a supporting tool when Itime, (www.itime.se) a small web based tool to keep track on time, was developed.

1.3 Outline

Chapter 2 presents the differences between web applications and traditional applications. The following chapter, chapter 3, gives an overview of proposed process models and summarizes them. The next chapter, chapter 4, presents a discussion of why the models presented in chapter 3 are not widely used and introduces the suggestion of a web development roadmap. Chapter 5 summarizes the findings and discusses what could be done in the future in order to increase the efficiency of web application development. Appendix A presents the web development roadmap methodology illustrations and Appendix B contains the in-depth descriptions of the illustrated activities. Appendix C provides explanation and definition to frequently occurring abbreviations.

2 The special nature of web applications

Although the rapid development of the Internet, the web and the different web techniques not much have changed in the way most people develop web sites or web applications. Unfortunately the practices, which developers follow for web application development, today are as poor as the practices were when the web was in its infancy.[7] Important web applications are hacked in much the same way as important software systems were hacked before the dawn of software engineering in the 1960s and 1970s[6]. Web application development is today on the verge of experiencing a similar crisis as software did 1960s[7].

Computer scientist and software engineers try to avoid this deja vu by applying the knowledge gained from computer professionals that acknowledged that computer applications involved much more than expertise in programming and general intelligence. The discipline of web engineering has been created as a step towards creating a repeatable and systematic process for web application development, although not completely acknowledged more and more people start to acknowledge it as a discipline among disciplines[1, 6, 7, 15, 17, 16].

2.1 Levels of perception in web development

Figure 1 shows how web development can be perceived at six different levels[14,15] and at the same time illustrates the three different types of web based systems and how they relate to each other. These are the steps each organization, or developer, must climb in order to create a successful web based system or execute successful web based projects over time.

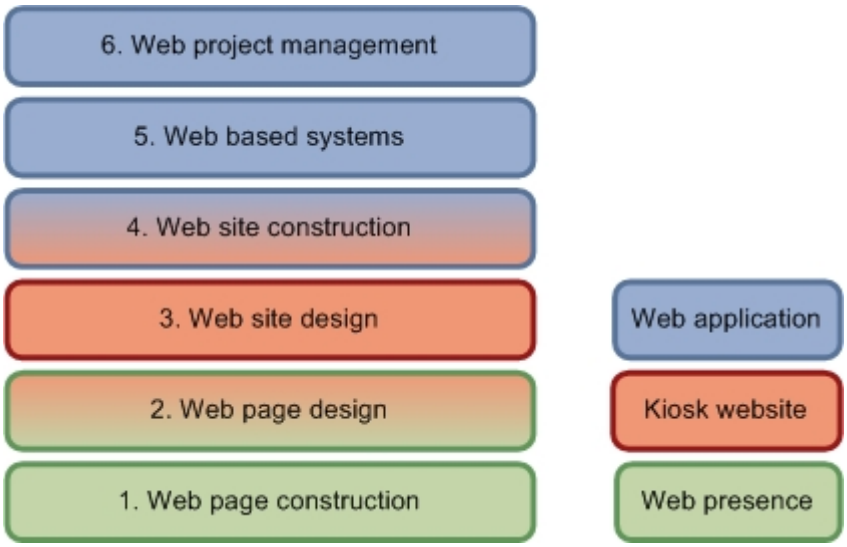


Figure 1: *The six levels of perception in web development related to the different types of web based systems*

The different shades of grey illustrate the three types of web based systems and to what levels of perception they correspond to. The web presence can be described as a small static collection of web pages. In order for an organization to create a web presence they need to be able to construct web pages and they must be capable of some web page design methods.

The vast majority of websites on the Internet can be categorized as kiosk websites – i.e. they are web sites whose main purpose is to present some kind of information and let the user navigate

through the information. The difference to the web presence is that the kiosk website has a clearly defined purpose and goal, the web site is continuously maintained and the organization sees the web site as a tool to be actively used.

The last kind of web system is the web application type. It is a website that works as an application where a set of web-pages form up the graphical user interface and performs a similar function for a set of users. These applications often require formal project management, stake-holder analysis, throughout requirements analysis, validation and verification and are often developed by a several persons with different and clearly defined roles.

These three different types of web systems require the organization to have reached different levels of maturity in order to create different types of web projects. An organization at level 3 has a completely different approach to a web development than an organization at level 1 or 6.

2.1.1 Level 1 - Webpage construction

For most people the web manifests it self through the web pages which are brought up and shown in the web browser. This is the outcome of the simplest and most visible level 1 in the figure 1. Pages are created in a mark-up language known as HTML which is very easy to understand, learn and master.

All that is needed in order to construct a webpage is a web browser and a text-editor. At this level there are today several integrated development environments, IDE, whereas Macromedia Dreamweaver and Microsoft FrontPage are the two most wide-spread, which assist in the web page construction. Thus, the developer does not need as in depth knowledge of HTML since the knowledge is embedded in the tool.

Web pages created at this level are almost always created ad hoc and with the only intention to satisfy the need to have a web presence with e.g. contact information with no regard to company graphical profile, visitor usability or accessibility.

2.1.2 Level 2 - Webpage design

When the organization has had a web presence for a while there are two possible ways to go. Either the organization decides to drop the website or to develop it further.

If the decision is to further develop the website, the next step into level 2 considers fonts, color schemes, document specific style sheets, consistent page navigational layout, some usability and accessibility considerations and some information structuring.

Some ethical and legal issues are also included at this level, due to the nature of the web and the visible source of web pages it is very easy to copy other people's ideas, images and designs. The ethical and legal issue is to know what can be copied and what is prohibited to copy[15].

2.1.3 Level 3 - Website design

The next step following design of separate pages into a set of pages is to create a policy concerning the whole site. The main purposes with the website is defined, time is spent on categorizing information not after the own organization but with regard to the potential user types, usability and accessibility requirements are defined as well as the different stakeholders are analyzed, simple requirement and problem definition documents are formed and the web site becomes a central part of the company marketing efforts.

2.1.4 Level 4 - Website construction

When level 3 is mastered and the web site has been online for a while some new problems occur. The web site has to be maintained and continuously updated. Having designed and constructed

several pages in the previous two levels, small site-wide changes tend to become very time-consuming and thus costly.

The need for a sound maintenance policy becomes apparent. The solution is presented in level 4 through a site-wide consistent design. Content and design is separated through external templates, external and site-wide style sheets and document mark-ups. Often the notion of a middleware, such as PHP, ASP or PERL, is introduced at this level.

2.1.5 Level 5 - Web based systems

The developer at level 3 and level 4 can create fairly complex web sites and web applications. But after a while the information stored in the repositories needs to be changed, and if consideration to information infrastructure has not been taken, there is a fairly large problem at hand.

Design for maintainability and scalability cannot be added later on but has to be incorporated from the beginning. Web based content management systems; CMS are today very popular as tools for information management. Quality factors as performance and security are other examples of other properties that also have to be incorporated from the beginning and which become of greater importance at level 5.

2.1.7 Level 6 - Web project management

The final step of maturity of the model in Figure 1 is the level of Web project planning and management. At this level risks are identified and managed, projects are well planned to meet requirements and business demands and maintainability is considered and taken into account in order to be able to repeat and continuously improve the development process.

2.1.8 Conclusion

As can bee seen in the previous sections, a developer or an organization, that has reached the last level of maturity will start and carry out a web project very differently from an organization situated at level 3 or lower. The explanation to the failure of many web projects, still today, is that organizations enters web development at level 3 and does not have the knowledge of the higher maturity levels.

Quality factors as defined by the ISO9126-1 2001[7] are not taken into account but most importantly the main purposes with websites and web applications are not identified and regarded. Web application development is seen as an implementation task and not as a task consisting of several non-programming activities such as information structuring, user classification, usability and accessibility requirements identification, graphical and aesthetical design as well as human-computer interaction design.

2.2. Common web application properties

One of the explanation to why web development is about to suffer the same crisis as the classical software engineering discipline did in the 1960s is that web applications differs in some parts from conventional software. Large web based systems, comparable to large software systems in function and number of lines of code, still have more similarities to small conventional programs developed by small teams than to its considerable larger counterparts.

The following list of properties points out some of the typical properties of web applications.

Web applications, web sites and web based systems…

- … often suffer from very compressed development schedules and budgets
- … are subjected to constant evolution with shortened revision cycles
- … are subjected to the notion that "content is king"
- … are often developed with insufficient requirements and problem definition specifications
- … are often developed by small development teams
- … are often developed by developers from a variety of backgrounds, experience and age.
- … are often built on emerging technologies and methodologies
- … are seldom tested entirely because of the lack of accepted testing processes
- … are more dependent of user satisfaction and are subjected to a bigger threat from competitors since it is relatively easy for the users to try competing web based systems.
- … are subjected to incomplete and evolving standards to which the web applications must comply, depending on the specific circumstances (e.g. accessibility standards for government sites)
- … require the developers to have understanding of additional disciplines such as hypertext, graphic design, information architecture
- … are often subjected to heavier security considerations due to the public nature of the Internet.
- … must often comply to several legal, social and ethical regulations
- … are often developed in rapidly evolving implementation environments, encompassing different hardware platforms which are often changed from project to project.

Figure 2: *Some of the most typical properties for web applications - it should be noted that most of these properties are applicable on the development of smaller traditional and conventional applications*[17,16].

Of the above mentioned items most of them are self-explanatory but number three ("content is king") and number eight ("user satisfaction and threat of competition") are worth elaborating on.

A web site's main purpose is to provide information through content to its user. Information systems have until a couple of years ago only handled transactional data which in most cases has consisted of numerical data with short textual description string. This type of data is easily structured, manipulated, sorted and searched.

The new type of web applications contains text, numbers as well as multimedia, i.e. pictures, media clips and audio. This information is not as easily structured, thus not manipulated, sorted nor searched. Furthermore a website quickly grows into thousand of files or, as in most cases today, into thousands of database entries. Web application easily mixes static files with dynamic files which freely accesses information stored in databases. As content the information is inseparably woven together with the procedural processing.[17, 16]

The second difference worth mentioning is the user satisfaction and the threat from one's competitors. In software engineering one of the main aims during requirements elicitation is to identify the user base. Most of the analysis and design methodologies are aimed towards user identification.[17] When the developers are creating user interfaces for their applications they must take into consideration both current, future and unknown users. Indirectly they also compete with other interfaces that both collaborators and competitors create, since users browse the web

and if they see possibilities and smart solutions in one web application they will no be satisfied with anything less in the other.

2.3 Web application characteristics

The size and complexity of web applications and web based systems vary widely. It can reach from a single-page presentation of the local company with a few hits per day to the official Sydney Olympic Web Site which received more than one billion hits during the two weeks the Olympics lasted[11].

The complexity does not have to be in terms of number of hits, as mentioned in the previous example, but can also be measured in terms of the dynamic nature of the content of the web application. For example online news-paper web applications often consists of several types of multimedia woven together in articles, it may range from simple images to radio and video clips.

The common factor for each web application, independently of the type, is that a web application is always dependent upon a good mix of reliability, aesthetics, content and performance. Figure 3 shows the characteristics of simple as well as of advanced web systems.

Simple web systems (kiosk websites)	**Advanced web systems (web apps.)**
Primarily textual information in non-core applications	Dynamic web pages because information changes with time and user's needs.
Static information content	Large volumes of information
Simple navigation	Difficult to navigate and find information
Limited usefulness	Integrated with database and other planning, scheduling and tracking systems.
Limited functionality or interactivity	Deployed in mission-critical applications.
Stand alone systems	Prepared for a seamless evolution.
High performance is not a major requirement	High performance and continuous availability is a necessity
Developed by a single individual or a small team	May require a larger development team with expertise in different areas.
Security requirements minimal	Calls for risk or security management and assessment
Easy to create	Needs configuration control management
Feedback from users minimal and not sought	Necessitates a project plan and management.
Web site mainly as an "identity" for the current clientele, and not as a medium of communication.	User satisfaction is vital.
Infrequent access	

Figure 3: *Comparison of simple and advanced web applications.* [16]

One example of each category may be a small company's web site seeking a web presence that contains contact information and a company presentation, whereas Microsoft's mail web application Hotmail is a good example of the latter.

Web applications have been divided further into different categories[16] as for example workflow, informational or transactional etc. This classification as well as the web application breakdown into static versus dynamic and visible versus invisible pages are well beyond the scope of this paper and can be found in the references cited.

2.4 The need for a structured approach

Development of simple web systems such as a single or a couple of web pages does not necessitate a development methodology although it may draw advantages from it. But, it should be noted that even the smallest web project contains non-technical activities which are often disregarded – and hopefully can be avoided being ignored with a structured approach.

But in order to succeed with the development of an advanced web applications or a large website, a sound and structured methodology and process is crucial. A sound methodology can help understanding the different domains (requirements, objects, behaviors, business goals and rules etc)[13] of the web application as well as it addresses a certain level of quality, on-time delivery, budget constraints and risk management.

The Cutter Consortium showed that more than 50% of web based project suffered from schedule delays, budget exceeds, poor deliverable quality, poor functionality and more than 80% of the delivered systems did not meet the business needs[13].

In the absence of a disciplined approach to Web-based systems development, we will sooner than later find out that our web based applications are not delivering the desired performance and quality, the development process becomes increasingly complex, difficult to manage and improve and at the same time very expensive and grossly behind schedule[18].

At the same time it is extremely cost-ineffective disregarding from the knowledge elicited from years of software engineering. A structured process is almost per definition repeatable and hence savings in both time and money should be possible. Similar requirements for different types of modules should be reusable, ready-to-use function and class libraries should be able to cut down development time and graphical stock-templates and stock-graphics should make graphical user interface prototyping easier.

2.5 Web engineering

The previous sections as well as the common sense, experience and realization shows that a structured, repeatable and easy-to-use process and methodology is to wish for.

While there are many differences and unique properties of web applications compared to conventional software applications, as discussed in sections 2.2 and 2.3, there are also common factors and similarities. These include; the need for methodologies, requirements elicitation, implementation, testing and maintenance of those parts that deal with programming and functionalities[16].

While web application development can draw advantages from software engineering in these areas, there are several more issues that need to be addressed to which software engineering does not provide any activities. For example, if Figure 1 is taken into consideration, only levels 4-6 deals with processes that normally fall under software engineering.

Other examples are for instance information structuring which falls completely outside the area of SE, or the user interface design which normally fall under the human-computer interaction

discipline[16]. Other examples of where web development reaches outside the software engineering scope are; the use of object-orientation (computer science, CS), the use of patterns (CS), the use of repositories as databases, lists etc (CS) and internet protocols as TCP/IP (information technology, IT).

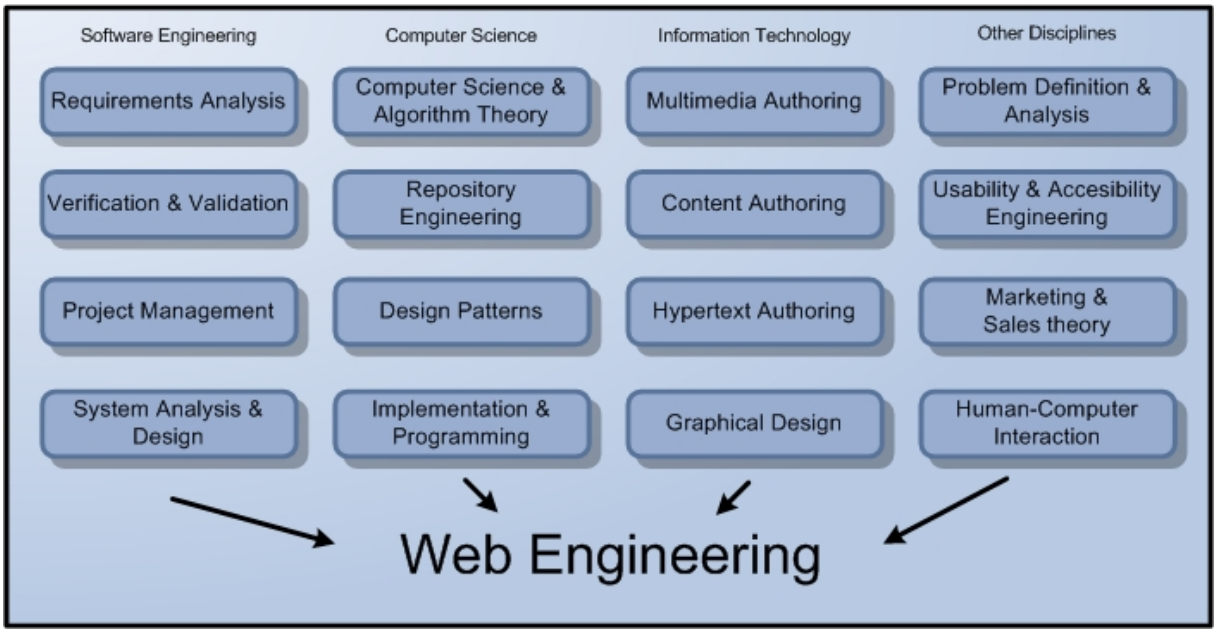


***Figure 4**: Web-engineering - a multidisciplinary field*[16,17,18].

The term proposed for "the practice and research in developing, testing and maintaining web applications" was, at the first international workshop in the year of 1998, Web Engineering[17]. According to its authors, Web Engineering is per definition:

"*The application of systematic, disciplined and quantifiable approaches to development, operation and maintenance of Web-based applications.*"[17]

Web engineering is today widely acknowledged, among computer scientist and academics, as a discipline among disciplines drawing from computer science, software engineering, information system and telecommunications and considered as discipline in its own right rather than subsuming under software engineering.

Although web engineering involves some programming and software development and at the same time involves some of the principles of software engineering, it should be noted that there are major differences between web applications development and software development as well as many differences between web engineering and software engineering.

1. Most websites and web-based systems are currently document-oriented containing either static or dynamic web pages.
2. Web-based systems are to a higher degree focused on look and feel, favoring visual creativity and incorporation of different media in presentation and interface. More emphasis is placed on visual creativity and presentation in front-end user interfaces.
3. Most web based systems will continue to be content-driven.
4. Most web based systems need to cater to users with diverse skills and capabilities, complicating human-computer interaction, user interface and information presentation to a multiplicity of user profiles.
5. The double nature of the Web serving as both an application and delivery medium is not

yet fully understood.

6. The Web exemplifies a greater bond between art and science that generally encountered in software development.
7. Most Web based systems need to be developed in short time, making it difficult to apply the same level of formal planning and testing as used in classical software development.
8. The Web is different from software related to the delivery medium. Traditional software generally operates in a well-defined environment whereas Web-based systems, especially at the user end, have to cater in diverse environments and platforms.
9. The type of individuals who develop Web-based systems are vastly varied in their background, skills, knowledge and systems understanding, and as well as their perception of Web quality and Web based systems.
10. Although the web based systems usability success rates are improving, they are still at approximately 67%[19]. This means that in a web-based system one third of the time, the user doesn't find the information he or she is looking for.

