Project Plan for making the Municipality Website web 2.0

User-Centered Systems Design

Uppsala University 2010-03-10

Christoffer Davidsson, chda1605@student.uu.se
Emilie Ejdemo, emej5588@student.uu.se
Joakim Malmquist, joma5535@student.uu.se
Olov Sandberg, olsa4377@student.uu.se

Table of Contents

1 Introduction

A website that is said to be *Web 2.0* allows its users to interact with each other or to contribute and change the website's content. On traditional websites, Web 1.0, the users were passive and simply viewing the content provided by the website. (O'Reilly, 2005)

The municipality in Uppsala believes that it is time to change their existing website to a Web 2.0-site. This is, to implement the new technology into their homepage to make a more interactive experience for the visitors.

This project will be realized with help of methods found in the book *Software for use: A Practical Guide to the Models and Methods of Usage-Centered Design* written by Constantine & Lockwood (1999). The book has a usage-centered approach, but to make the project more user-centered the different methods and activities from the book will be adapted to the Scrum model.

1.1 Purpose

The purpose of this project is to empower the user and make the municipality's website more usable and interactive. This will make it easier to expose and handle relevant information about happenings and practical issues for the citizens, as well as giving the inhabitants a sense of involvement and power. This document will serve as a project plan for reaching this goal while providing a time and cost budget.

1.2 Assumptions and definitions

The new web 2.0-site will be based on the assumption that a website already exists. After analyzing and examining the old website some decisions will be made about which features should be kept, which should be updated or improved, and which can be deleted.

In general the citizens of Uppsala will use the new website to get information and contribute with their own information. The elder users (55+) will probably, at least in the beginning, use the system mostly to get information since they are not that used to the web 2.0 features. A goal is to make the elder users being able to contribute to the system through the user interface. The younger users will most likely be the ones contributing their own information. The site should be accessible to everyone who can make use of the Internet in an everyday life.

2 Theory

According to Constantine and Lockwood (1999) a highly usable system is easy for people to learn and easy for people to use productively. Usability is never defined explicitly but the authors identify the following five characteristics of a **usable** system:

- Learnability
- Rememberability
- Efficiency in use
- Reliability in use
- User satisfaction

It should be pointed out that Software for Use (Constantine & Lockwood, 1999) is not focused on *user*-centered design; it rather focuses on *usage*-centered design.

2.1 The process according to Constantine and Lockwood

Constantine and Lockwood (1999) believe that the usage-centered design consists of five key elements, which include:

- Pragmatic design guidelines
- Model-driven design process
- Organized development activities
- Iterative improvement
- Measure of quality

Pragmatic design guidelines are some basic guidelines that help the designers to create a usable system; these guidelines contribute to the five characteristics mentioned above. *Model-driven design process* contains of models that help the developers to get a better understanding for usage and a better communication with the users. *The organized development activities* imply that the activities can be rearranged to suit many different software development life cycle models. *The iterative improvement* suggests that, through usability inspections and tests, sequential refinements can be made. The innovative *software metrics* can measure the early indications of the design *quality*. All together these key elements form the coherent approach.

To get a successful usage-centered software design these questions need to be answered:

- Who are the users and how will they relate to the system?
- What tasks are users trying to accomplish through the system we are designing?
- What do they need from the system in order to accomplish their tasks and how should it be organized?
- What are the operating conditions under which the system will be used?
- What should the user interface look, feel and sound like and how should it behave?

Each of these questions refers to a model. The first question refers to *the role model*, which is the relationship between the users and the system. The second question belongs to *the task model*, which defines the tasks that the users need to accomplish. The third question alludes to *the content model*. The content model describes what the interface will consist of in form of tools and materials and how it will be organized into useful collections. The fourth question refers to *the context* where the system is deployed and used, which is described by the operational model. The last question is about the visual design and the operation of the system, which is *the implementation model*. In Figure 1 below the logical relationships between these models are shown.

