## Projektmanagement

### Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 23.02.2012 | 1.0 | Erste Version des Dokuments | DT |
| 24.02.2012 | 1.1 | Review | CH |
| 13.04.2012 | 1.2 | Änderung Daten Daily Scrum | DT |
| 09.06.2012 | 1.3 | Review und Korrekturen | LE |
| 12.06.2012 | 1.4 | Review | DT |

### Projektplan

Das Management des Projektes Videowall wird im Redmine[[1]](#footnote-1) durchgeführt.

#### Releases

Da das Projekt nach dem Vorgehensmodell Scrum (siehe Unterkapitel I.5 Vorgehensmodell) durchgeführt wird, gibt es nur zu Projektende einen Release.

Während des Projekts wird immer am Ende eines Sprints ein voll lauffähiger Prototyp vorliegen.

Folgender Release ist vorgesehen:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Typ | Beschreibung | Datum |
| 1.0 | Finalversion | Fertige Version, Bugs gefixt | 15.06.2012 |

#### Milestones

Ein Milestone ereignet sich jeweils am Ende eines Sprints. Die Sprints, wie auch die Milestones, sind im Redmine dokumentiert.

Auf der Website https://redmine.elmermx.ch/ steht der detaillierte Projektplan zur Verfügung. Die detaillierte Planung der jeweiligen Sprints erfolgt schrittweise nach dem Vorgehensmodell Scrum (siehe Unterkapitel I.5 Vorgehensmodell).

#### Zeitplan und Zeiterfassung

Die einzelnen Arbeitspakete (Tickets) sind den jeweiligen Sprints zugeordnet. Das Projekt ist in 16 Sprints unterteilt. Das Ende eines Sprints entspricht jeweils einem Milestone.

Die komplette Zeitplanung und die Zeiterfassung werden auf dem Redmine-Server durchgeführt. Für jedes Arbeitspaket wird der Zeitaufwand geschätzt (siehe Kapitel I.1.2.4 Aufwandschätzung) und ein Ticket erstellt. Diese Tickets werden dann den jeweiligen Sprints zugeordnet. Wurde an einem Ticket gearbeitet, wird die dafür aufgewendete Zeit auf das Ticket gebucht. Die Erfassung der Zeit für die jeweiligen bearbeiteten Tickets wird jeweils sofort nach Abschluss der Arbeiten vorgenommen. Somit ist die Zeiterfassung stets aktuell.

Die Reportfunktion bietet einen Überblick über den geplanten und den tatsächlichen Zeitaufwand. Zudem ist es möglich, den Arbeitsaufwand mittels einer cvs-Datei zu exportieren und z.B. in Excel anschaulich darzustellen.

#### Aufwandschätzung

Die Aufwandschätzung ergibt sich durch den geschätzten Aufwand pro Ticket im Redmine.

Der Zeitaufwand, welcher das Abarbeiten eines Tickets benötigt, wird mittels Planning Poker geschätzt. Dazu überlegt sich jedes Teammitglied alleine, wie viel Zeit für die Abarbeitung eines bestimmten Tickets benötigt wird. Laufen die Schätzungen für ein Ticket bei der Besprechung auseinander, kann eine Diskussion versteckte Anforderungen aufdecken. Schliesslich einigt sich das Team auf eine Schätzung.

### Projekt**o**rganisation

#### Team und Verantwortlichkeiten

##### Lukas Elmer (Abk. LE)

Kenntnisse in: Ruby on Rails, PHP, Python / Django, Typo3, Wordpress, Java,   
XHTML, JavaScript, C++, C#, Ubuntu Server

Rolle/Verantwortlichkeiten: Architektur, Serverunterhalt von Redmine, Konfigurations-  
management

Mailadresse : lelmer@hsr.ch

Skype Adresse: lukas.elmer

##### Christina Heidt (Abk. CH)

Kenntnisse in: Java, HTML/CSS, C++, C#, Photoshop

Rolle/Verantwortlichkeiten: Grafisches Design, Risikomanagement, Anforderungen,   
Sitzungsprotokollierung

Mailadresse: cheidt@hsr.ch

Skype Adresse: christina\_heidt

##### Delia Treichler (Abk. DT)

Kenntnisse in: Java, HTML/CSS, C++, C#

Rolle/Verantwortlichkeiten: Überwachung und Erstellung Projektplan, Teamsitzungen,   
Usability-Tests

Mailadresse: dtreichler@hsr.ch

Skype Adresse: de-lia

#### Aufgabenteilung Programmierung

Es ist geplant, jedes wichtige Feature der zu implementierenden Applikation zuerst als kleine Anwendung zu programmieren, welche dann nur eine bestimmte Funktionalität beinhaltet. An der Entwicklung dieser Mini-Applikationen (für weitere Informationen siehe Kapitel TODO: Entwurf, Miniapplikationen), deren Funktionalität nach der Fertigstellung in die Hauptapplikation übernommen wird, sind alle Teammitglieder beteiligt.