Figure 5: Differences concerning Web Engineering vs. Software Engineering and Software Development vs. Web-application Development

3 Proposed iterative methodologies

According to Sommerville[7], a software process is a set of activities and associated results which lead to the production of a software product. Conventional software processes usually address, as mentioned before, the four development phases: analysis, design, implementation, test and finally maintenance and evolution. All software engineering methodologies address these phases, although in different ways. Due to the special nature of web applications, described in Chapter 2, the methodologies best suited for web application development are iterative, incremental and evolutionary methodologies.

The incremental and evolutionary methodologies have, among other, the advantages of not letting the customers wait until the complete system is developed in order to gain from it. The first increment is produced to satisfy the customer's most immediate needs and can be immediately used. The second advantage is that customers can be involved early in the development process and the first increment can be used to gain knowledge on the requirements for the later phases. Hence the customer can change requirements and gain system experience during the development process.

Furthermore the risk of failure is lesser, since the problems encountered during development is quickly identified and encountered in the prototypes and it is most likely that the final product will be successfully delivered to the customer. The final advantage is that since the most important parts of the system are developed first and later increments integrated into the original parts of the system, it is inevitable that the first and most important parts of the system receive most of the testing. Thus the customer is less likely to encounter problems in the core parts of the system and thus the overall stability of the system is ensured[7].

This chapter introduces some methodologies especially adapted for the web as they have been proposed.

3.1 Web application development process evolution

3.1.1 Web engineering and Software engineering phase comparison

Due to the special nature of web applications a process well suited for web system development must regard several issues not usually addressed by software engineering. R. Glass abstracts in the 2001 Cutter IT Journal[8] a methodology framework for developing web applications. It is a projection of web application development phases onto the four fundamental software engineering activities software specification, software development, software validation and software maintenance and evolution.

Glass argues that during the software specification phase, developers should build a model of the applications in the terms of a domain. Ideally this domain should concentrate on the problem to be solved. The domain should be separated from the software considerations and whether the application solving the problem is to be delivered on the web or not.

During the next phase, the software development phase, a model based on the solution should be produced. It should be clear that the web application development process, in contrast to the general software development process should, assumes a web-based solution and hence must accommodate this. During the following implementation phase the model should be transformed into actual software.

When the implementation is finished the transformation should be verified and validated against the requirements and the problem definition. If the implementation validates, the product either iterates again or is released and moves on into the last phase.

The last phase, the software maintenance and evolution phase, is concerned with modifications and extensions to the software, which may occur in any of the previous phases. General software models are hard to relate to the web development model since web application's implementation model is decomposed into a level of resources[3]. The resources have unique addresses (URLs) and they are delivered on request from a web browser to a web server. The responses from the web server can be generated dynamically or can be completely static. Even so, they are inherently specific, which means that they are hard to be captured in classic abstractions.

3.1.2 The simple document resource model of the web

As the previous section states, one of the largest problems in relating web applications to general software is the notion of resources. The World Wide Web was originally developed as a medium to distribute research findings between researchers at different universities. The main objective was to make it easy to publish notes, research documents and conference proceedings. The notion of Web application development essentially boiled down to document development by an author or a small number of authors[3].

In relation to Glass' abstraction, the first web developers, during the analysis phase, decided on the content of the document, i.e. what should be published. Then during the next phase, the design phase, they decided on how the document was going to fit into a specific context, i.e. how to structure the document into the web site or into the hyperlinked chunks of information. Then the next step was to mark-up the specified content with HTML and CSS so that it formatted well in a web browser. It was roughly tested, i.e. checked that if it had the desired look-and-feel. Finally the maintenance was handled by the authors themselves and it mainly consisted of adding, updating or modifying the existing information[8].

As can be seen very few of the previously described activities are of traditional software engineering type. This web implementation model was designed to meet the life-cycle requirements of a document and it was deliberately kept simple in order. The development process of single page websites or so called "web presences" is still very similar to the previously described.

3.1.3 Major web development process activities

But even if the previously described methodology is well suited for small web sites it does not satisfy the requirements of the kiosk websites nor advanced web applications. The following sections introduce the activities to which all web development processes in some way must respond to or regard.

The introductory analysis phase is even more important in web development than in traditional development. During the analysis phase the main purposes, stakeholders and goals should be identified and analyzed. The overall function and operational environment of the system, including the corporate and business objectives should be understood and analyzed. This phase is especially important since stakeholders – especially visitor-type stakeholders often have to be approximated due to the enormous variety of visitor-types that a web site may have.

During the following specification phase the findings of the previous phase are specified. A model should be built of the application domain separated from software considerations. The phase should focus on the requirements required in order to solve the problem defined in the analysis phase. Furthermore should the general project properties, such as project planning, web configuration management and project follow-up be outlined during this phase.

If the proposed solution is supposed to be web-based the following phase regards the conceptual design phase. During the conceptual design phase tasks such as navigational design should be regarded. The conceptual design should also explore both the human and cultural aspects of the

web based system. This is especially important for large international web based systems which are accessed by users from several different countries and cultures.

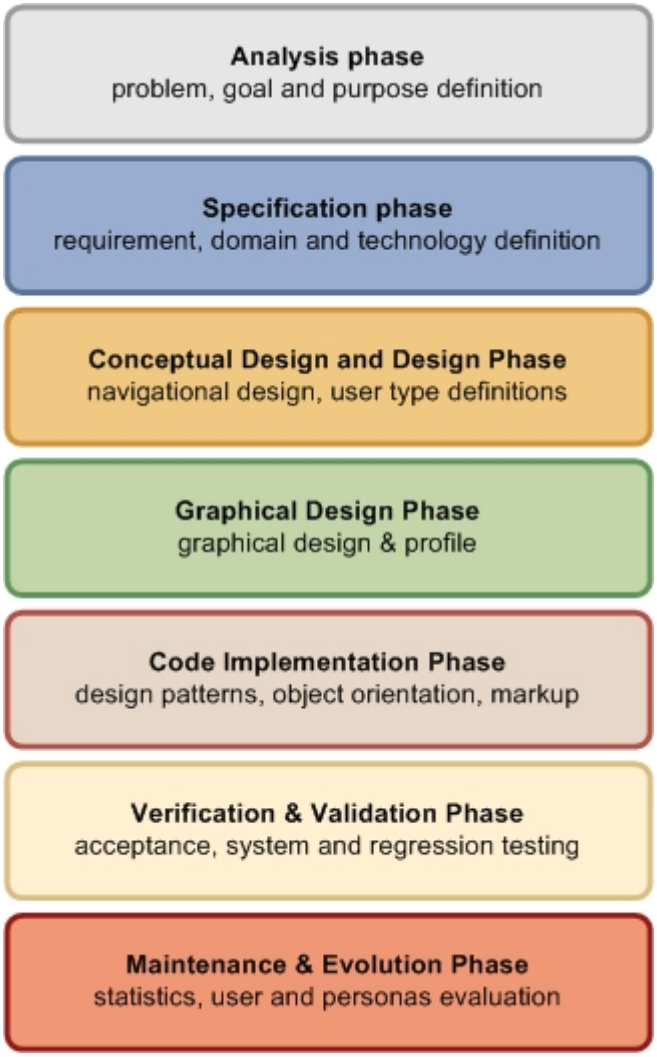


***Figure 6**: Major web development process activities.*

The phase following the conceptual design is the graphical design phase. The graphical phase is almost as important as the functionality since for the most users the web manifests itself as the pages that are viewed in the browser. Hence not only the graphical user interface is important but as important is the web system's look and feel.

When the conceptual design and the graphical prototype are finished the next step is to transform the different designs into actual web pages and server-side functionality. The implementation should follow prevalent standards such as object-orientation, refactoring, make use of design patterns and separation of logic, content and design.

During the verification and validation phase some traditional software testing approaches such as unit, system and regression testing should be used. But additional web specific testing should be conducted. The web specific testing may include xhtml and css-standard compliance testing as well as different user testing.

The last test conducted in the verification and validation phase is the acceptance test and when the acceptance test is approved the application moves into the last activity of the current life-cycle – the maintenance and evolution phase. In every iteration of the methodology chosen, it should be detailed how content maintenance should be carried out. The reason is that web maintenance is even more, than in traditional software, a continuous activity. Depending on the nature of the application the maintenance can become quite complex and not solely rely on the technical domain. This especially applies to content generation which obviously lies outside the technical domain.

3.2 Software reuse approaches

As previously mentioned the best known software engineering models such as the waterfall model, explorative models, the prototype model and spiral model are not easily applicable on web application development. They can be used only with serious constrains and this making them ineffective. Since the late 1980s several new models have been introduced that, while based on the classical process models, focused on the object-oriented development of software systems [24].

In comparison with the first programming languages and software engineering the first server side web application programming languages were function oriented. But since the introduction

of the re-use based methodologies and the common acceptance of the advantages with object-oriented code, the function-oriented methodologies has almost become obsolete[7,25]. But it should be noted that the client-side programming languages, or script languages, are still mainly procedural.

There are different ways to achieve reuse in the context of web applications - templates in the form of HTML and descriptions in the form of XML may be reused[29]. Information and data can be reused by accessing shared databases. Furthermore components that exhibit some kind of non-trivial behavior can be reused. An example is the code implementing a shopping basket in different e-commerce applications. Even though many of the supporting web technologies do not support reuse, e.g. they do not support inheritance, polymorphism or object composition[28] mechanisms in HTML or XML. For example web based shopping baskets might not be found in a single component but decomposed into several.

As a consequence the most important kind of reuse - design and architectural reuse - has not been addressed nor explored thoroughly[29]. One reason may be due to the non object-oriented nature of the Web. Navigation patterns as a way to record, convey and reuse design experience has been introduced[30], since complex corporate web-applications need a way to maximize their reuse of larger design structures.

Appendix A describes and introduces several different proposed methodologies for maximizing the reuse of different components and parts of web applications as well as serve as repeatable, systematic and structured process models.

3.3 Comparison of the Process Models

The World Wide Web was, as mentioned in section 3.1, originally developed as medium to distribute scientific papers. Thus the first methodologies were developed to make this process as simple as possible and the web implementation model was designed to meet the document's life-cycles[36]. The methodologies are based on the assumption of resources, i.e. mostly self-contained chunks of information. The resources are maintained and authored independently of other resources and links were just means of combining different sets of information into web sites and semantically related structures[36]. But since the early and mid-1990s the Web has reached far beyond its original purpose - Internet is today one of the most important infrastructures of the western world, with uncountable different areas of use. The notion of resources was no longer enough to fulfill the needs.

At first engineers, developing web based systems, leaned towards using Software Engineering process models such as the Waterfall models and Spiral models, but they quickly realized that these models were unsuitable for web development.

The hypermedia community then developed the first hypermedia process models. According to the definition found at google.com[41] hypermedia is *"The combination of text, video, graphic images, sound, hyperlinks, and other elements in the form typical of Web documents."* The first proposed model was the HDM model for the structured design of hypertext-applications[42]. But the problem with the HDM model is that it is more a design model than a process model[24]. Hence further development of the process models was needed.

The next step was to incorporate the notion of object-orientation into the previously developed models and hence the OOHDM was born. At the same time a new branch, based on relational models, of process models was developed. RMM became the most widely-spread process based on ER-notation.

But similarly to their predecessors, both the RMM and OOHDM models suffered from several serious disadvantages, for example, both models require extensive specialized training. The most

recent process models take a slightly different approach to web development compared to the earlier ones. Modern process models such as XP, WSDM and WCPM are more aimed towards being toolboxes with tools which can be used to solve different problems and situations.

The following figure serves as a brief overview, as well as a comparison, of the proposed process models and methodologies for web development. All models are introduced and described shortly in Appendix A.

Figure 7. Presented methods summary and evaluation.

Model	Advantage	Disadvantage	Main Characteristics	Conclusion
Waterfall	Is easily applicable as a sub-process since the phases are easy to understand. Makes many processes or sub-processes transparent	The process is very rigid and focuses on process artifacts such as design documents which do take time to develop and hence increases development time. A mistake in the requirements phase can not be detected until near the end, when the customer gets to see the (nearly finished) product.	Divided into four phases: • Analysis • Design • Implementation • Test	The different waterfall model phases may make nice diagrams in books, but they do not exist in reality. But it should be noted that an adaptation of the Waterfall model can be used as a sub-process in other more general and overarching processes.
Spiral models	Builds on an iterative methodology which favors incremental development, rapid prototyping, increased parallelism, and concurrency in design and builds activities. Clearly defined phases.	The traditional spiral models are not specially designed or developed for web development and hence do not address or support any of the special activities needed.	Divided into four phases: • Analysis • Design • Implementation • Test	Although the model needs some adaptation before it can be used extensively for web development it is very good base for further process improvement and evolution.

Model	Advantage	Disadvantage	Main Characteristics	Conclusion
XP	XP's greatest advantage is that it is becoming more and more accepted in the SE community and hence it is well-documented and people are gaining experience in the work methodology. It allows for rapid development, small development teams and quick turnovers during development.	It is a general process which is not specialized for web development and it does not address some of the more specific web issues – such as the non-technical or non-coding tasks. It's also relative hard to implement into an organization due to inertia to accept new methods such as pair-programming, test-driven development and documentation-less development.	Incremental and iterative developments - small improvements after small ones Continuous, often repeated automated unit test, regression testing. Pair programming Onsite Customer Refactoring Shared code ownership Simplicity Feedback Organizing the system with a metaphor Sustainable pace	XP definitively shows promise as a methodology suited for web development. It should be noted that it is not directly applicable but many of the XP concepts are directly transformable to web engineering processes.
HDM	One of the greatest advantages with the HDM method is its ability to decompose hypertext applications into fine-grained artifacts for reuse while considering and ensuring structural relations. It can also be integrated with other process models in the design phase.	The applicability is severely limited by the methodology. Does not allow reuse through object-oriented principles. Mapping of a hypertext-application designed with HDM to web applications is difficult.	ER Design Entity Types Derived Entities Navigational Design Links (Perspective, Structural, Application) Derived links Browsing Semantics Different semantics	Being the first model proposed, HDM suffered from many weaknesses and hence it is today almost obsolete. Some of the concepts has been developed further and has served as a base for several other methodologies such as the OOHDM.

Model	Advantage	Disadvantage	Main Characteristics	Conclusion
OOHDM	Offers a clearly defined procedure for the development of hypermedia-applications. Modularity and reusability of design concepts is relatively high due to the degree of abstraction found in the resulting models	Generality of the modeling approach tend to lead to a higher complexity. Does not provide assistance for automating the implementation of reusable design artifacts. Lacking system support that adequately corresponds to the basic principles of the Web. Formalized diagrammatic notations and require specialized training[44]	Conceptual Modeling Conceptual classes, objects Navigation modeling Navigational classes, objects, links Navigational transformation Abstract Object Interface Models Abstract Data View (ADV) Implementation	Being a further development of the HDM method OOHDM did not suffer from as many weaknesses as the HDM did. But the largest problem with the OOHDM method is that it is not a method which is easy to use. Many consider it over-intellectualized and hence it is not widely used. However some of the activities have successfully extracted and implemented into other models.
RMM	Relational model is well understood and supported[45] Is not tied to a specific implementation[45] Hypermedia model relates well to domain[45] Problem domain model is clearly separated from the application (navigation) model: for one conceptual model it is possible to build different application models[43].	Restricted to a specific class of applications[45] Focuses on design and implementation phases of hypermedia product lifecycle and leads the seamless evolution to the lifecycle of a Web-application unsolved[45]. Limited expressiveness (especially compared to OOHDM)[45] Maintenance and reuse are mainly limited to data rather than artifacts or components. Reuse of artifacts in other processes is not explicitly supported. Formalized diagrammatic notations and require specialized training[44]	ER design context Entity, attribute, relationships Slice Design context Navigational Design context Node-link Conversion context User-Interface Design context Hyperbase population context Prototyping context	RMM introduced when it was published many good ideas and concepts – among other things it was based on the well understood ER-model. However it still is a process which does not regard all the activities of web development and it is not likely to be used by other than trained software engineers.

Model	Advantage	Disadvantage	Main Characteristics	Conclusion
WSDM	User-centered Works well with informational web sites Increases automatically the usability. Addresses “soft”, non-technical issues. Easy-to-use and grasp and hence easy to implement in an organization.	Limited to one type of web-sites - "kiosk" web-sites Typical problem areas such as maintenance are not addressed. No explicit support for reuse. Does not address technical issues such as coding-issues, object orientation etc.	Mission Statement Specification Audience Modeling Conceptual Design Implementation Design Implementation	The WSDM method is the only one of the presented that truly considers the activities concerned when creating a web site. This is its main strength but also its enormous weakness – it cannot be used for anything else then the so called kiosk websites. However the concepts presented are usable for larger web application developments, but only for the front-end of the application. Since the method does not address technical issues such as code-testing, object modeling or component reuse.
WCPM	Allows for integration of arbitrary processes. The reuse model is explicitly supported	Requires specialized training	Process model Reuse management Component technology WCML documents	The WCPM model is the most recent proposed model. It is based upon extensive reuse management and incorporation of other methodologies, these are great advantages but unfortunately it suffers from the same problems as its predecessors – it is very hard to use in any practical situation. It requires extensive specialized training.

3.4 Why the hypermedia models are not used

Currently there is one problem, that all the methodologies mentioned in the previous section, have in common – the models are not used. One of the most obvious reasons to why the methods are not used may be lack of knowledge that these methods exist. Most of the methodologies are presented in academic papers and as either conference proceedings or as conference workshop papers. It is a well known fact that practitioners have little interest in academic journals or conferences. At the time of writing (January 2005) there are two books at Amazon covering Web Engineering, although it is well acknowledged that books move a discipline onwards[44].

Another explanation may be inertia - the software development industry is well-known for being very slow in accepting changes in techniques even when the usefulness in the techniques are not questionable. This fact is well-supported by the slow acceptance that agile methodologies experience. Many companies have problems with the transition from older methodologies to, for example Extreme Programming, since it embraces many seemingly unorthodox principles, such as pair programming, documentation-less development and increased productivity in the short term.

One other explanation is probably that most of the other proposed methodologies presented are only documented in academic papers of 3000 to 5000 words together with incomplete and insufficient examples of the methods at work. Hence the methods are very difficult to apply in a real-world environment and together with the fact of custom-built compilers for mapping specific project-designs to executable code makes many of the methodologies direct unsuitable for application development. Garzotto and Paolini thought, when they introduced the first versions of HDM, that it would be an approach towards the development of application generators and compilers, but this has not happened in practice. The research models, experimental tools, artifact decomposition methods, document standards has so far failed to create a lasting impact on the practitioners community or even creating a new generation of CASE tools. In the absence of these tools the community is still bound to use paper-based methods and hence ad-hoc methods.