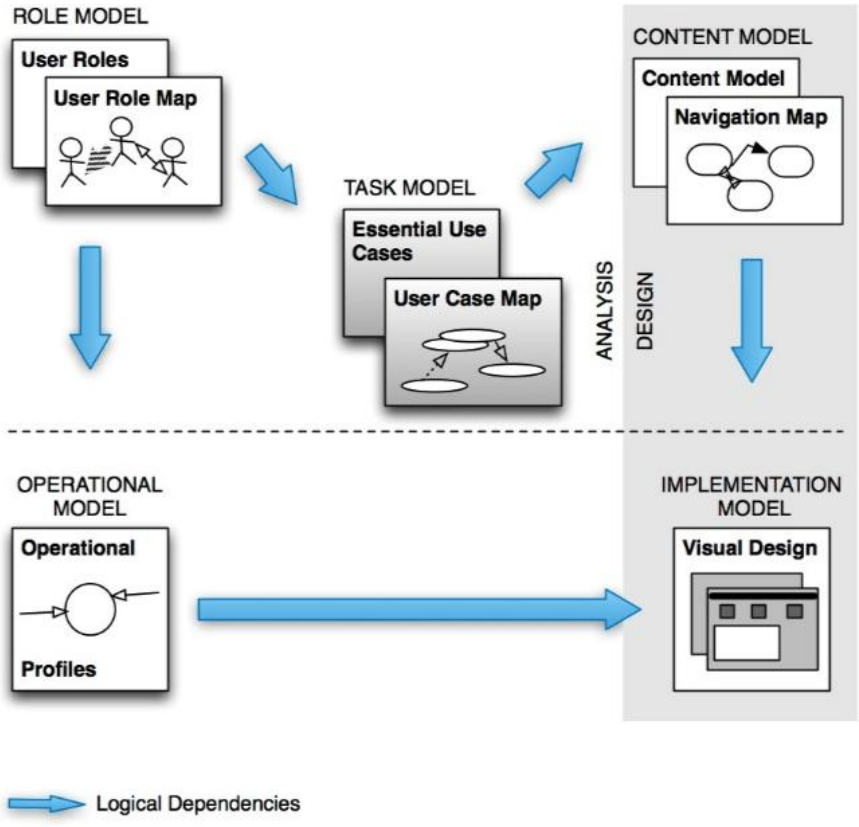

Figure 1. Essential models, logical relationships

Constantine and Lockwood (1999) describe the usage-centered design as a collection of activities that contribute to software usability. The relationships between the activities are logical but the can be arranged in many different ways. Figure 2 shows how these activities can be arranged and how they are connected to each other.

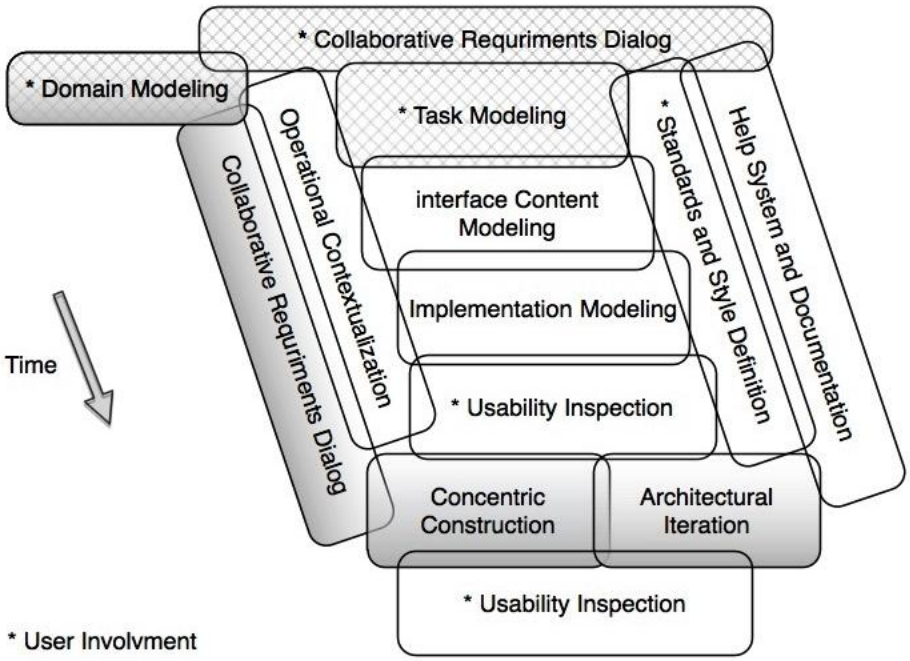

Figure 2. Usage-centered design activity model (Constantine & Lockwood, 1999)