Barry & Lang state in their report[54] on how different methodologies are used, that understand ability, ease-of-use and widespread acceptance and reputation among developers are major issues in method selection for web system development. Methods such as RMM and OOHDM which are better documented than many of their counterparts and based on well-known modeling techniques such as UML and ER-notations are developed by software engineers to be used by software engineers. Hence the probability that graphic designers, marketing personnel, end users or other stakeholders without specialized training in the formalized grammatical notations of ER, will use these modeling concepts is very low.

3.5 How to get developers to take advantage of the methodologies.

It is acknowledged that web based systems has higher demands on process methodologies than other systems. Although the formalized methods proposed are not used by practitioners they should not be discarded as useless. For any method to be used it is required that they should be heavily abstracted that is they should prescribe broad guidelines instead of over-intellectualized task-lists[44]. It has been shown that a methodology cannot in any case be better then the people who apply the method. Hence the methodology must be adapted to the people working with it and the situation at hand. Methodologies and techniques should therefore rather be tools in an organizations toolbox for software development applicable in different situations and by different persons.

The over-intellectualized methodologies such as the HDM, OOHDM, RMM and WebComposition should be decomposed into a kind of patterns which are applicable in different situations. For example the HDM methodology gives a very good abstraction of the components of a webpage into entities. The ideas of domain separation in the RMM method should be considered. Parts of the knowledge originated from the hypermedia methods should be mapped into usable tools[44]. Since the lack of CASE-tools is one of the reasons for non-use, a new generation of easy-to-use CASE-tools could very well introduce the methodologies behind them.

Another way is to get the findings of the academic world into the web development community. This can be done by posting articles in industry magazines, posting and supporting discussions in internet forums or by engaging in consultancy etc. Fortunately the trend is on its way towards this. Articles, such as excerpts from this thesis, are starting to get published and acknowledged, at least in the open-source community. Methodologies are more and more presented as tools and solutions to different kind of problems, very much in the same way as patterns are.

4 The Web Development Roadmap

The following sections of this report present such a toolbox for development of web sites and other web based systems. The roadmap is a graphical representation of the process involved of creating a web based system in a sound, structured and repeatable manner. It is based partly on the cumulative experience gathered by the developers, project managers and consultants working at the four Swedish companies Ivtaco, Users View, Quality Systems AB and Uninet AB, and partly on the ideas and concepts presented in chapter 3 of this report as well as on the Australian company Step Two Design's articles on Intranet Development[61]. Hence the process is not founded on a theoretical basis but rather on the practical experience. The process' aim is not to be yet another proposed methodology for web development but to be a hands-on process that actually can be used in the several different ways that a toolbox can be used.

The idea is that the roadmap can be used as a reference, a check-list or just as a box of tips during the whole development life-cycle. It should give an overview on what should be done in each phase as well as introduce what is to come.

4.1 Structure of the web development roadmap

The web application roadmap is provided in three supporting forms – a picture outlining the process methodology as a whole[a], a spiral-model of the process methodology to illustrate the different iterations in each life-cycle and finally it consist of supporting details and guidelines for use, provided in Appendix C.

In the process methodology illustration, the core activities are depicted vertically in the centre of the picture, marked with a grey background and red-colored connecting arrows in between. These core activities, with straight corners, underpin the creation of the following activity and hence both time and resources should be focused on conducting these core activities.

The majority of core activities, in the process methodology illustration, have supporting activities depicted as the boxes, with rounded corners, on the left and right of the core activities. These indicate techniques that might be appropriate or most effective. It should be noted that most techniques that deal with programming language specific issues at hand are PHP-oriented. PHP is a widely-used general-purpose scripting language that is especially suited for Web development, more information on PHP may be found on the PHP website at http://www.php.net.

Microsoft offers some similar support with the .net platform but the support is generally not free and it does not enjoy the same support in the developer community. But it should be noted that the web application roadmap can be used by companies or developers working in all environments since most of the process-activities are platform-independent.

The next form of the web development roadmap is the web development spiral. As stated in chapter 3 the most suited processes for web development are iterative evolutionary techniques. The best way to illustrate the iterative life-cycles of the methodology is as spirals. The different spirals illustrate the core activities in each development process. The idea is to use the spirals to get a quick overview of the process and what phase the project is currently in, then to use the process methodology illustration to get a grasp of the involved techniques and sub-activities and then finally to use the sections in Appendix C to get the additional in-depth information on each technique and core-activity.

---

[a] Cf. Appendix B

4.2 Using the web development roadmap

As previously stated the spiral and process methodology illustrations give a quick overview of the core activities in each project phase. The methodology starts at the top of the first figure in Appendix B, at start, and works down towards the "launch web site" or "launch web application". As the spiral pictures illustrate it is not a simple waterfall-like top-down process but a highly iterative one – hence the spiral illustrations and the double-arrows in the process methodologies.

The order of the activities in which they should be conducted is illustrated. The activities have been carefully positioned to indicate which pre-requisite tasks have to be completed before the next activity can be executed. To mention an example, before html-testing may start the middle-ware, which generates the html, must be tested successfully.

Not all of the activities will be needed in every project and that is not either the main aim of this roadmap. The purpose is that this roadmap should work as toolbox where certain tools may be available although they will not be used. But in any larger web development most activities will be necessary. The roadmap also regards the differences in creating a new web based system or redeveloping (i.e. maintenance, or further evolution) an existing one in the following sections.

4.3 Web development roadmap – pre-development activities

The purpose of the pre-development activities is to build a solid foundation for the web development project to come. The main aim of the phase is to identify and elicit the goals, purposes and overall user-requirements of the system as well as determine whether it is economically reasonable to carry out the project. If the results show that the project should be carried out, the activities aim at getting support for the development within the organization by developing business cases and arranging funding.

4.4 Kiosk web site methodology

The kiosk website spiral on the previous page illustrates the process described in section 4.7 in an iterative, evolutionary and prototype-based way. The advantages of having an iterative evolutionary process for web development are many and described in chapter 3.

The process starts in the middle of the spiral with the requirements specification which is considered the base for the spiral model. The requirements in turn serve as a foundation for the following conceptual designs which is followed by the page and graphical layouts. When the conceptual and graphical designs are finished they should be evaluated through user and personas testing as described in the previous sub-chapter. It should be noted that it is advantageous to create several designs of each sort in order to be able to really select the most appropriate – cf. the collaborative technique described in section 4.6.3.

The test results should be reviewed and a modification specification should be made. This specification specifies the next step which is to select one, or improve one of the conceptual designs. If the conceptual design is changed or improved the following graphical design should also be updated.

After the first graphical profile has been created the first implementation can be conducted. Although the content drafting activity precedes the first actual implementation activity in the illustration, these to activities may be conducted at the same time. Then the first prototype is tested – initially from a technical point of view with unit and regression test and later tested from the user's point of view.

The results of the test then form the modification specification and a new life-cycle begins. The spiral continues to work in this way until it is time for the acceptance test which is conducted when the system is considered to be finished. During the acceptance test it is decided whether to publish the system or not. If the system is taken online the first post-development activities take place.

4.5 Web application methodology

As stated in Appendix B.8 there are two types of web based systems; the kiosk-websites and the web application web sites. In reality most major web sites consists of these two types seaminglessly integrated into each other. The kiosk-website is the front which the visitor sees and uses, whereas the web application is used for content-maintenance and authoring.

The development of a web application is very similar to the development of a kiosk website. The only difference lies in the size of the development and hence business criticality. Experience shows that more time and resources have to be spent on testing in order to ensure the stability and functionality of the system. Hence several other traditional software engineering activities are used and carried out. Examples of such activities are metrics collection and evaluation, extensive regression, system and real-user testing, risk analysis and requirement reviews.

In many cases it is also advantageous to let the web development process encompass other process methodologies for different and specific phases, for example XP is very well suited to be used as a sub-process within the implementation phase.

5 Conclusion

There is no doubt that web based applications, information systems and web sites are growing in use, acceptance and importance. Despite the rapid growth both experience and several investigations show that the majority of the web based systems we have today are still developed ad hoc.

The need for a sound, structured and repeatable process is unquestionable – high performance and business critical information systems are very hard to develop without proper processes and activities – not if a certain degree of quality is to be sustained and the systems delivered on time and satisfying the requirements. Solutions and processes have been proposed but they have not managed to change the development approach.

The reasons are many but the most important reason is that most of the proposed methodologies have not become accepted due their internal complexity and requirement of specialized training. Most of the methodologies are also only presented in relatively short academic papers which in general most developers or companies does not read.

Experience shows that in order for a process to become accepted it has to be well easy to understand, enjoy quite a support by a community and at the same time be presented in printed book-form. At the same time the potential user of the process does have to realize the benefits of using such a process or methodology.

The suggestion in this paper consists of the web development roadmap which approaches the problem from a slightly different angle then the other presented methodologies. The roadmap is kept simple, in order to avoid complexity and its main focus is to reduce development time and thus the cost, by presenting and describing essential core activities. The roadmap is also partly based on experience gathered during countless web based project and thus the approach is more hands-on than many other theoretically based models.

At same time the roadmap moves towards the notion of processes being more toolboxes from which activities can be drawn and used when the situation requires it. Hence an experienced developer can incorporate several other methodologies into the roadmap without having to modify it.

5.1 Benefits of the web development roadmap

The web application roadmap has been drawn from the practical experience gained by working with countless web development projects. It is at the same time also based on the theoretical findings of the proposed methodologies presented in section 3. Hence the web development roadmap offers several benefits for teams or developers participating in web development projects.

For example it provides a hands-on and best-practice methodology which can assist teams and stakeholders to coordinate and successfully accomplish a web development project. It prioritizes certain key activities and hence ensures that no important activities are missed. Furthermore it offers guidance on suitable techniques in each phase of the project and gives tips on where to find related material. Since most of the methodology is written in a non-technical way and hence can be used be non-engineer staff it also helps bridge the communication barriers between different web development project stakeholders.

In regard to the description of a sound methodology for web development mentioned in section 1.1 the web development roadmap:

- does consist of phased approach based on the ideas of the traditional evolutionary spiral model.

- does encompass the fundamental and traditional software engineering activities such as analysis, specification, design, implementation and verification and validation.
- is flexible and responds well to rapid changes in requirements since it is based on the notion of incremental development with different software-increments and prototypes.
- is able to encompass other process models as sub-activities since it only is a general framework with different phases and proposed activities. For example traditional review-techniques are applicable as well as modern agile methodologies such as extreme programming (XP).
- is decomposable into both different phases and activities. The roadmap offers a recommended path to take during the development, but it does not require the user to use it. The activities can be seen as parts of a toolbox where only some tools are needed at some times.
- considers the fact that a web development involves personnel from several disciplines and hence the activities are customized for their intended audience. Programmers and engineers receive detailed and in-depth information on coding-standards, object libraries etc, whereas the content authors receive web authoring guides especially designed for the web system's intended audience.
- allows and encourages extensive reuse of both process artifacts such as requirement documents and low-level object models or code snippets.

5.2 Drawbacks of the web development roadmap

The identified drawbacks of the web development roadmap in its current state are:

1. The documentation of the web development roadmap is at best inadequate.
2. The web development roadmap is not completely platform or technique independent. Some approaches are completely dependable on the development frameworks chosen.
3. The web development roadmap has not been tested in enterprise application development.

5.3 Looking ahead

Although the web engineering discipline is relatively young compared to other technical disciplines, it is gaining more and more attention by researchers, developers, academics and more importantly by the practitioners and end-users. Web engineering still does need to evolve and mature to effectively be able to handle the challenges presented by today's and future complex web based systems. Another important aspect to be noted is that the evolution of processes and methodologies are moving towards the "toolbox"-type of processes and activities. The goal is that every organization or developer should a set of tools at disposal which can be used and applied in different situations.

At the same time different processes' activities and ideas are transferred from one process to another – cf. the WebComposition process which allows for an internal waterfall or XP process within its scope. This is an area that needs further study together with other areas such as web system testing, verification and validation, web system metrics, quality control and assurance, web engineering education and requirements analysis and system design. More effort should also be spent on creating practical and hands-on approaches which can be used outside the academic world and thus make it more acceptable to the people working with web projects.

The web plays an essential role in the western society and more and more system and infrastructure activities depend on the Internet and the web. The need for an engineering approach, the need for a sound, structured and repeatable process will only increase over time. The hope is that the industry working with the web will move from a "nice-to-adopt" to a "must adopt" attitude and strategy for the development of large, high-performance, evolutionary and/or mission-critical web based systems[5].

References[b]

1. Ginige, A. and Murugesan, S. "Web engineering: An introduction", *IEEE Multimedia*, 2001, vol. 8, no.1, pp. 14-18
   Can also be found at: http://computer.org/multimedia/mu2001/pdf/u1014.pdf
2. Ricca and Tonella, P. "Testing Processes of Web Applications", *Annals of Software Engineering,* vol. 14, 2002, Kluwer Academic Publishers, pp. 93-114
3. Murugesan, S. and Deshpande, Y. *Web Engineering,* Springer, 2001
4. Maurer, F. and Martel S., "Extreme Programming - Rapid Development for Web-Based Applications", *IEEE Internet Computing,* January-February 2002, pp. 86-90
5. Ginige, A and Murugesan, S., "The Essence of Web Engineering", *IEEE Multimedia*, 2001, vol. 8, no.2, pp. 22-25
6. Pressman, S. "What a tangled web we weave", *IEEE Multimedia*, 2000, vol. 17, no. 1, pp. 18-21
7. Sommerville, I. *Software Engineering 6th edition*. Harlow, England: Addison-Wesley. 2001.
8. Glass, R. "Who is right in the web development debate?", 2001, *Cutter IT Journal*, vol. 14, no. 7, pp. 6-10
9. Holzschlag, M. E. "Integrated Design - How Specialization Limited the Web", 2001, *Web Techniques*, Sept. 2001 can also be found at:
   http://www.newarchitectmag.com/archives/2001/09/desi/
10. The Sydney 2000 Olympic Network,
    http://compnetworking.about.com/library/weekly/aa082900a.htm
11. Lee, C. S. and Shirani I. A., "A component based methodology for Web application development", 2004, *Journal of Systems and Software*, vol. 71, pp. 177-187
12. Schwabe, D. et al. "Engineering for Reuse", *IEEE Multimedia*, 2001, vol. 8, no. 1, pp. 20-31
13. Epner, M., "Poor Project Management Number-One Problem of Outsourced E-Projects", *Research Briefs*, Cutter Consortium, 7 November 2000.
14. Ginige, A. "Web Engineering: Managing the Complexity of Web Systems Development", Workshop on web engineering, 2002, pp. 721-729
15. Deshpande, Y. et al. "Web Engineering", *Journal of Web Engineering*, 2002, vol. 1, no. 1.
16. Deshpande, Y. and Hansen, S. "Web Engineering: Creating a Discipline among Disciplines", *IEEE Multimedia*, 2001, vol. 8, no. 2, pp. 82-87
17. A Little History of the World Wide Web, http://www.w3.org/History.html
18. Murugesan, S. et al. "Web Engineering: A new Discipline for Development of Web-Based Systems", *WebEngineering2000*, LNCS 2016, pp. 3-13, 2001
19. Olsen, H. "Web-usability is improving" Can also be found at:
    http://www.guuui.com/posting.php?id=1540)

[b] All URLs verified on 2005-04-14

20. Dart, S., "Containing the Web Crisis Using Configuration Management", *Proc ICSE Workshop on Web Engineering*, 1999.

21. Mendes, E., Mosley, N. and Counsell, S. "Web Metrics - Estimating Design and Authoring Effort", *IEEE Multimedia Special Issue on Web Engineering*, 2001, vol 8, no 1, pp 50-57

22. Olsina, L. and Rossi, G. "Measuring Web Application Quality with WebQEM", *IEEE Multimedia*, 2002, vol2, pp 20-29

23. Schwabe, D. and Rossi, G. "Developing Hypermedia Applications using OOHDM",1998

24. Gaedke, M. and Gräf, G. "Development and Evolution of Web-Applications Using the WebComposition Process Model", *International Workshop on Web Engineering at the 9th International World-Wide Web Conference (WWW9)*, Amsterdam,The Netherlands, May 15, 2000. Can also be found at: http://www.teco.edu/~gaedke/paper/2000-www9-webe.pdf

25. Yourdan, E. "Modern Structured Analysis", Yourdon Press, Englewood Cliffs, New Jersey, 1989.

26. Schwabe, D. "HDM - A Model-Based Approach to Hypertext Application Design", http://www.inf.udec.cl/~yfarran/HDM.htm

27. German, M. "The Object Oriented Hypermedia Design Method", Power-Point Presentation 23 Oct, 2003

28. Voostind. "What's so good about OOP?",*Answer on the Sitepoint Advanced PHP Forum*, http://www.sitepoint.com/forums/showthread.php?t=59898#post439958

29. Schwabe, D. et al. "Web Design Frameworks: An approach to improve reuse in Web Applications", can also be found at: http://www.inf.puc-rio.br/~schwabe/papers/WWW9WebEngineering.pdf

30. Rossi, G. and Garrido, A. "Capturing Hypermedia Functionality in an Obkect Oriented Framework", *Building Object-Oriented Application Frame-works*, Wiley, 1999

31. Schwabe, D. "Resumo do OOHDM", http://www.oohdm.inf.puc-rio.br:8668/space/resumo+do+OOHDM

32. OOHDM Wiki at http://www.oohdm.inf.puc-rio.br:8668/space/start

33. Isakowitz, T. et al., "RMM: A Methodology for Structured Hypermedia Design", 1995 can also be found at: http://delivery.acm.org/10.1145/210000/208346/p34-isakowitz.pdf

34. Gaedke, M. "Web Engineering – Aspects of Reuse in the Web", PowerPoint presentation http://research.microsoft.com/collaboration/university/europe/events/dotnetcc/Version4/Slides/gaedke.ppt

35. De Troyer, O.M.F., "WSDM: A User Centered Design Method for Web Sites", *Proceedings of the seventh international conference on World Wide Web 7*, pp 85-94, 1998.

36. Gellersen H.-W. and Gaedke M, "Object-oriented Web application development", IEEE Internet Computing 1999, vol. 3, pp60-68

37. Turowski, K., "Web Engineering - Objectives and Research Directions", *SAP Innovation Congress 2003 PowerPoint presentation"*, can also be found at: http://wi2.wiwi.uni-augsburg.de/downloads/Topical_Keynote_SAP_IC.pdf

38. Gaedke, M. et al. "A Repository to facilitate Reuse in Component-based Web Engineering", *International Workshop on Web Engineering at the 8th International World-Wide Web Conference (WWW8),* Toronto, Ontario, Canada 1999

39. McClure, C.L., "Software Reuse techniques: adding a reuse to the system development process", Prentice Hall, Upper Saddle River, N.J. 1997

40. Obreiter, P., "Applying component based web engineering in a international Enterprise", http://wwwipd.ira.uka.de/~obreiter/publications/2000AIT.pdf

41. Google Search results on the term "hypermedia", http://www.google.se/search?hl=sv&lr=&oi=defmore&q=define:hypermedia

42. Schwinger, M. "Towards Modeling of DataWeb Applications", PowerPoint presentation at: http://www.schwinger.at/LIB/2000/AMCIS2000/AMCIS2000.ppt

43. Barna, P., "Methodologies for Web Information System Design", Can also be found at: http://wwwis.win.tue.nl/~hera/papers/ITCC2003b/itcc2003b.pdf

44. Lang, M. "Hypermedia Systems Development: Do We Really Need New Methods?" *IS2002 Preceedings*, http://proceedings.informingscience.org/IS2002Proceedings/papers/Lang148Hyper.pdf

45. Goble, C. "Information Retrieval, Hypermedia and the Web", PowerPoint Presentation at: http://www.cs.man.ac.uk/~carole/teaching/cs3352/cs3352-10.ppt

46. Maurer, F. and Martel, S. "Extreme Programming - Rapid Development for Web-Based Applications", *IEEE Internet Computing*, 2002, vol:2 pp 91.

47. Liu, D. "Extreme Programming", Wiki at http://c2.com/cgi/wiki?ExtremeProgramming

48. Agile alliance manifesto http://www.agilealliance.org

49. Williams. L. et al. "Strengthening the Case for Pair Programming*", IEEE Software*, vol. 17, no 4, 2000

50. Bauer, M. "Agile Web Development", http://www.designit.com.au/articles/agile_web_development

51. Feature Driven Development http://www.pcoad.com/download/bookpdfs/jmcuch06.pdf

52. Kate, "Extreme Programming and Web Development Process", Extreme Programming and Web Development Process

53. Uden, L. "Design Process for Web Applications", IEEE Multimedia 2002 vol. 2, pp.47-55

54. Barry, C. and Lang, M. "A Survey of Multimedia and Web Development Techniques and Methodology Usage.", IEEE Multimedia, 2001 vol. 8, pp. 52-60

55. Baresi, L. et al., "From Web Sites to Web Applications: New Issues for Conceptual Modeling" http://www.elet.polimi.it/upload/baresi/pub/papers/WWWCM.pdf

56. Bolmeson. J, Kiosk Website developer and project manager. Can be contacted at: jan.bolmeson@ivtaco.com.

57. Fernebrant, M, Art Director and Graphical Designer. Can be contacted at: magnus@fernebrant.se

58. Nyman, CE, CEO, claes-erik.nyman@qualitysystemab.com and Bolmeson, J., CTO, jan.bolmeson@qualitysystemab.com.

59. Westerlund, B. and Orremo, H., Usability and Accessibility Experts, bjorn@usersview.se and henrik@usersview.se.

60. Elisabeth Brevensson, CEO, eb@uninet.se and Thomas Björck, CTO, tb@uninet.se

61. Step Two Design, "Intranet Roadmap", http://www.steptwo.com.au

62. Calabria, T., "An introduction to personas and how to create them", 2 May 2004 http://www.steptwo.com.au/papers/kmc_personas/

Appendix A – Proposed web development methodology descriptions

A.1 Waterfall process model

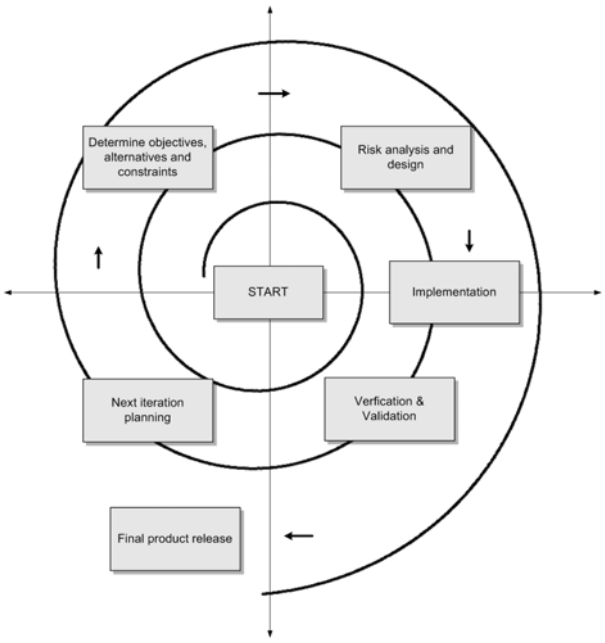
The waterfall process model is the classical representation of the software development lifecycle. It is a model that is broken down into four main phases; analysis, design, implementation and testing. It is called the waterfall model since each artifact in the model seems to logically flow into the next. The waterfall model was the first process to be presented and it was based on the empirical observation that the cost of changes rises exponentially by phases. The conclusion was that big decisions were to be made up front because changing them is very expensive.

The waterfall model is a very rigid and stiff model, not very well suited for web development since 30 years of progress in programming languages, databases and development practices has largely voided this assumption, but it is buried so deep that it seems certain to last for some time.

A.2 Spiral model

While the waterfall methodology offers an orderly structure for software development, demands for reduced time-to-market make its serial steps inappropriate. Hence the next logical and evolutionary step from the waterfall process model is an approach where the various steps are staged for multiple deliveries or handoffs.

The final evolution from the waterfall model is the spiral model, taking advantage of the fact[7] that development projects work best when they are both incremental and iterative. That is, when the team starts small and benefits from enlightened trial and error along the way.



Graphical illustration of the spiral process model

The evolutionary process model begins at the centre position and moves in a clockwise direction, cf. figure on the following page. Each traversal of the spiral typically results in a deliverable. For example, the first and second spiral traversals may result in the production of a product specification and a prototype, respectively. Subsequent traversals may then produce more sophisticated versions of the software.

The spiral process model reflects the relationship of tasks with rapid prototyping, increased parallelism, and concurrency in design and builds activities.

The process should still be planned methodically, with tasks and deliverables identified for each step in the spiral. Furthermore the process can encompass other process models and hence prototyping may be used in one spiral to resolve requirement uncertainties and hence reduce risks. This may then be followed by a conventional waterfall development.

A.3 Extreme Programming, XP

Extreme Programming, XP, is one of the most popular modern agile software development processes. Its aim is to increase software organization's responsiveness while decreasing the development overhead. The XP method focuses on delivering executable code over artifacts and tends to see the people programming the code as the project's greatest asset. XP is, unlike the other methodologies presented, not a methodology specialized for the Web[46].

The reason to why it is presented is that it responds well to many of the situations which arise during web development. For example, many web projects are, as previously mentioned, developed by small teams and with compact time-frames – issues that XP handles very well.

XP Overview

XP consist of twelve related practices and works best for small development teams consisting of somewhere between 5-15 people. Rather than focus on deliverables such as requirement documents, design papers and different plans - the objective is to create as much executable code and automated test drivers as possible in the shortest amount of time. This works very well with web projects where the time-to-market is often very limited. The difference between using XP and classical ad hoc development and hacking of web applications, lies within the notion that XP highly values simple design, counter hacking claims by emphasizing refactoring, strong regression testing and continuous code inspection by, for example, pair programming[46].

XP is well suited for web development in more aspects than with the relatively low time-to-market. The focus on small teams allows for face-to-face communications instead of a paper-based one. This also reflects the nature of Web applications well since in Web projects documentation is often postponed due to lack of time[46]. The face-to-face communication has also the advantages that technical writers do not have to make assumptions of what the readers want to know. Face-to-face communications are faster and the communication is often limited to existing issues[46].

Maurer and Martel at the Glasgow University shows that productivity gains, in a studied company with 16 developers of a Web based application, reached to somewhere between 66% and 300%[46]. While the development process went from a fairly constant Java and object-oriented approach to XP - their findings show that all other aspects of the project stayed constant.

XP Practices

Customer Satisfaction

One of the first practices that XP addresses is customer satisfaction - building the web's greatest and most high-qualitative application is irrelevant if it does not meet the customers' demands and needs. XP addresses customer satisfaction mainly in two different ways - one way is to have an on-site customer and a kind of prototyping with several smaller releases where the most important functionality is implemented first.

On-site Customer. Identifying and analyzing requirements correctly is essential for any software project to succeed. As previously stated, this can be a problem in web-based projects due to rapid and frequent changes in requirements. XP handles this problem by initially collecting user stories, i.e. textual user scenarios[46], and getting a good understandment of the problem at hand. To clarify the requirements elicited and prioritizing correctly, the XP methodology uses an on-site customer representative. Thus the programmers can, when an issue arises, ask the on-site customer instead of speculating on customer preferences[46]. The other advantage of an on-site customer is that requirements can be changed, explained and implemented on very short notice - thus leading the development team on right track and focus development efforts on the most pressing needs. It

should be noted that in most cases an on-site customer can be replaced with a good customer-communication policy, i.e. the customer does not have to be on-site – the communication can take place through email or over the phone.

Small releases. Since requirements may change frequently XP focuses on executable and working deliverables during each iteration of the spiral model. The notion behind the executable releases is to provide the customer with a release that has at least a partial business value. This creates project-transparency for the customer, thus reducing failed projects due to not-met business goals and at the same time it reduces customer risks. Another advantage of small releases is that the impact of planning errors is reduced[46].

Software Quality

Metaphors. The main aim of XP, as previously stated, is to maintain software quality without slowing down the development process. One of the first things done in the XP methodology is to abstract the main purpose of the system into a metaphor - the metaphor can sometimes be a single user story that portrays the idea and gives everyone the system basics[46]. In comparison with traditional Software Engineering this metaphor can be seen as the most high-level system architecture abstracted.

Testing. That testing improves software quality is a well-known fact within the software engineering community. Thus testing is an essential part of XP. The customer defines an acceptance test which the development team implements[46]. Each developer writes unit tests before writing the actual code, thus ensuring the programmers think about the problem before starting the implementation. The other purpose of the unit tests is to serve as a detailed specification of the method's functionality. The unit tests furthermore serve as material for an extensive regression testing.

Simple Design. XP is a methodology that as, other agile methodologies, assumes that requirements are always changing, the future evolution of the system uncertain and that the costs of change are not exponential. Hence, the most cost-effective development should focus on solving today's problems and not be concerned of future evolution directions. The idea is that the savings done by not making bad guesses on the system futures will be higher than the costs of anticipating and implementing changes to the system[46,47,48]. Another assumption is that no documentation is needed if the code is well-documented and well-structured.

Refactoring. As other software source-code, web application source code tends to deteriorate over time. XP's suggested solution to the problem is to favor refactoring, thus keeping the source code as simple as possible. This is also a requirement since no other documentation exists.

Pair programming. One of XP's more discussed issues is pair programming, i.e. letting two persons share the same computer. Even if the developer cost is doubled it has been shown that the overhead effort is approximately 15% higher than solo-programming but with the effects that both the software quality and the job satisfaction is increased[49] as well as the calendar time reduced.

Development Process

The planning game. Interests and responsibilities in XP are divided mainly into two categories, business interests and technical responsibilities. The business interest decide on the scope and priority as well as on release date, i.e. when must the next release be available. In order to be able to answer these questions, feedback is needed from the technical staff whose responsibility it is to make reasonable estimates, describe consequences on directions taken and maintain the process and keep the team organized[46]. The goal of the planning game is to balance customer interests with the expertise of the development team. The developers estimate the time and effort, while

the customer picks tasks for the next iteration with the only constraint being the development speed of the group[46].

Collective ownership. In XP all the members own all the code. Developers that can add value to a portion of time is allowed and even required to do so[46]. The practice creates a greater identification of the group with the code and works against territorial opposition. The practice works partly because of the notion of automated unit tests - a developer can directly get feedback on whether the change added value or not.

Coding standards. Due to the collective ownership a coding standard is needed since different developers code in different ways. The coding standards purpose is to make the code easier to understand as well as to improve consistency between members.

Continuous integration. Developers should integrate code as often as possible. Hence it is ensured that there is always an executable version with all the newest functionality that can serve as a baseline for all work. The executable version may then also serve as project indicator for the customer and other stakeholders.

A.4 Other agile methodologies

The previous section described the Extreme Programming (XP) methodology and some adaptations to web development. This section presents other suggested agile methodologies well suited for web development. Agile methodologies are well suited since they are mostly light-weight, relatively easy to implement and they help focus on the direction without the overhead cost of more traditional software engineering approaches that are rigid and cumbersome[50].

Feature Driven Development (FDD)

FDD is a model-driven short-iteration process. The process starts with defining an overall model shape. The next step in the model is a series of two-week "design by feature, build by feature"-iterations, i.e. the process aims at having a working result every two weeks[51]. The features are defined to be "useful in the eyes of the client"-results. The reason for the latter definition is that most of the iterative processes are everything but short and "useful in the eyes of the client". An iteration like building an abstraction layer is not directly customer-friendly[51].

The FDD clearly defines the roles and responsibilities required which is a clear advantage in web development projects where many different persons or competences are required. The one to two-week timeframe is also especially well suited for web development. Another important issue that the FDD supports is the issue of scalability - a process must be able to be scaled both upwards and downwards. FDD supports the notion since it is technology independent to the extent of object-orientation.

A.5 The Hypertext Design Model (HDM)

One of the first models introduced was the Hypertext Design Model, which was a model for the structured design of hypertext applications, i.e. relatively simple web sites and web based systems. It was primarily developed by Schwabe, Garzotto and Paolini in the early 1990s.

The main idea behind HDM is to decompose chunks of information into sizeable structures called entities. An entity is a physical or conceptual model of the domain and can be for example a company policy document or a specific car. These entities are then grouped into different entity types, which are abstractions from the real-world. Thus an entity is the smallest part, not conditioned by the existence of other entities, of information that can be added or removed from an application.

The entity is then divided further into components. Each unit shows the content of a component under a particular perspective. Entities therefore derive their information content from their components, which in turn derive it from their units[26]. The units are the smallest parts of information that can be showed by an HDM-application and they can easily be compared with the notion of "web pages".

A HDM information structure is then interconnected by three different categories of links. Structural links that interconnect components belonging to the same entity, perspective links that connect different units of the same component and finally application links that connect same or different types of entities and components in arbitrary patterns set up by the author[26].

In a view of a web based system the structural links interconnect the different parts of a document, cf. an index in a book. The Perspective links interconnect for example all English or Swedish sections of a document. Application links can be compared to a library registry keeping track of all the books in the library.[26]

The different kinds of links then have different representational roles. A hypertext application can namely roughly be divided into two parts - a hyperbase structure and an access structure. The hyperbase structure represents the core of the application. It consists of entities, components, units and the, in the previous section discussed, three types of links. The hyperbase, i.e. the application, is browsed or used by traversing the different kinds of links.

The only thing that has to be done before the user can start exploring the hyperbase is to define a starting point, i.e. an entry point to the application. Access structures have the purpose of allowing the user to choose convenient entry points for further navigation.

Hierarchy term	**Example 1**	**Example 2**
Entity Groups or Types	"Company Policy Library"	"Cars"
Entity	Company Policy Document - "Environment policy - S31"	Volvo
Component	Chapter 1 in the "Environment policy - S31" is a component of the entity	"Engine" is component of the car Volvo
Component-child	Section 1 in Chapter 1 in the "Environment policy - S31" is a child of the component "Chapter 1"	"Start-engine" is component-child of "Engine"
Perspective	in English	non-functional
Component + Perspective = Unit	Section 1 in Chapter 1 in the "Environment policy - S31" in English	A non-functional "Start-engine"

Explanation of the hierarchical terms within the HDM[18,26].

HDM can be used in different ways such as a modeling device or as an implementation device. When it is used as a modeling device it supports producing high level specifications of existing or yet to be developed applications. On the other hand if HDM is used as an implementation device it can serve as a basis for designing tools that directly support the development of the application. One of the central advantages of HDM in the design and practical construction of hypertext applications is that the definition of a significant number of links can be derived automatically from a conceptual-design level description.[26]

A.6 The Object Oriented Hypermedia Design Method (OOHDM)

The Object-Oriented Hypermedia Design Method, OOHDM is the natural evolution of the HDM discussed in section A.5. The OOHDM is more adapted to the new generation of the Web and Internet that we have today. The OOHDM argues that navigation and functionality of a web-based system should be seaminglessly integrated as well as the navigational structure should be decoupled from the domain model of the application[27]. Furthermore in contrast to the HDM the OOHDM offers a clearly defined procedure for developing hypermedia applications.

OOHDM consists of four main activities - Conceptual design, Navigational Design, Abstract Interface Design and Implementation. They are performed in a mix of incremental, iterative and prototype-based development styles[23]. During each activity a set of object-oriented models describing the actual activity are built upon the previous iterations. The main purpose of the model is to maximize abstraction power and reuse opportunities throughout the process[31]. This is done through classification, aggregation and generalization/specialization. The following figure summarizes the steps, products, mechanisms and design concerns in OOHDM.

Activity	**Products**	**Formalisms**	**Mechanisms**	**Design concerns**
Requirements gathering	Use Cases, Annotations	Scenarios; User Interaction Diagrams; Design Patterns	Scenario and Use Case Analysis, Interviews, UID mapping to Conceptual Model	Capture the stakeholder requirements for the application.
Conceptual Design	Classes, sub-systems, relationships, attribute perspectives	Object-Oriented Modeling constructs; Design Patterns	Classification, aggregation, generalization and specialization	Model the semantics of the application domain
Navigational Design	Nodes, links, access structures, navigational contexts, navigational transformations	Object-Oriented Views; Object-Oriented State charts; Context Classes; Design Patterns; User Centered Scenarios	Classification, Aggregation, generalization and specialization.	Takes into account user profile and task. Emphasis on cognitive aspects. Build the navigational structure of the application
Abstract Interface Design	Abstract interface objects, responses to external events, interface transformations	Abstract Data Views; Configuration Diagrams; ADV-Charts; Design Patterns	Mapping between navigation and perceptible objects	Model perceptible objects, implementing chosen metaphors. Describe interface for navigational objects. Define lay-out of interface objects
Implementation	Running application	Those supported by the target environment	Those provided by the target environment	Performance, completeness

Summary of the steps, products, mechanisms and design concerns of OOHDM as presented by OOHDM's author Schwabe, D. on the OOHDM wiki[32].

Requirements gathering

The first step before developing a software system is always to define what is to be developed. This is done by a feasibility study which is followed by a requirements elicitation and analysis[7]. In order to succeed with this task - all stakeholders need to be identified and so do the tasks they are

supposed to perform. Next, scenarios for different user-types and stakeholders are collected or drafted. The scenarios are then used as a base for creating use cases, which is represented by User Interaction Diagrams. The diagrams provide a graphical decomposition of the interaction between the system and the user during the execution of a task.