Time flows from top to bottom but this should not be confused with the Waterfall process. The process begins with the *Collaborative Requirements Dialogue, Task Modeling and Domain Modeling* that are crosshatched in the figure. The Collaborative Requirements Dialogue establishes the system requirements through conversation and negotiation between developers and their users and clients. Task Modeling use Role and Task Models to create a final picture of how work can be supported. This is also the core of the usage-centered design process. Domain Modeling develops an entity relationship model or a domain class model through representation of all the interrelated concepts and constructs, which are located in the application model. Task Modeling interacts with Domain Modeling that is outside the immediate scope of usage-centered design. *Interface Content Modeling* comes after and overlaps with task modeling. Task models and content models are often developed through cooperative activities. Interface Content Modeling follows and overlaps with *Implementation Modeling*, which is the process of detailed design and prototyping. Usability Inspection comes after the Implementation modeling and the implementation activities; *Concentric Construction* and *Architectural Iteration*. Some of the modeling and design activities are running in parallel. These are *Operational Contextualization* and *Standards and Style Definition*, which both are

complex and specialized. The former activity accommodates the design to the operational conditions and the context where the system will be deployed. The latter activity is shaping and being shaped through Collaborative Requirements Dialogue to *Usability Inspection*.

This activity model gives an overview of the complete usage-centered design process but projects will not require all of these activities in their total and most specified form. Simply, there is no need in using all activities and in this exact order.

3 Project process

Constantine and Lockwood (1999) write that *"the activities of usage-centered design can be incorporated into almost any software development life cycle, both classical and agile, both rigorous and informal."* (p.24) According to this statement, it should be possible to incorporate the activities presented in the theory section into a Scrum-like development process. The different activities will be used where and when appropriate. In short, Scrum is an agile development process in which a system is developed in increments during iterations called sprints that last between three to eight weeks. Each project team is fairly small, consisting of five to nine people and the main document produced is a product backlog with continuously added requirements. It is important to remember that requirements change.

A project in Scrum begins with an initial meeting where basic requirements are set resulting in a first version of the *product backlog*. The product backlog will change during the course of the project as requirements change. This is followed by a number of sprints which all start with a *sprint planning*, where the most important and needed parts of the product backlog are selected for a *sprint backlog*. The features and requirements in the sprint backlog is what will be designed, developed and analyzed during the current sprint, hence the product is developed in increments. Every day of a sprint starts with a *daily scrum* to keep track of how the project is going and to keep everyone up to date. At the end of a sprint there will be a *sprint review* during which the statuses of the features in the sprint backlog are presented and the finished functionality is presented to the product owner and the stakeholders. This is followed by a *sprint retrospective,* where all team members collaborate to learn from the past sprint and to identify improvements that could be made in the upcoming sprint. Figure 3 below shows the Scrum model that will be used for this project.