Conceptual design

When the requirements phase of an iteration is finished a conceptual model of the application domain is created by using object-oriented modeling principles, augmented with some primitives such as attribute perspectives[30] - similar to the HDM-perspectives discussed in earlier. In the conceptual design phase there is no concern for the type of uses nor tasks - only for the application domain semantics. A conceptual schema is built upon the design of sub-systems, classes and relationships[30].

It is important to capture the domain semantics as "neutrally" as possible, i.e. represent objects, their relationships and collaborations in a well known object-oriented model and disregard from the types of users and tasks. Methods to produce or acquire the result of this phase - the conceptual schema - are not provided by OOHDM but are up to the engineer to decide upon.

Navigational design

In the third phase of the iteration, the navigational structure of the application in terms of navigational contexts is described. That is, the navigational model is a subjective view of the conceptual model. The navigational design should be induced from navigation classes such as nodes, links, indices, and guided tours[30]. These navigational classes take into account the different personas[c] and tasks.

Nodes in OOHDM represent views on the conceptual classes defined in the previous phase. Different navigational models can be designed to express different views on the same domain, i.e. different perspectives. By defining the navigational semantics in terms of nodes and links, we can model the movement in the navigational space independently of the conceptual model. This can be explained as the numbers of links to other pages (subset of nodes) that a user can choose to browse at any given moment. The navigational model is not locked to the conceptual model but can evolve independently, thus for example simplifying maintenance[30].

The products of the navigational design are the navigational class schema that defines the navigable objects of the application. The second product - the navigational context schema defines how navigable objects are clustered together and navigated. Thus the navigational model takes into account what objects will be navigated, whether navigational objects might look different depending on the context in which they are navigated and which connections and access structures exist among the navigable objects.

Abstract interface model

The abstract interface model is created by defining perceptible objects such as pictures and graphical representations of the interface classes. Interface classes are defined as aggregations of primitive classes such as HTML-elements and recursively of interface classes. Interface objects map to navigational objects, providing a perceptible appearance. Interface behavior is declared by specifying how to handle external and user-generated events and how communication takes place between interface and navigational objects[30].

---

[c] Cf appendix D for definition

Implementation

During the last phase, the implementation maps interface objects to implementation objects and may involve well-known and elaborated architectures such as client-server or repository architectures. In these architectures the applications can be seen as clients to a repository holding conceptual objects.

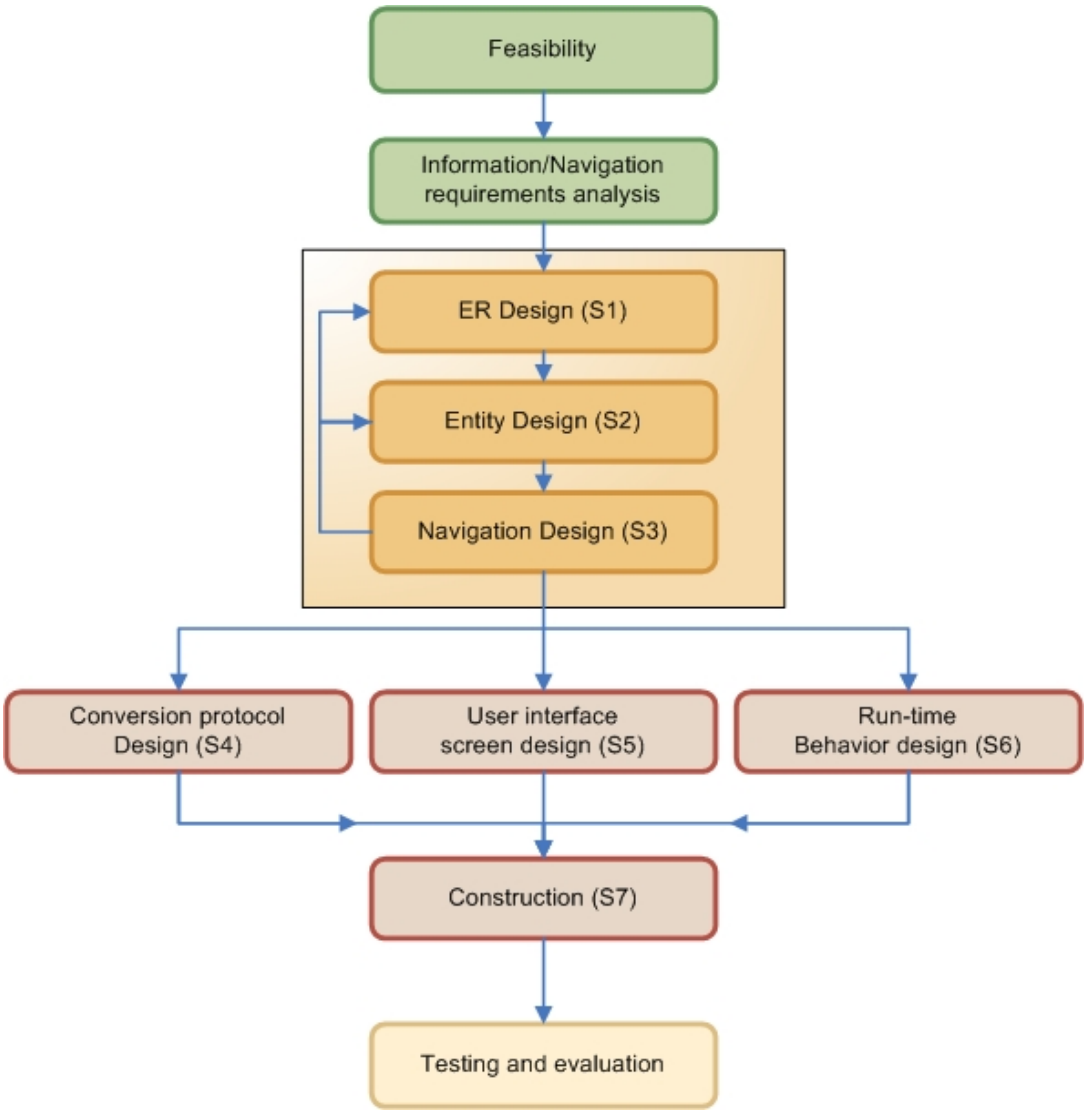
OOHDM Conclusion

The main purpose of the OOHDM-methodology is to seamlessly integrate navigation and functionality as well as decouple the navigational structure of the domain model of the application. This is based on the three cornerstones of the approach which are:

1. Navigation objects are views (in the database sense) of conceptual models
2. The use of appropriate abstractions to organize the navigational space
3. The separation of interface issues from navigation issues.

A.7 Relationship Management Method (RMM)

Next to the HDM methodology the Relationship Management Method (RMM) was one of the first methodologies developed in 1995 by Isakowitz et al. The name "relationship management" stems from the authors view of hypermedia as a way for managing relationships among information objects[33]. The RMM-method is a platform-independent process-model, divided into seven detailed phases, for hypermedia applications that can also be used for the development of web applications. The figure below depicts the complete software development cycle, with the seven phases marked S1-S7.



The RMM design methodology[33]. The arrows connecting the various stages are tagged with the objects to be used as input for the next step. The picture depicts only two feedback-loops but the RMM supports feedback loops for all objects and phases.

S1 - ER Design

When the initial requirements document is in place the second step is to represent the information domain of the application via an entity-relationship (ER) diagram[33]. The entities and relations created in this phase will later serve as a basis for the hypermedia application. In many situations ER-diagrams may be reused directly - if for instance the repository is a relationship

database. The main objective of the ER diagram is to make links between the objects explicit, as these are the navigable elements, through which the user browses the web applications[33]. An analysis of the domain using the ER approach helps identify important relations and can help solving the problem with designers who only link entities that are semantically related[34].

S2 - Entity Design

This step in the figure, which is unique to hypermedia applications, determines how the information in the chosen entities will be presented to the user and how the user will be able to access it. In the simplest form all information can be displayed to a user on a single webpage with a scrollbar. Although this approach is very simple for the developer the usability aspects are not very good - the user would more appreciate the information subdivided into separate but interrelated wholes. An example can be, instead of listing all companies in a phonebook in one section it might be advantageous organizing the companies by first-letter or business area.

The organization of entities into slices is called the slice design phase[33]. The links between the different slices are so called structural links - which are consistent with the HDM notation.

S3 - Navigational design

In the third step the paths, which will form the links through which the user may navigate, are designed. Each associative relationship in the diagram created in step 1 is analyzed and if it according to the requirements analysis should be made available for navigation it is replaced by one or more access structure, cf. the HDM access structure notation. By default access structures enter an entity via its head slice but the designer is able to specify a different entry point.

S4 - Conversion protocol design

In the fourth step a set of conversion rules transforms the ER-diagrams and structures into an object of the target platform - in our case a web page. This transformation can be made either manually or by an off-the-shelf CASE-tool[33].

S5 - User-interface design

This phase involves mapping of each ER-object into a screen layout. It includes button layouts, appearances of nodes and indices and location of navigational aids[33].

S6 - Run-time behaviors design

In contrast to phase S5, the decisions on link traversal, history, backtracking, and navigational mechanisms are implemented in this phase. The developers are also to consider whether to build node contents and link endpoints at application development or whether to have it computed at runtime. The RMM methodology, although geared toward the latter, also supports the former[33].

S7 - Construction and testing

Construction and testing is performed as in traditional software engineering, but special consideration should be taken to test and validate all navigational paths.

A.8 Web Site Design Method (WSDM)

A user centered design method for creating web sites WSDM (pronounced WISDOM) is a user-centered method for the design of kiosk web sites. The WSDM model is a model considerably different from the other presented since it is explicitly starting from the requirements of the users or visitors. The WSDM-model solves most problems caused by the fact that most web sites does not have any underlying design at all, or that they are data-driven[35].

Introduction

Most of the methodologies presented in the previous sections address the problem with the lack of design methods for general web based information systems. The methods were originally developed for hypertext or hypermedia applications in the early or mid-1990s and do not in a good way deal with the web specific issues at hand.

The WSDM method proposes that the starting point for developing a web based information system should the requests of the potential visitors or users of the system. Most web development projects namely start at the other end - how do we categorize and structure our information. The problem is that in the majority of cases information is grouped semantically in the own organization's hierarchy - but what does a visitor from the outside know about the organization's internal hierarchy? Methodologies that use the information-first-approach are called data-driven whereas the WSDM methodology is called user-centered.

Designing "kiosk"-websites

According to Isakowitz[33] web information systems are dividable into two main categories - the kiosk-type websites and application-type web information systems. The vast majority of the websites on the Internet can be called kiosk-type websites. They are web sites whose main purpose is to present some kind of information and let the user navigate through the information cf. www.cnn.com website. An application-type of website is a website that works as an application where a set of web-pages form up the graphical user interface and performs a similar function for a set of users, cf. www.ebay.com.

The WSDM approach is mainly designed to work with "kiosk"-type of websites. The design issue of kiosk-websites is to structure the information in a clear and accessible way for the visitor. If the information is not well structured from the start, sooner or later maintenance problems will occur. These problems are easily compared to problems that occur in relational databases - namely redundancy, inconsistency and incompleteness.

These problems in the next step lead to usability problems experienced by the users of the web site. If for example there are information redundancies the user will get irritated on getting the same information on several pages. If there are inconsistency problems the user will probably distrust the information and even the whole web site. Finally incompleteness may also lead to distrust and having the user leave the web site due to broken links or not finding the information the user thought he would find on the website.

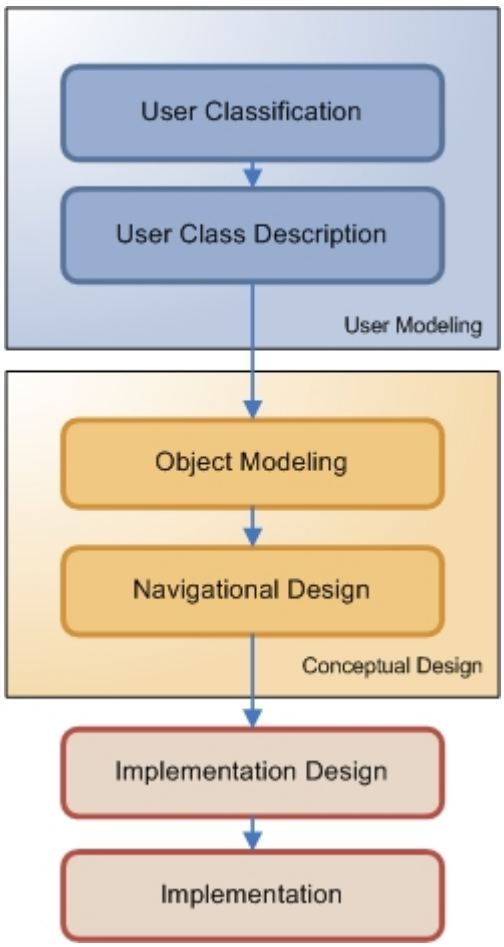
Two other important but often neglected aspects of web sites are actuality and visitor-types. A website lacking actuality will lower the users' confidence in the company as well as in the information presented on the website. The other neglected fact is that a web-site does have many different types of visitors which all request different types of information. A company's website will probably be visited by potential customers, customers, suppliers, company employees and stock-holders. These groups all visit the website with different purposes which all need to be satisfied and this should be reflected on the website.

The method

Although the method is user-centered it should be noted that this is not the same as user-driven. In the latter the requirements are normally captured by interviews and scenario analysis - this is not possible due to the vast variety of users that will use a web site. The main goal of the web site is to offer information and the success lies in whether it is possible to do so in such a way that both the provider and the inquirer benefits from it.

The method consists of four main phases: User Modeling, Conceptual Design, Implementation Design and the actual Implementation[35]. These are related to the other software engineering phases as parts during the design and implementation phase - the other phases such as

requirement, maintenance, evolution and innovation are not covered by the WSDM method. The method assumes that mission statement - the websites purpose for the organization already has been formulated. The figure below shows an overview of the WSDM phases.



The four phases of WSDM - User Modelling, Conceptual Design, Implementation Design and the actual Implementation. In the figure the phase User Modelling is also subdivided into two phases User Classification and User Class Description and the Conceptual Design is decomposed into Object Modelling and Navigational Design.

User modeling

When a user usually visits a web site it is with questions in mind. The website should anticipate the users' questions and try to answer them. That is the reason why WSDM's first phase concentrates on the potential users of the website. During the User Classification the potential users of the web site are identified and classified. One way of performing this classification is to analyze the mission statement and then divide the organization or process into a number of activities. Each activity involves people and these people are the potential users of the website. The result of the phase is a schema representing the activities and parties. It should be noted that a user may be present in several user classes - i.e. user classes need not to be disjoint.

The next sub phase in the User Modeling phase is the User Class Description where the user classes are analyzed in more detail. The analysis is performed with two different focuses, one focus on the information requirements and the other on characteristics. The assumption made is that all users from a similar class will have approximately the same requirements. Namely, the users may differ on how they want to have the information presented - e.g. a foreign visitor may want to have the information presented in his own language. Therefore also the characteristics of the user classes are identified. The information requirements can be seen as the answer on the conceptual question "what", and the characteristics answers the conceptual questions "who and how?". Furthermore, if within one user class, users have different characteristics which result in different usability requirements - the user class is divided into so called perspectives[35].

Conceptual Design

The Conceptual Design phase consists of two sub phases, the Object Modeling and the Navigational Design. During the Object Modeling the information requirements of the different users and user classes and their perspectives are formally described. During the second sub phase, the navigational design, the navigation of different users is also described formally.

The formal specification of the Object Modeling Phase is a model called User Object Model (UOM). The UOM is a model of the business objects in the organization from the viewpoint of the users in the user class description. The UOM takes into consideration the fact that a user only has limited knowledge of the organization and thus the UOM will only cover a part of the total set of business' objectives. The UOM model can very well be described by traditional ER-models

which describe different types of objects, relations between these objects and or constraints on the same objects and relationships.

During the navigational design phase a conceptual navigation model is constructed. The Navigational Model consists of a number of so-called navigational tracks which each user group is supposed to follow when looking for information.

Implementation Design

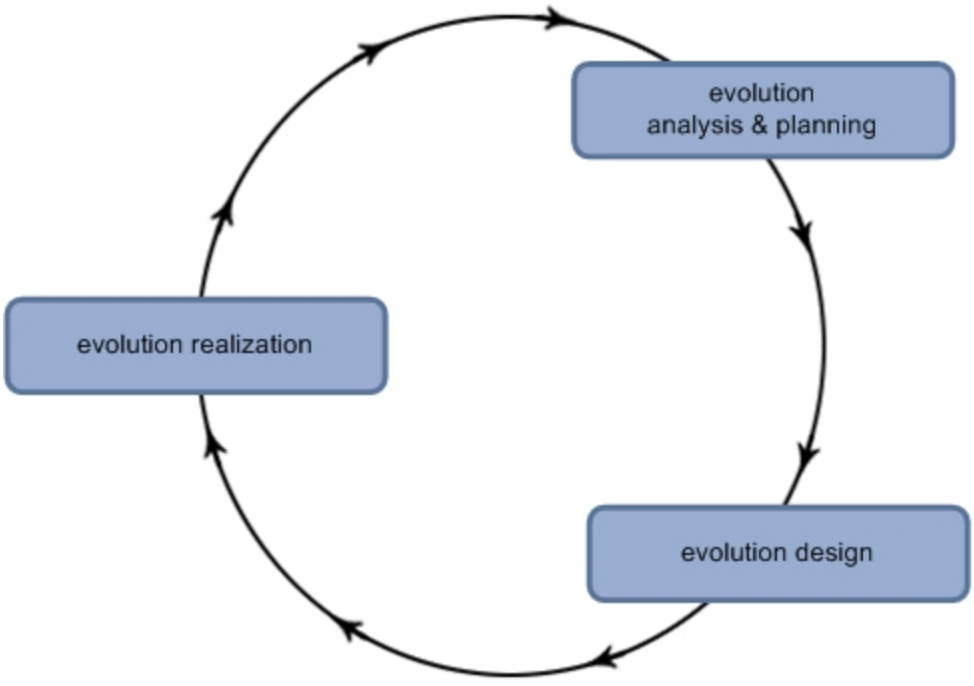
In the fourth phase the web sites "look and feel" is designed. The main goal of this phase is to produce a consistent, pleasing and efficient look and feel for the conceptual design made in the previous phase. The result of the implementation design is an implementation model.

The actual implementation

During the last phase the actual implementation is performed. That is, the implementation model is translated into e.g. a set of HTML files which will form the web-site or a set of dynamic script-files which for instance collect information from a repository such as a database.

A.9 WebComposition Process Model (WCPM)

The last model to be described is the WebComposition model. It is a process model that consists of several phases - all derived from modern object-oriented process models as well as solutions addressing the need of software reuse[24]. The process model is of spiral and evolutionary type consisting of evolution analysis and planning evolution design and evolution realization, cf. figure below.