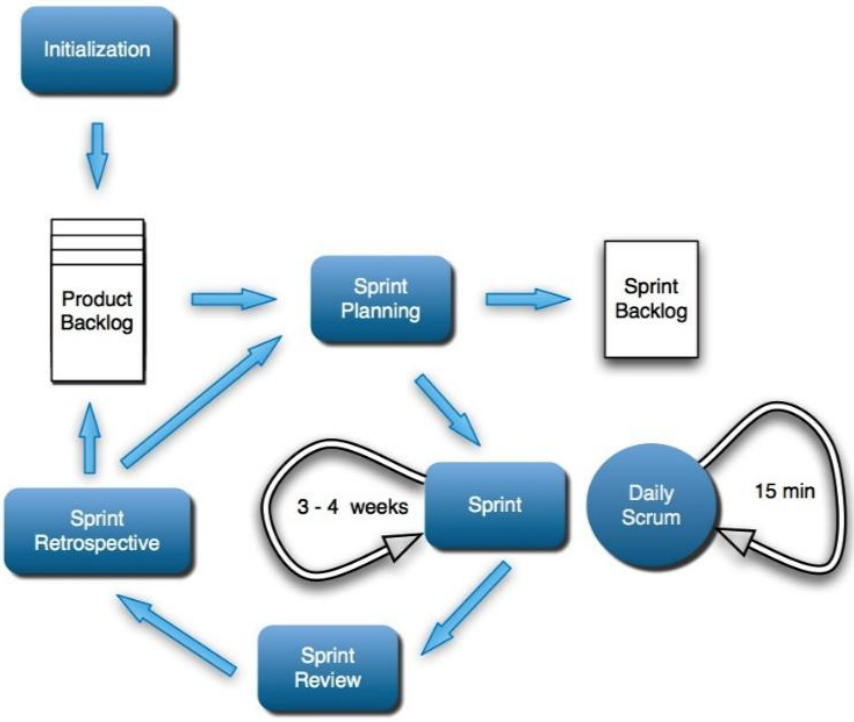

Figure 3. The SCRUM model (Davidsson, 2010)

3.1 Roles

The proposed roles and people in the project should include one *project manager* functioning as a *Scrum master* and also being part of the development team since this person has knowledge about different areas in system development. One *municipality representative* would be the *project owner* and be part of the stakeholders, which also include two or three *representative end users* from the municipality. The project owner is responsible for the backlog; a current to-do-list. In addition to this there will be a need of having two *developers* and one *usability expert,* whose main purpose is to be a link between the users and the developers and to be part of the activities involving the users. The users include two different types; the people working at the municipality and the citizens in the community. Graphical design such as icons and buttons will be outsourced to a local design and graphics company. When and in which activities the different roles will participate are described further on in the activities section.

3.2 Sprints

Since this project will result in a web site, the development will be relatively rapid and the sprints should therefore be kept short. This is due to the fact that a lot of work has been done; there exist a lot of open source modules, API and tools. A proposed length of three to four weeks per sprint seems rational, however this could be changed during the course of the project if there is a need for longer sprints.

4 Activities

Constantine and Lockwood (1999) propose the following activities aiming at usage-centered design and development. They will be described and associated with different activities of the Scrum development process used during the project. Activities that for some reason do not involve the end users enough, in our opinion, will have user involvement added and some activities left out or not used fully according to Software for Use.

4.1 Collaborative Requirements Dialogue

This activity will start with a meeting between the Scrum Master and the Product Owner to discuss the upcoming project. During the initiation, before making the product backlog, the Collaborative Requirements Dialogue will take part. Here the developers and the users will communicate and negotiate to determine the requirements of the system. This is important to do to make the developers aware of what the users need and want. To get to know what the users working at the municipality need and want there will first be a workshop where the project will be introduced and discussed. The result of this first phase is a product backlog consisting of goals, requirements and desired functionality of the project. The Scrum Master will be responsible for this. In addition to this, the different user roles will be defined in the User Model and the usability expert will be in charge of this modeling.

4.2 Task and Domain modeling and Operational context

After the workshop there will be more specified investigations in form of observations of the users working in the real user-environment and interviews with the users that mediate information on the website. During the observations use cases will be made in cooperation with the end users. These are done to understand their work tasks and the operational context. This is important for making the system more efficient and useful for the users, which is partly enabled through the workshop. Before the interview it is important to define the agenda, purpose and relationship. The interview itself should be open with some pre-determined questions. A long list of questions can make it feel more like an inquisition than collaboration. In these activities developers are able to understand ideas and concepts of the application domain, which should yield work task terminology. The other users, the people living in the community, will be examined through user surveys on the website and an investigation of the current interface problems. These problems will be developed through the

employees working in the reception that answer user questions at the municipality office. When this is complete these questions can be answered:

- What will the users of this software be doing?
- What will they be trying to accomplish?
- What do they need from the system to accomplish it?
- How should the system supply what they need?
- What in the system is not working or is ineffective?
- How could the system be made or effective in supporting use?

The answers to these questions and results of these activities will be a Task and Operational Model for which the usability expert is responsible. It is also likely that the product backlog will be updated as a result of these activities.

4.3 Interface Content modeling and Implementation modeling

When the requirements have been specified and analyzed the design of the interface will be initiated. In the interface content modeling low-fidelity prototypes, in form of post-it notes, will be used to get an overview of the working space. With an interface that is ready to use but still incomplete the implementation modeling will take part. The design gradates and prototypes will be made. The prototype will be specified for the layout of the user interface to define the interaction between the users and the system. These will be used later on in the Implementation and the usability expert is responsible for putting together and handling the prototypes.