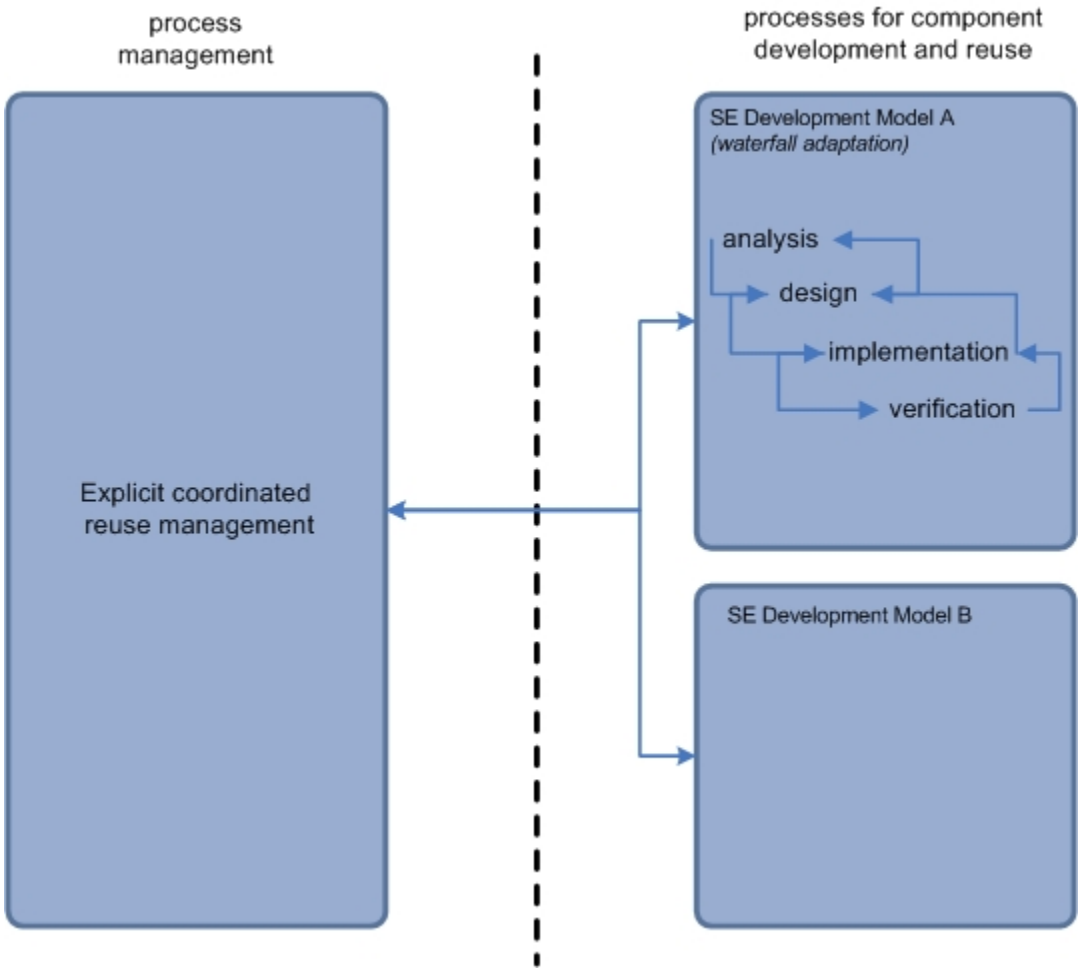


Evolution spiral of the WebComposition Process Model. As the figure shows, building and maintaining an application is an endless circle of evolution analysis and planning, design and realization. However as the next section points out an evolution cycle does not just consist of just one heterogeneous process[24,36,40].

It should be noted that the WebComposition process-model is one third of the WebComposition approach which also consists of a component technology and reuse management. The approach is based on component-oriented model of Web software[37]. The component model is based upon the assumption that all components can be described in a standardized way and contains any sort of artifacts such as HTML-code, script-code or layout-elements[37].

One of the most important properties of the WebComposition Model is that it is an open process model, i.e. that inside the evolutionary spiral model several other orthogonal processes following other approaches can exist[24]. The WCPM model does not define a specific development process but it can rather be seen as framework in which arbitrary development processes like OOHDM and RMM can be integrated. Thus it is a model which can be implemented and is suitable for a large variety of applications.

As an example two development teams working side-by-side can favor different approaches such as eXtremeProgramming (XP) and Waterfall-approach and still have it to subside under the WebComposition model. This is possible thanks to the generalization of artifacts which applies to storage and management of artifacts.



Coordination of orthogonal processes in the WebComposition Model with an waterfall adaptation in the software development model A.

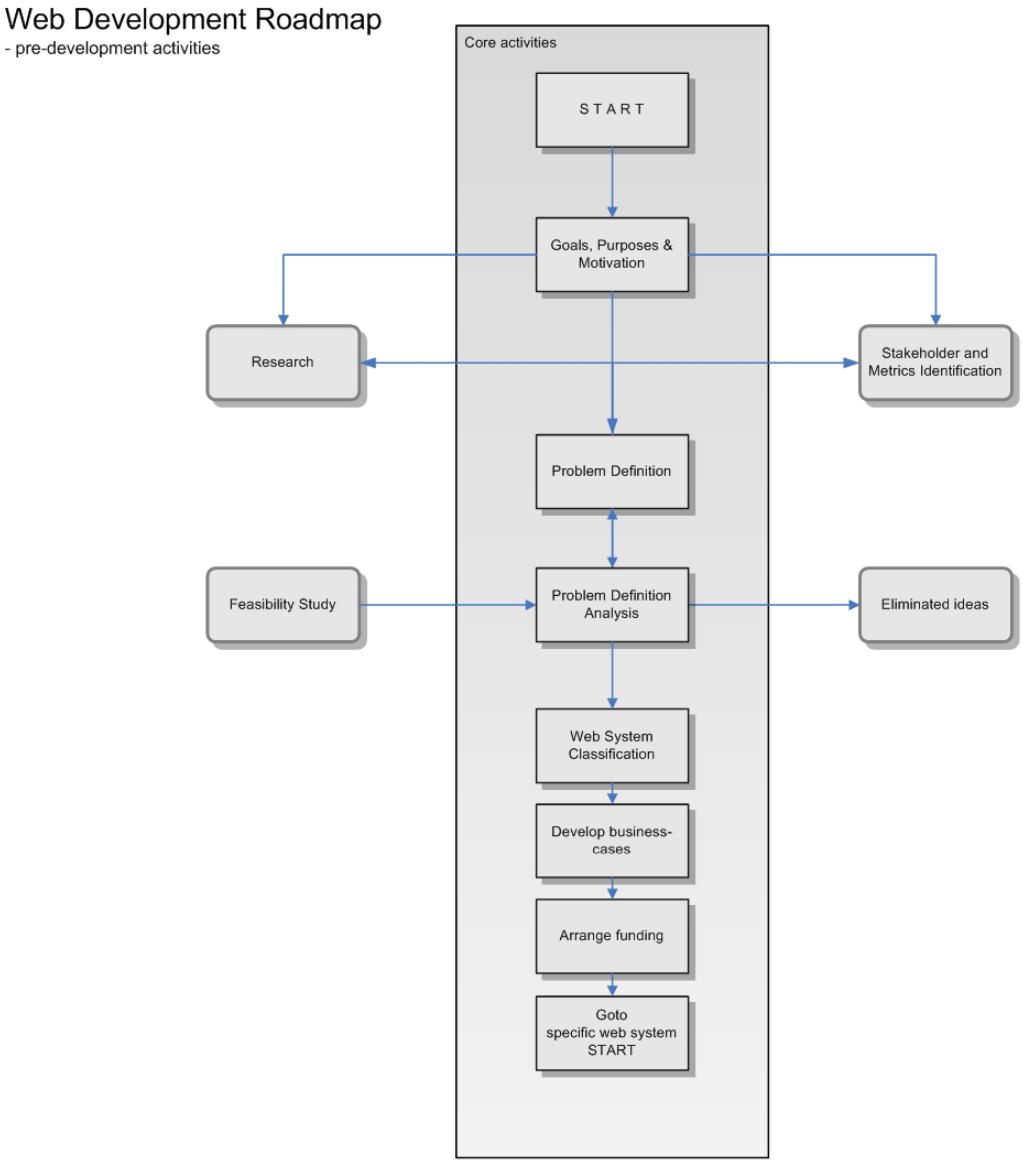
The mixed use of different process models is made possible thanks to an explicit and coordinated reuse management[38]. This means that all models to some extent must be modified to fit the reuse management and this can be done by e.g. generalization[39]. This may seem as an inconvenience

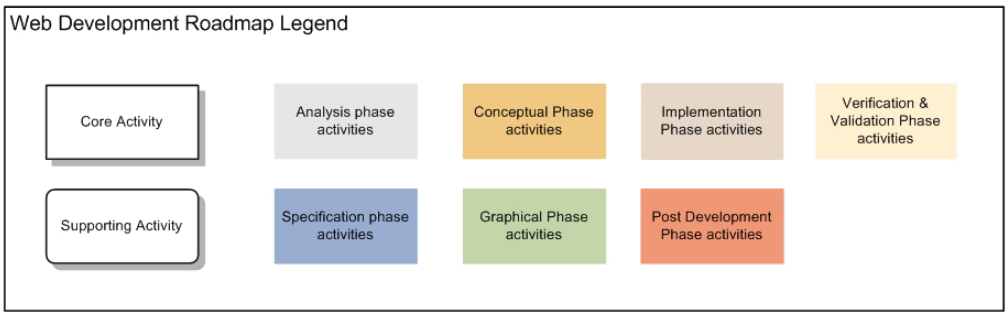
but since it only applies to the storage and management of artifacts it does not matter since they are a part of the application of a process model[24].

The WCPM approach defines an object-oriented model that uses a XML-based descriptions for each Web entity such as individual links, layout fragments, images or even complete HTML-pages. These XML-based descriptions are made with the WebComposition Markup Language (WCML). Thus the WCML documents can be seen as a kind of component stores[37]. The actual web resources are then later, during the development, mapped into e.g. HTML-documents by a WCML-parser and compiler.

Through the use of WCML object-oriented abstractions and reuse is made possible, thus making WCML applications modifiable and extensible and at the same time being in the concordance with the basic principles of the web thanks to the XML language. The WebComposition Process model allows for the evolution of a Web application to be planned and based on the concept of domain engineering. This allows for a seamless evolution and maintenance.