4.4 Implementation

The sprint backlog will consist of the features to be designed, implemented, analyzed and tested during each sprint. Use cases produced during task and domain modeling will be the basis for the features in the product backlog and therefore the features present in each sprint backlog. The most important features and requirements will be implemented first and less important ones later on in the process, and this is known as *Concentric Construction*. This ensures that the last working version of the product will be as use worthy as possible according to basic core requirements. The implementation will be performed during the sprints, in between analysis, modeling, design and testing. During the first sprint there will be less focus on implementation and more focus on modeling and figuring out what needs to be implemented during the upcoming sprints.

As the product grows and more features are added, some form of chaos and disorder will be introduced in the system architecture. This is due to the fact that we do not design the system as a whole before starting the implementation. A need to restructure code and modules will emerge when there is knowledge of which new features will be added. It is proposed that the basic software architecture is reviewed and revisited after each sprint planning, Constantine and Lockwood (1999) call this *Architectural Iteration.* This could be done as a first step of the implementation and most likely to be needed towards the end of the project.

The main artifacts produced during implementation are the code and the system itself. Features implemented and requirements met as a result of the implementation during each sprint will be marked as finished in the product backlog and therefore implementation also indirectly affects the product backlog. The responsibility for implementation is divided between the developers, including the Scrum Master. The usability expert will take part of the implementation as the programmers feel the need for usability guidance, and as a link between the models and the implementation.

4.5 Collaborative Usability inspections

Usability inspection is a systematic examination of a complete product, design or prototype. It includes features with both heuristic evaluation and pluralistic usability walk-through, and it borrows techniques from expert evaluations. Constantine and Lockwood (1999) describe pluralistic usability walk-through as *"an attempt to help developers put themselves in the shoes of users"* (p.400). These collaborative models of usability have the advantage of bringing together representatives from both the development and the user communities. These are the developers, the end users and the usability expert. Usability inspection will be conducted during each sprint as soon as a new feature is implemented or designed. When an inspection of a design document, a prototype or a working version of the system is made, many usability problems can be avoided already from the start.

The purpose of the inspections is to identify defects. To find the defects, Constantine and Lockwood (1999) suggest that the developers need to practice their thinking to be more like the users, however, in this project real end user representatives will be used. The overall organization or architecture of the product is also being examined and critiqued. It is important to direct critical attention to the product and not to the people who designed or developed it. Developers and the specialists should not argue with the participating users during the inspection. The reason for this is that the developers should not influence the users; the system must be something new for the users.

During the inspection it is important to have a strong leader to keep track of the plan and keep work under control. The usability expert will serve as an inspection recorder maintaining a complete log of identified defects.

4.6 Standards and style definition

Standards and guidelines is a way to aim for usability through consistency and speeding up work by having some pre-made design decisions to fall back upon to. It is tempting to formulate these early and try to reinforce them throughout a project, but Constantine and Lockwood (1999) argue that the definition of standards, style and guidelines should be concurrent during the entire process. *"The standards and guidelines should follow from an understanding of the work and design requirements".* (p.36) Otherwise the developers might run the risk of not understanding the logic behind them.

This project will follow the previous municipality standards and guidelines but question them where they do not make sense. Guidelines and standards should also be developed for this project as the project goes along. They could be proposed during the sprints and then formalized and defined during a sprint retrospective for use and application further on in the project. The project manager, in collaboration with the usability expert, will be in charge of making these standards and style definitions a part of the product backlog, or a separate document serving as an appendix to the product backlog. All developers will be responsible for following the standards and guidelines.

5 Time plan and costs

Because of the task, making a web 2.0-application for employees at the municipality and citizens in the community, the project could last for a long time since new requirements and functionality could be added almost indefinitely. We expect that all of the desired functionality and our goals will be captured and achieved within the deadline of 18 weeks corresponding to six sprints. Hopefully we will also have time to implement some additional and non-vital features in the scope of this time line. The first sprint is represented in Appendix 1, which shows the time of different activities. An overview of all six sprints is presented in Appendix 2 as a Gantt chart.

5.1 Costs and time

The average cost of the Scrum members is set to 900 SEK per hour and person (Lööv, 2010-02-25). This is without the cost for users and their wages or participation costs, for example, the company that will make the final graphical design. The total costs and time for the different roles during the entire project are shown in Table 1. This information is based on the time schedule and the Gantt chart in Appendix 1 and 2.

Table 1. The total costs and time for the roles

ROLES:	**COSTS (SEK):**	**TIME (hours):**	**DAYS (8 hours):**
Scrum Master	288 900	321	40
Project Owner	260 325	289	35
Usability Expert	551 475	613	77
Developer 1	461 160	512	64
Developer 2	85 855	95	12
Developer 3	357 210	397	50
TOTAL:	**~2 005 000**	**2227**	**278**

6 Discussion

Constantine and Lockwood use the term usage-centered design, which can be interpreted in the wrong meaning of how to involve and consider the users' needs in their work situation. In our opinion it is not the usage of the system that is important to handle, but the ones using the system; the users and how they actually use and interact with the system. That is why we have to consider the users and apply a user-centered approach.

Our experience from the book is that users and developers should collaborate and be able to discuss the requirements. We believe that the authors forget one important issue about the goals of the project and the organization and how they relate. It is critical to establish the goals and the requirements with the employees that get affected by the new system and make sure that every department connected to these users are aware of the goals and the demand of the new system. This should also be encouraged through the entire development process. Even the management should participate and understand the goals and the purpose of the development since the new system could affect the entire organization.

The authors never mention that it is important to have a holistic interface since the users' work tasks and the context of use often change during the system development process. Our opinion is that the users should be involved in more activities than Lockwood and Constantine suggest in the book. Several methods are described in detail but not how to use them in the usage-centered work during different activities. One example is prototyping. In the Collaborative Requirements Dialogue and in the end in the testing phase Lockwood and Constantine concentrate on the usage-centered concepts, that emphasis the needs of the users. This aspect is important to apply in every part and continuous feedback operations should be present during the entire process.

As mentioned before it is critical to have a user-centered system process and evaluate the design with users in every iteration and increment. We want to emphasis the relevance of also evaluating the design with the real end users that are not directly involved in the project. These users can give a totally new opinion of the entire system. The extent of user involvement depends on the type of the project, possibilities and the conditions of the project. It also depends on how much the users are affected of the new system. The employees at the municipality must continuously take part of the system development.

In the end we want to say that Software for Use, if modified and used carefully, is a good framework for making a user-centered process. The methods and tools are described in detail but not how to use them in order to understand and realize the users' need in a final usable IT system. Therefore we applied Scrum, which is easy to adapt, flexible and built on iterations, to make the usage-centered model more user-centered. One conditions for this is to use a usability expert, acting as the link between the end users and the system developers.

7 References

Constantine, L. & Lockwood, L., 1999, Software for Use: A Practical Guide to the Models and Methods of Usage-centered Design. New York: ACM Press

Davidsson, C., 2010: The Scrum Figure.

Lööv, J., 2010-02-25: Email.

O'Reilly, What is Web 2.0, 2005: http://oreilly.com/web2/archive/what-is-web-20.html, 2010-03-02.

Appendix 1: Time Schedule

First sprint

	Scrum master	Product owner	Usability expert	System developer 1	System developer 2	System developer 3		Sum:
1. Collaborative Requirements Dialogue								
Meet with product owner	2	2						4
Workshop (Srum m. & usability expert	10	8	10	8	8	8		52
decide milestones, goals and req.)								
2. Task and Domain modeling and								
Operational context								
Investigations (interviews and observations	16		16	16				48
with representive users from each department)								
User surveys and interviews with service	16		16	16				48
department employees								
3. Interface content modeling and								
Implementation modeling								
Analyze requirements (SM and PO analyse	4	8	8	8	8	8		44
before the entire scrum team)								
Prototyping			16		16			32
4. Implementation								
Implementation	20		10	40	40	40		150
Test: Result meet requirements		10	10					20
5. Collaborative Usability Inspections								
Examination of the complete release/design	10	10	10	10				40
/prototype								
Identify defects								
6. Standards and Style defintions								
Scrum master is reponsible of maintaining	1							
the standards and new perspectives								
Total time per role	79	38	96	98	72	56	**Total hours:**	439
					Total cost of first sprint (900 SEK per hour):			395100
					(travel expenses are excluded)			

Appendix 2: Gantt chart