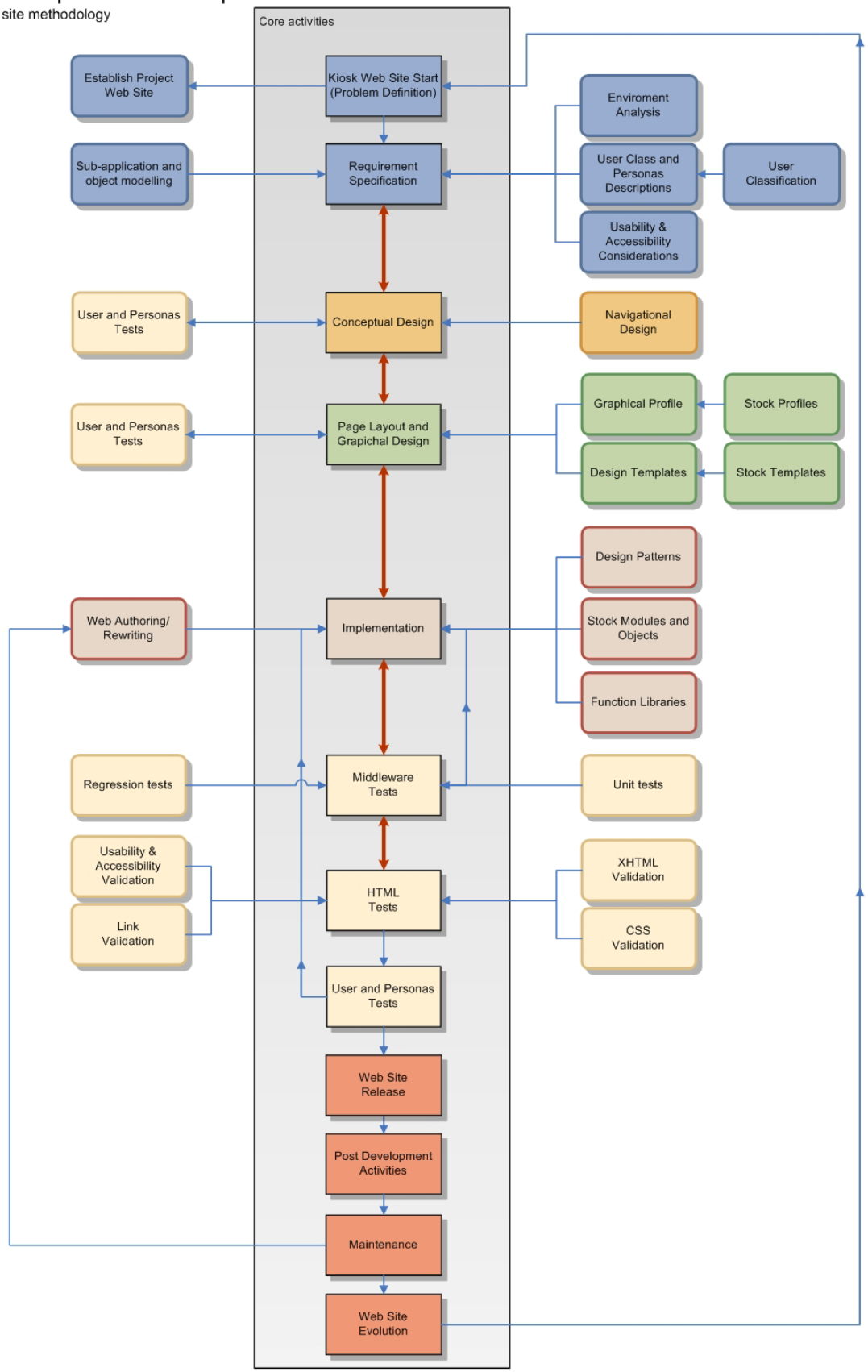
Appendix B – Web development roadmap





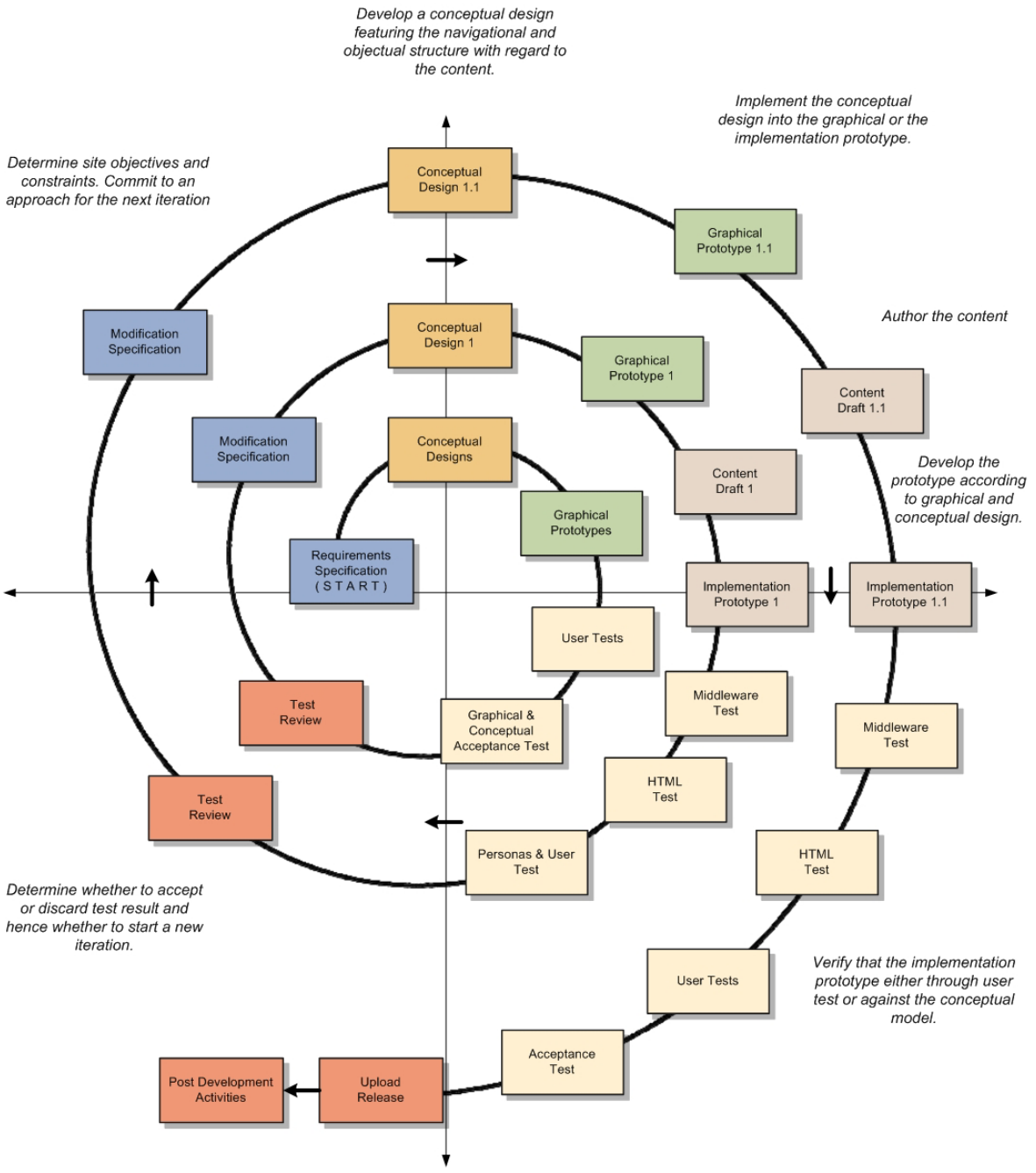
Web Development Roadmap

- kiosk web site methodology



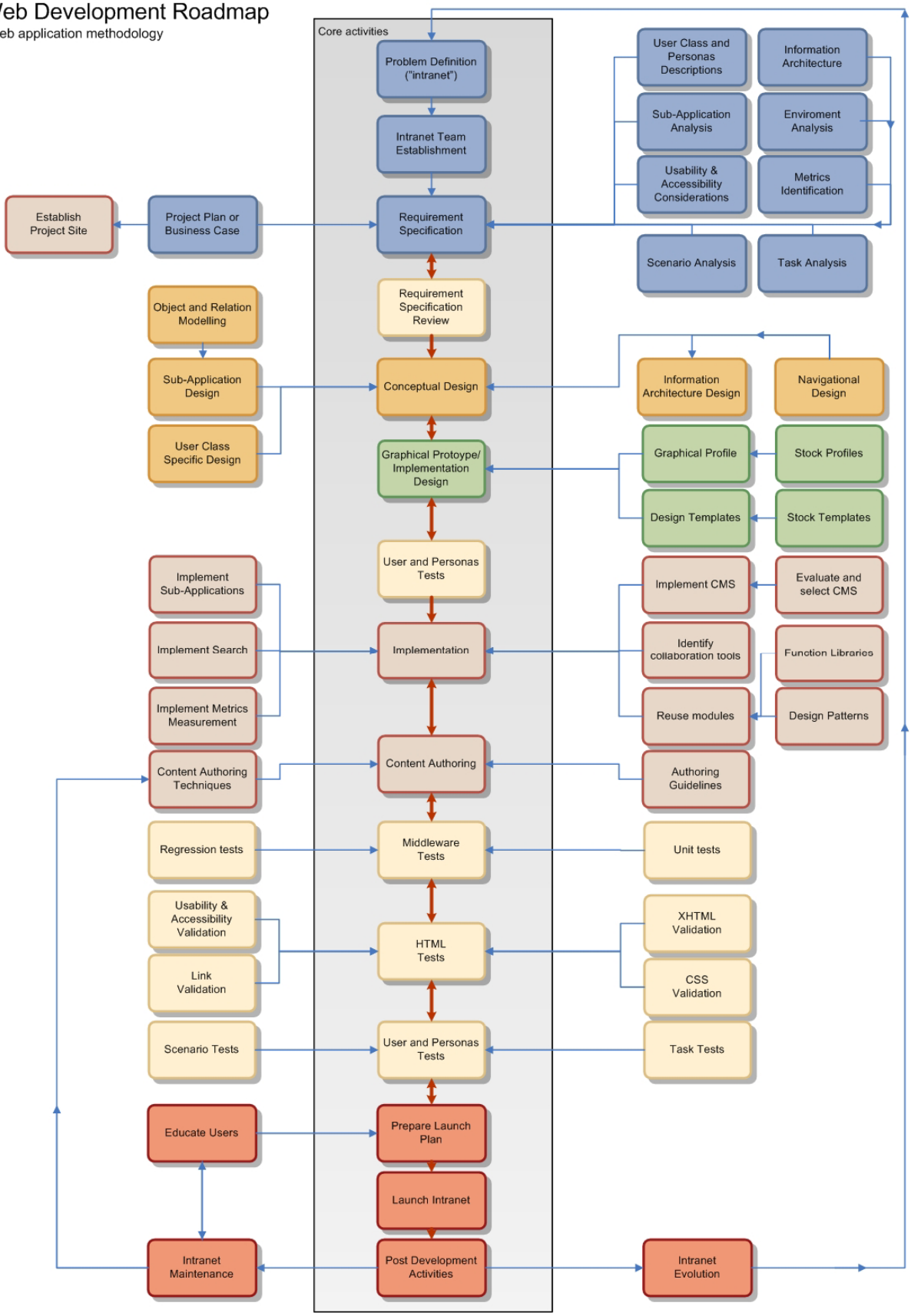
Web Development Roadmap

- kiosk web site methodology spiral representation



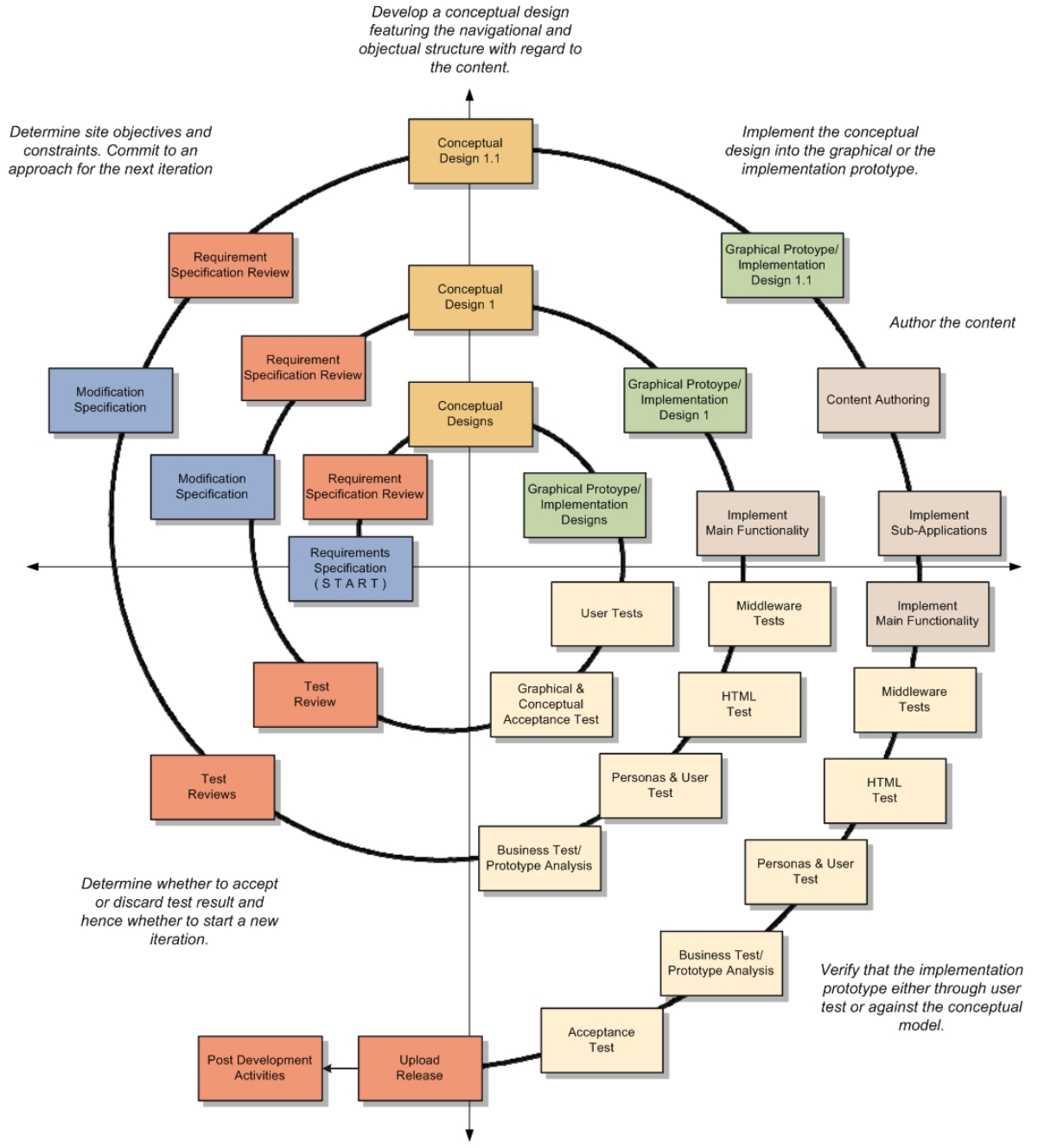
Web Development Roadmap

- web application methodology



Web Development Roadmap

- web application methodology spiral representation



Appendix C – Web development roadmap activities

The different roadmap process activities are described below in alphabetical order.

Arrange funding

The last, but most important, pre-development activity should be to arrange funding for the project. The funding should not only be established for the actual development or purchase but also for the maintenance of the system.

When developing an internet web site experience shows that only one funding-model is applicable and that is to provide the budget out of the overall company funds. But for intranets and extranets Step Two Design proposes two additional funding models.

1. asking business units to fund the publication of their content
2. instituting an intranet levy, that charges business units a set fee for each staff person with access to the intranet.

A mix of these approaches can also be taken – such as funding the initial development with company funds and then switching to any of the other models when the development is finished. The most important task in this phase is to plan beyond the initial development and to ensure that the system is able to be sustained and further developed.

Conceptual Design

The next core activity following the requirement specification is the conceptual design phase. The conceptual design phase includes using the results from the requirements specification and problem definition to determine the information architecture of the system. The information architecture should include a conceptual design of the overall site structure, structure of individual sub-sections, overall navigation methods as well as cross-linking between sections.

It is very important for the conceptual design to be business-oriented and focus on the most wanted response – both from the users and stake-holders point of view. In order to be able to focus on e.g. what the different stakeholders request some of the following techniques can be used.

- Card-sorting techniques. These are the quickest, simplest and most cost-effective techniques which allow the users to sort a series of cards, each labeled with a piece of information of web system content or functionality, into groups which make sense to them.
- Collaborative design techniques. Collaborative or participative design techniques have the aim of letting different stake-holders such as staff, developers and business representatives, work together to form and generate ideas for the conceptual design. If it is possible several independent teams can work together and their results can be combined in the end.
- Contextual inquiry is a type of stakeholder interview that is conducted at the user's place of work or place where he or she will be using the system. It combines overviewing the user to complete tasks, with the opportunity to learn about what they are doing. During the interview, the artifacts used by the user can be collected. Artifacts such as forms and cheat notes can serve as invaluable information for the design of the system.

The outcome of this phase should be conceptual sketchings or documents illustrating the overall site structure. Experience shows that the best way to illustrate the structure is through organizational charts or schemas.

CSS Validation

Similarly to the XHTML-validation, the CSS validation can be carried through with the help of W3 validator for CSS which can be found at: http://jigsaw.w3.org/css-validator/.

Design Patterns

A design pattern may be a document that describes a specific design problem. A typical design pattern describes the problem, the recommended solution, the rationale behind solution and any related or alternative patterns to be considered. Design patterns also proved a number of advantages over more traditional templates and solutions such as:

- Design patterns can be mixed and matched to address unique design problems
- Design patterns empower the developers and architects to discuss the rationale behind the recommended solution
- Provides alternative approaches

Design Templates

Since almost all web based systems contain the same type of pages a certain degree of reuse should be applicable. Just as with the graphical stock profiles, there are companies specializing in creating page templates which can be bought.

Develop business cases

Depending on the project management practices used and the funding organization, one or several business cases may be required. The overall goal of the business case is to give an idea of how the organization will benefit from the web based system to come. In contrast to the project plan the business case should also contain more intangible benefits such as increased staff-moral, work motivation and higher customer satisfaction etc. The business case should demonstrate the organization's change in the way of working[61].

Experience shows that the business case's in general should contain the proposed activities, potentially extending the initial project plan into a "traditional" software engineering project-plan. It should contain the project's projected time-frame, the funding and staff and technology resources required. But since in most cases the business case act as a support for the company board it should focus on the benefits of the project as wall as the costs and risks of not proceeding with the project. In most cases it has proven to be most advantageous of letting marketing, or other non-technical personnel, writing the business-cases since engineers often tend to focus on technical issues.

Environment analysis

It is important to decide on the technical environment to be used for the web site. In short it can be said that any web based system does need four general and technical components in order to work.

1. It needs an operating system on which all the other components run.
2. The second component needed is a web server. The web server is the program that takes request for pages from a browser, interprets the request, and returns the results. For static HTML pages, it simply retrieves the HTML file that the browser requests. For dynamic pages, when a browser requests a page, the web server transfers control to a program or module that interprets the script and returns the results.
3. The third component to be considered is the database. The database stores information that can be retrieved, stored, and manipulated by the scripting language. For web-sites it

can be used to store content (i.e. articles, news stories, calendar events) and user information (i.e. preferences, permissions, contact information).

4. The fourth and last component needed is a script engine. When a browser requests a script, the web server transfers control to a script engine which reads a program written in a scripting language and returns the result in the form of HTML.

Currently there are two main competing web platforms with slightly different properties.

1. LAMP – Linux, Apache, MySQL and PHP. This technical environment is today the most popular and fastest growing web platform. It is completely based on open-source products and hence it is almost free of charge. The advantages of using this option are several; it is cheap, it enjoys a wide-spread support by the community, it is well acknowledged, it is very mature and allows for extensive reuse.
2. WISA – Windows, Microsoft ISS, Microsoft SQL Server and ASP. This is the technical environment proposed by Microsoft. In contrast to LAMP, WISA is expensive to acquire, corporate and very well-documented and well-supported by Microsoft.

Experience shows that LAMP is to be recommended for most web sites because of its cost, PHP's simple integration and community support and Apache's security. On the other hand WISA is to be recommended for the enterprise-level sites because of the SQL Servers very advanced functionality and Microsoft's support services and economic stability. There are other excellent operating systems, web servers, databases and scripting engines but in most cases these are not to be recommended.

Establish project web site

The development should start with establishing a temporary project web site that contains all the information on the project at the same place. It has the advantage that the information and deliverables may be shared among the teams and people involved as well as communicate the progress of the development to the rest of the organization.

The project site should be small and Step Two Design offers following guidelines for the project site[61]:

1. The project site should be kept small, manageable and practical
2. It should be ensured that the setup and maintenance of the site does not divert the team away from the real task at hand of designing the new system.
3. The team members should be encouraged to take responsibility for keeping the project site up-to-date, and use the site as a mechanism for fostering team collaboration and knowledge sharing.

Goals, Purpose and Motivation

Software applications are generally built to fulfill a specific purpose, solve a problem or make handling of something more effective. But for example, the majority of web based systems – the kiosk web sites - lack a specific purpose. Many companies can be said to be on the net without in fact being there. Companies have web sites or web presences for various reasons, maybe just because the competitors has one, because the company board thinks the company needs one or just because of the overwhelming fear of being left behind in the web revolution.

Company customers are directed to web sites which are not backed up by any off-line activities, which are outdated or even lack information. The view of the Internet has been similar to the view of a new Klondike where money is generated by itself without any more work then the initial. Unfortunately this is a false believe, which has had the consequences that many websites has decayed over time. The reason is simple. When the company website does not generate profit

- the company loses interest in the web site - and finally the web site is abandoned. When the company web site is abandoned the presumptive customers may not see the company as serious or active and the company loses customers and hence more money. The company is caught in a negative spiral.

Fortunately this trend is easily changed. If the web site's purpose is clearly defined it is easier to measure immediate profits, when the web site generates profit the company gets more interested in it and hence the web site becomes important for the business. Having a well-planned and well-developed web site may have several advantages for a company.

Some advantages of having a good web site with a clearly defined purpose

- The website positions the company against its competitors.
- The website gives the company's presumptive customers possibility to get information on the company in advance.
- The website improves a company's reputation, phone contacts and sales promotions.
- The website gives credibility to the company.
- The website may show the company's earlier projects and customers.
- The website may answer customer questions.
- The website can be used to increase company sale
- The website can help lower print-costs, since material can be published and distributed on it.
- The website can help with employee-recruitment.
- The website makes it easier to receive customer feedback
- The website is online and open 24/7, 365 days a year
- The website may serve as a information source for resellers
- The website can help distributing a newsletter

In order to achieve the advantages listed above the purpose and the goals for the web site must be clearly defined. Developing a problem definition for a website does not need to be difficult and there are a variety of questions, some of them listed below, that can help stimulating ideas and help developing the definition.

If a website already exists and the project is a modification or upgrade of the site, the history of the current web site should be taken into account. While the problem, stated when the website was developed, may have changed over time, it may still be a good-starting point for the new upcoming project. Below follows some of the questions that should be asked before developing a web site.

Questions to be asked before starting a website development

- Generally all the "Why" questions should be asked.
- What is the main purpose of the site?
- What is the main motivation for the site?
- Why is the site needed?
- Who is the site intended for?
- How will the online-presence support the real-life work?
- What will the web-site bring the Internet?
- What will the Internet bring the company/organization/person?
- How will the Internet's possibilities help the company/organization/person?

- How will the website increase the company's/organization's/person's profit?
- What is the goal with the website and when and how will it be evaluated?
- What does differentiate the website from existing and similar websites?

The secret behind a good web based system is to think of the profits and revenues in advance. The idea with the website must always be to solve a problem that cannot be solved in any other way and to get some profits out of it. The profits does not have to be in economic terms but it can be in goodwill, reputation, saved time, improved service, better first impression etc. When this goal is defined the first step towards developing a successful website is taken.

This section has so far only discussed the goals and purposes of web sites. This part is equally important when creating intranets, extranets or any other web based systems. The main difference is often that the users of the system are easier identified than with an online website. Hence the needs analysis is often easier conducted and carried through.

The needs analysis should identify the objectives of the web based system, such as reducing cost and improving organizational efficiency. It should also regard the business related issues that are faced by the company employees. It may include cultural issues, training issues, knowledge sharing and business process bottlenecks.[61]

The needs analysis should also identify the high-value content and the functionality to support the day-to-day activities of the staff as well as key strategic initiatives[61]. Next to the functionality it is important to identify the high-level information architecture, usability and content issues. It is very important to conduct the needs analysis throughout, since it is the only way to ensure that the system will meet both the needs of the staff and the organizations.

Graphical Profile and Stock Profiles

It is very important for the web based system to follow prevalent corporate graphical profiles in order to maintain and increase the systems value within the organization. It is even more important when it is concerning a system which will be used by people outside the organization.

At the same time it is important to regard the usability and accessibility requirements specified during the initial development phases. In order to shorten time in this phase there are several companies and web sites that offer stock (non-unique) profiles at a relatively low cost.

HTML tests

When the middleware tests are finished and the html-output is generated it is possible to test and validate the output that reaches the end-user.

Identified purposes, goals and metrics – the project plan

When the feasibility study and the problem definition analysis have been conducted, a list of goals should be summarized from the identified purposes. It is very important that the goals set up represent outcomes that are measurable. Goals should be formulated in way such as "improving sales" or "reducing time spent" rather then "providing a central information repository".

The reason behind the special goal formulation is that the only way to assess the web systems' rate of success is by metrics measurement. A goal formulation in the form of measurable outcomes allows and prepares for such an assessment. The web systems should therefore contain methods that allow for the specified metrics collection.

Another activity that should be carried out during this phase is to establish the web system ownership within the organization. This is as important for web based systems such as web sites as it is for more complex web based systems such as intranets. The owner of an intranet should be a particular business unit or department who are ready to be committed to the development of the system. At the same time a team of people, which may be the members of the team owning the system, should be assembled in order to maintain the continuity of knowledge and experience – i.e. a permanent role which is still active after the deployment of the system.

In addition to establishing the ownership an appropriate project sponsor should also be selected. The primary role of the sponsor is to provide high-level strategic direction for the project[61]. He will among other things also be responsible for developing the system as an integral key business tool and resolve conflicts among stakeholders[61].

The findings of this phase may be summarized into an actual artifact such as an initial project plan – this may be recommended for large development projects but it is not as necessary for a smaller kiosk-website project which only involves a handful of people.

Implementation

When the conceptual design and the page layout phase have been carried through the actual implementation phase can start. In many ways the implementation of web based system, especially web systems back-end implementation, does not differ from "traditional" software engineering implementation phases. Hence many of the activities in these methodologies are directly transferable to this phase. For example the activities and ideas in XP and FDD are very well suited, especially ideas such as on-site customer, pair programming and object oriented reuse.

The outcome of this phase should be a development web site which can used, tested and verified against the requirements and the problem definition.

Kiosk Web Site Start (Problem Definition)

When all the pre-development activities have been completed the actual web system development process starts. The findings in the initial project plan or problem definition serve as the basis needed for the process to start.

If the activity is entered after a development has been completed, i.e. in the web site evolution phase it is recommended that the initial pre-development activities are carried through.

Link Validation

Link validation testing involves activities to find the number of non-functional links within the system. The easiest way to identify broken links is to use the W3 validator at http://validator.w3.org.

Middleware Testing

As mentioned in the environment analysis phase, most web based system makes use of some kind of a scripting engine. The scripting languages are commonly called middleware since they generate the content which the server sends to the user.

In general the middleware consists of programming languages similar to traditional application programming languages. Hence it is possible to test these in pretty much the same as normal with unit and regression tests.

Navigational Design

The most important thing to be considered during the information structuring is the users' or visitors' limited knowledge of the internal structure of the information or organization. Hence the need for conducting initial user and personas test in order to ensure that the navigational model is and remains user-focused.

Furthermore it is most likely that the web system is supposed to include a wide range of functions next to the content. This may include, for example, user registration, discussion forum, news system or other business-focused applications. Once the overall information architecture has been determined the applications to be included should also be specified and considered. The final design of the application should be completed once the page layouts in the graphical design phase are finished.

Page Layout and Graphical Design

When the conceptual design is approved by the user and personas tests, the following activity is the graphical prototype design phase. During this phase a graphical representation should be done based on the results of the conceptual design.

Experience shows that the majority of web based systems include approximately five different page types which need to be layouted.

1. The start (home) page for the users. This page is supposed to serve as portal or gateway to the rest of the system and hence these first pages often contain a lot of different information.
2. The sub-category home-pages, even called transitional pages. These pages serve as first pages for any given sub-category and their function is similar to the home-page – they should serve as a portal to the rest of the category.
3. The typical content page. This is the page lowest in the information architecture and it hopefully contains the information the user requests.
4. The search result page which presents the results of a search conducted by the user. In most cases the easiest and fastest way to find information is to search for it – hence the search result page will be one of the systems most used and therefore it requires special consideration.

It should be noted that if a content management system (CMS) is to be used, the system's constraints on page layout should be identified and regarded.

When the major page layouts have been done, it is time to cloth them in a graphical design. This includes the general look-and-feel for the system as well as the specific graphical designs for all page layouts. The outcome of this phase should be different images which later can serve as templates for the developers who are supposed to implement them to the web environment.

Post Development Activities and Maintenance

After the web site has been published on the Internet the post development activities take place. The post development activities involve a range of activities such as search engine registration, metrics collection and web site marketing.

The largest and most time-consuming post-development activity is maintenance. It is extremely important to keep the web site or web system alive and up-to-date in order to keep visitors trust and have them returning to the web site. Different kinds of surveys may be used in order to improve the quality of the web site.

Requirements specification

The problem definition created in the activities described in section 4.4 serve as the starting point for the requirements specification. Functional as well as non-functional requirements should be identified and specified. The requirements analysis and specification is an iterative process which involves domain understanding, requirements collection, classification and prioritization. This can easily become a very tiresome process in which it is very easy to get stuck, since it may involve widely separated issues such as content authoring and server environment decisions.

Fortunately there are some short-cuts that can be taken in order to reduce time spent in this phase. Since most web development projects are very alike requirements from one project are often easily transferable into one other. If the organization does not have the experience required there are companies, who actually sell finished and review requirements documents.

The following four sections explain more in detail what areas should especially be considered when conducting a kiosk-website development project.

Problem Definition

The results of the research, stakeholder input and answers to questions mentioned are combined into a problem definition. The problem definition may be an actual artifact such as a paper but it is not a necessity.

In most cases the problem definition may be discussed during a meeting or through email communication. It should be noted that the problem definition is not a mission statement and that it should be written in user centered language. The reason for it being written in the users' language is because the problem definition's main purpose of existence is to serve as support for the management when they are deciding whether to pursue the project or not. Below follows an example of what a Problem Definition may look like.

Ivtaco.com Problem Exploration

Summary: This document summarizes some the current problems that are facing this company. It is believed that a web site may be the solution to some of these problems.

- The company is a Web Development Company and in order to increase our credibility our site should follow the latest standards and promote our in-house knowledge. The current web site does no such thing.
- The company is spending a great deal of money on sending out technical papers through mail. The current cost for printing and shipping a technical document is approximately $5 per copy.
- Our list of offered courses is constantly requested by potential customers. A great deal of time is spent on keeping the list up-to-date.
- Investors and other parties have expressed an interest in obtaining more timely information.
- The current web site does not reflect all our business areas.
- The current web site does not reflect our current graphical profile.

Conclusion: It is believed that a re-work of the current web site will address many of our current problems. For example, a new, re-worked web site may be able to reach our foreign customers better than the traditional means and the cost savings by providing most product data sheets online may be significant compared to traditional methods.

In most cases the problem definitions and answers to the question in are very similar. The time spent in these initial phases can many times be reduced since many of the problems are reoccurring; the list below mentions some of these problems for kiosk-type websites.

Some frequently reoccurring kiosk-website problems
• The company does not have a web site and does want to inform the world of its existence. • The company developed a web site some time ago and have realized that they are not able to update the information. They want to be able to administer the web site's information themselves in an easy way. • The company's graphical profile has changed and they want the web site to reflect the changes. • The company has had a website for quite some time and want to add additional functionality – e.g. event calendar, discussion forum or news system. • The company does have to change or upgrade the website due to obsolete web pages. I.e. the web pages were created a couple of years ago and modern browsers do not longer support them. • The company wants to change web hosting provider.

If the problems identified in previous process iterations are saved they can be used in new projects in order to cut the time spent on identifying problems. Unfortunately the same is not as easily done with more advanced web applications since these may solve vastly different problems.

Problem Definition Analysis, Feasibility Study and Eliminated ideas

When the problems are defined the next step is to analyze them. During the analysis it is decided whether all the problems in the problem definition can be solved by a web based system and if it can be done at a reasonable cost – a feasibility study is conducted.

If the example in the previous section is followed, it may for example be considered too expensive to transform the old code-base into a new more standard-compliant code-base – especially if the old one is still working.

But the problems, whose solutions are not considered reasonable, should not just be discarded but instead saved for future reuse.

Stakeholder input and Research

One very common error done during web site development is to proceed from the internal organization of the company and the knowledge within it. That is, the information the company thinks the visitors request may not coincide and vice versa. In order to avoid this confusion, a quick stakeholder analysis should be conducted, in which the main visitor-type should be identified and in the best of worlds interviewed on what information he would like to have on the website. Unfortunately few projects do have the kind of budget that allows such interviews.

Research may also be conducted in order to find out what the competitors or similar web sites offer. An analysis of other sites may very well help improving the one upcoming as well as at the same time give valuable knowledge of e.g. will it be profitable for us to do the same thing; how should we differ from our competitors, what problems have the other web systems suffered from.

Stock Modules and Objects and Function Libraries

There are today many companies that specialize in selling general components that can be used in web based systems – examples of such sub-applications can be calendars, news systems or editors. At the same time there are communities which develop modules which are free to use as long as the source is kept public.

Implementing such modules can shorten development time drastically but experience shows that one should be aware of the potential pit-falls of such approach. Since many of the open-source modules are developed by enthusiasts on their spare-time, many times the modules are not optimally or well developed, documented or supported.

Usability and Accessibility Considerations

If usability- and accessibility considerations are made early within the development projects it is ensured that the needs of the users with disabilities, such as visual impairment, dyslexia and mobility impairments are satisfied. Although usability may seem to be the same thing as accessibility there are some major differences. Usability is the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system. While the common definition of accessibility is the degree to which the software system protects system functions or service from being denied to the user.

There are different ways on how to ensure both accessibility and usability. The W3 Consortium has developed a comprehensive range of web accessibility resources, include web content accessibility guidelines and checklist, which can be found at www.w3.org/wai. Experience shows that it is recommended to follow these guidelines since they also increase the overall usability of the web system, i.e. the changes made to satisfy the needs of visually impaired people also be advantage to the normal users of the system.

Step Two Design proposes the following research methods for investigating the needs of the users[61]:

- Interviews with the staff with accessibility needs.
- Observation with accessibility needs using the current web system or a range of similar web sites.
- Research of accessibility standards and legislation.
- Investigation of assistive technologies.

Usability and Accessibility Validation

The usability and accessibility of the outputted webpage is validated against the requirements specified and this can be automated to some extent by using validators such as Bobby (http://bobby.watchfire.com) or WAI validator (http://www.contentquality.com).

User Class and Personas Descriptions

Since it for most web sites is very difficult to identify and classify the potential users and visitors often so-called personas are used. Personas are fictitious users that represent the needs of larger groups of users, in terms of their goals and personal characteristics. They act as 'stand-ins' for real users and help guide decisions about functionality and design[62]. Personas identify the user motivations, expectations and goals responsible for driving online behavior, and bring users to life by giving them names, personalities and often a photo[62].

These actions enable the developers and other stakeholders to stand in their user's shoes. The main aim is to get the developers to shift focus from their own and senior executives' ideas to the actual users' goals. The other advantages are that the needs of many different users are projected

onto a manageable set of personas that represent the needs of many. It is furthermore relatively easy to develop a persona and thus shorten the time spent on gathering user requirements. And finally disagreements over design decisions can be sorted out by referring to a personas best interest as well as the frequency of large and expensive usability test with real persons can be seriously reduced[61,62].

User and Personas Tests

In order to ensure the quality of the conceptual design, including the navigational design, it is very important to test and evaluate it. Experience shows that the best way to ensure that the conceptual model is user-focused is to test on potential users and to test by the use of the set of personas.

Exactly as with the conceptual design it is very important that the graphical design and the page layouts are accepted by the future users of the system rather then by the designer. The best way to ensure this is by continuously and frequently asks users and evaluate it with the help of personas.

When the final HTML tests are done it is time to evaluate whether the prototype or finished product satisfy the users' and personas' needs. Users may be monitored in order to get knowledge on how the system performs and if there have been any logical errors that should be corrected.

Web Authoring / Rewriting

The last but almost most important implementation activity is the Web Authoring / Rewriting activity. Writing for the web is slightly different from writing other material and therefore certain guidelines for authoring should be implemented. Experience shows that these guidelines should include among other things[61]:

- Usage of the standard templates
- Tips for a good writing style
- Guidelines for structuring pages
- Linking conventions
- Techniques for identifying the most appropriate content for the audience
- Benefits to the end users of a consistent style

It is important that the guidelines save the authors' time and help them to do their job well, it is not supposed to be a list of "must"-rules. The time it takes to rewrite material to online form should not be underestimated. Experience also shows that for larger documents, such as policies and procedures manuals, it should be considered of hiring a professional technical writer who is skilled in converting large paper documents to online material.

The company UsersView[59] offers some guidelines on web authoring:

- The most important message should be written first. The average visitor only reads the first three words in each section and does in general only read the first sections. Hence keyword should be present within the three first words.
- Use short sections but avoid one-sentence-sections.
- Use a reasonable number of headings and sub-headings
- Make use of lists but do not write everything in lists.

- Make keywords bold or italic.
- Write short and concise sections.
- Write texts objectively – users tend to disregard from obviously commercial sections and there is a risk for the message to be ignored.

Web Site Evolution

Over time it is most likely that the web site will grow and additional functionality will be needed. Depending on the nature of the additional functionality the process begins a new life-cycle. New content restarts the life-cycle from the implementation phase whereas a new functionality or sub-application usually needs the whole life-cycle to be re-run.

Web Site Release

When all the test results are considered acceptable the web site is moved from the development environment onto the real server and made online. At the same time the old web site is backed up in order to be able to re-roll the old web site if something unexpected happens.

Web system classification

When the previous phase, in which the purposes, goals and metrics have been defined, is complete it is time for the web system classification. This is more a process activity than a real world activity – since it only determines what type of a web based system is to be developed and thus determined what sub-branch of the process should be used.

In general there are four kinds of web based systems, but for make matters worse they are in general closely related and inseparable. The first depicted in the process schema is the Intranet, the intranet is the local web of a organization – it's main purpose is to distribute information among the employees and make the organization more effective. The requirements of a Intranet often differs from the requirements of a Internet web system since the stakeholders are often clearly defined, it is a system that is used on daily basis and the users become quite experienced in the system after a while, the functionality is more important than look-and-feel and for example security considerations do not have to be as harsh since often the intranet is separated from the outside world.

The next two classes the two Internet-based web systems – the kiosk-website and the application web site. The kiosk-website's main purpose is to distribute information to its visitor - it is from its visitor's point of view a collection of pages with certain requestable information. A great example of a kiosk-website is www.cnn.com which presents its visitor news from all over the world. The next types of internet web based systems are the web-application web sites, which present all their visitors a similar functionality. An example of a web-application web-site is any online bank. The online-bank offers its customers the same functionality of transferring money, buying funds or e.g. taking loans. These latter systems are in many cases much more advanced then the, in comparison, much simpler kiosk-websites.

But to make matters worse these two types of web-based systems are often inseparable since a web-site may in fact consist of two ends – a front-end presented to the visitor in the form of a kiosk-website and the back-end presented to the administrators in form of a web application which for instance manages the information presented in the front-end.

The last class of web-based systems are the extranets – the extranets are all the previously mentioned classes together. An extranet can be viewed as the external part of a company's Intranet in the form of a website.

XHTML validation

The de-facto standard validator for XHTML testing is the W3 validator at http://validator.w3.org.

Appendix D – Glossary

ASP	- Active Server Pages, Microsoft's technology to enables HTML pages to be dynamic and interactive by embedding scripts, i.e. either VBScript or JScript, Microsoft's alternative of JavaScript. Since the scripts in ASP pages (suffix .asp) are processed by the server, any browser can work with ASP pages regardless of its support for the scripting language used therein. *www.softwareag.com/xml/about/glossary.htm*
CASE-tools	- Computer-Aided Software Engineering (CASE) is a collection of tools and techniques that promise revolutionary gains in analyst and programmer productivity. The two prominent delivered technologies are application generators and PC-based workstations that provide graphics-oriented automation of the development process. *www.orafaq.com/glossary/faqglosc.htm*
CMS	- A CMS, or Content Management System, is an application designed to store, format, reproduce and manage Web/intranet data. The CMS usually uses a database to store the content and a server-side scripting language to recall and present the data. *www.sitepoint.com/glossary.php*
CSS	- Short for Cascading Style Sheets, a new feature being added to HTML that gives both Web site developers and users more control over how pages are displayed. With CSS, designers and users can create style sheets that define how different elements, such as headers and links, appear. These style sheets can then be applied to any Web page. *www.rustybrick.com/definitions.php*
ER	- Entity-Relationship, design tool used primarily for relational databases in which entities are modeled as geometric shapes and the relationships between them are shown as labeled arcs *www.harvard.edu/MST/simul/software/docs/pkgs/pgsql/glossary/glossary.html*
FDD	- Feature Driven Development, cf. Appendix A
HDM	- The Hypermedia Design Model, cf. Appendix A
HTML	- In computing, HyperText Markup Language (HTML) is a markup language designed for the creation of web pages and other information viewable in a browser. The focus of HTML is on the *presentation* of information—paragraphs, fonts, italics, tables, and so forth—rather than the *semantics*—what the words mean. http://en.wikipedia.org/wiki/HTML
IDE	- Integrated Development Environment
Kiosk web site	- web sites whose main purpose is to present some kind of information and let the user navigate through the information, cf. Appendix A
OOHDM	- Object-Oriented Hypermedia Design Model (OOHDM)
PERL	- Short for Practical Extraction and Report Language, Perl is a programming language developed by Larry Wall, especially designed for processing text. Because of its strong text processing abilities, Perl has become one of the most popular languages for writing CGI scripts. Perl is an interpretive language, which makes it easy to build and test simple programs. *www.rustybrick.com/definitions.php*
Persona(s)	- A persona is a fictive user pretending to be a future user of the system that is being developed. Cf. Appendix A
PHP	- PHP is a server-side, cross-platform, HTML embedded scripting language that lets you create dynamic web pages. PHP-enabled web pages are treated just like

	regular HTML pages and you can create and edit them the same way you normally create regular HTML pages. *www.terena.nl/library/gnrt/appendix/glossary.html*
RMM	- Relationship Management Model, cf. Appendix A
Spiral model	- (IEEE) A model of the software development process in which the constituent activities, typically requirements analysis, preliminary and detailed design, coding, integration, and testing, are performed iteratively until the software is complete. Syn: evolutionary model. Contrast with incremental development; rapid prototyping; waterfall model. *www.validationstation.com/glossary/glossarys.htm*
UML	- Unified Modeling Language (UML) is an Object Management Group (OMG) standard for modeling software artifacts. Using UML, developers and architects can make a blueprint of a project, much like ERD diagrams are used for relational design. See http://www.rational.com/uml/ for more details. UML diagrams can be constructed with Oracle JDeveloper. *orafaq.cs.rmit.edu.au/glossary/faqglosu.htm*
URL	- Uniform Resource Locator. A way of specifying the location of something on the Internet, e.g., "http://www.photo.net/wtr/thebook/glossary.html" is the URL for this glossary. The part before the colon specifies the protocol (HTTP). Legal alternatives include encrypted protocols such as HTTPS and legacy protocols such as FTP, news, gopher, etc. The part after the "//" is the server hostname ("photo.net"). The part after the next "/" is the name of the file on the remote server. *philip.greenspun.com/panda/glossary*
WCML	- Web Composition Modeling Language, cf. Appendix A
WCPM	- Web Composition Process Model, cf. Appendix A
Waterfall model	- The waterfall model is a software development model first proposed in 1970 by W. W. Royce, in which development is seen as flowing steadily through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. *en.wikipedia.org/wiki/Waterfall_model*
WSDM	- Web Site Design Method, cf. Appendix A
XHTML	- Extensible Hypertext Markup Language - A reformulation of HTML 4.0 in XML 1.0. XHTML is a new language for building web pages that has recently been proposed as a W3C Recommendation. This proposed Recommendation caused lots of debate on account of XHTML's usage of XML namespaces. *orworld.uni-paderborn.de/downloads/glossary/glossary.html*
XML	- The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language for creating special-purpose markup languages. It is a simplified subset of SGML, capable of describing many different kinds of data. Its primary purpose is to facilitate the sharing of structured text and information across the Internet. Languages based on XML (for example, RDF, RSS, MathML, XSIL and SVG) are themselves described in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form. *http://en.wikipedia.org/wiki/XML*
XP	- eXtreme Programming (XP) is a method or approach to software engineering and the most popular of several agile software development methodologies. *http://en.wikipedia.org/wiki/Extreme_Programming*