Gegen Ende des Projektes soll nochmals Refactoring betrieben werden, damit der Code für die Assistenten, die ihn übernehmen, möglichst sauber strukturiert ist. Dadurch werden die originalen Codeteile der Mini-Applikationen nicht mehr in ihrer alten Form in der Hauptapplikation vorhanden sein. Es wurde entschieden, dass das Refactoring aus einer Hand gemacht werden wird, damit die Applikation in einem Fluss und korrekt strukturiert ist.

Die zuvor erstellten Mini-Applikationen sind genauso bedeutungsvoll wie der Code der Hauptapplikation.

### Risiken

Das Risikomanagement befindet sich im Anhang (siehe TODO).

### Vorgehensmodell

Um alle Kriterien an diese Bachelorarbeit erfüllen zu können, wurden Punkte aus den folgenden drei Vorgehensmodellen angewendet: Scrum, RUP und UCD. Was genau aus diesen Modellen angewendet wurde und wie wird nachfolgend beschrieben.

#### Scrum

Für das Projekt Videowall wird hauptsächlich der Ansatz von Scrum verfolgt, weil dieses Vorgehensmodell auf die Eigenorganisation der einzelnen Teammitglieder ausgerichtet und äusserst produktiv ist, da Overhead so weit wie möglich reduziert wird.

Die nachfolgende Tabelle zeigt auf, welche Elemente (Rollen, Meetings und Artefakte) von Scrum wie gehandhabt werden.

|  |  |
| --- | --- |
| Scrum-Element | Umsetzung |
| Rollen |  |
| Product Owner, Scrum Master | Diese Rollen können personalbedingt nicht besetzt werden. Die Aufgaben des Product Owners und des Scrum Masters werden vom Entwicklungsteam übernommen. |
| Meetings |  |
| Sprint Planung | Die Planung des Sprints wird zu Beginn des jeweiligen Sprints durchgeführt. Termin: Montag, 13.00-13.30: Sprintplanung des nächsten Sprints |
| Daily Scrum | Das Meeting wird zu folgenden Zeiten durchgeführt:  Dienstag, 9.50-10.10: Daily Scrum  Donnerstag, 10.20-10.40: Daily Scrum  Weitere Daily Scrum Meetings nach Bedarf. |
| Sprint Review | Das Review Meeting findet jeweils am vorletzten Tag (Freitag) des aktuellen Sprints statt. (Freitag, 10.10-10.30: Vorbereitung, Vorbesprechung Meeting) |
| Artefakte |  |
| Product Backlog | Die Anforderungen an das Produkt sind als Tickets im Redmine erfasst. Die Schätzung des Aufwands wird für jede Anforderung auf dem entsprechenden Ticket nach dem Modell Planning Poker vorgenommen. |
| Sprint Backlog | Die für einen Sprint geplanten Aufgaben existieren im Redmine als Ticket, welche dem jeweiligen Sprint zugeordnet sind. Der Restaufwand, der für eine einzelne Aufgabe noch benötigt wird, ist über die Differenz der geschätzten und bisher gebuchten Zeit ersichtlich. Gleichzeitig dient das Redmine als Ersatz für das Taskboard. |
| Burndown-Charts | Das Gantt-Diagramm, welches sich im Redmine anzeigen lässt, erübrigt einen Burndown-Chart. |
| Impediment Backlog | Im Redmine Wiki besteht eine Seite zur Eintragung von Hindernissen und Problemen des Projektes. |
| Releaseplan | Die Tickets im Redmine bieten einen Überblick über den Zeitplan und die Termine/Meetings. Die erwartete Anzahl Sprints ist unter Roadmap ersichtlich. Für die Behandlung von Risiken siehe Unterkapitel I.4 Risiken. |

Tabelle 1 - Scrum Elemente

#### RUP

Da für die Bachelorarbeit viele Dokumente erarbeitet werden müssen, ist die Strukturierung der Dokumente an RUP angelehnt (Vorstudie, Anforderungen, Domain Analyse, Entwurf, Realisierung und Test).

Zusätzlich wird zu Beginn der Arbeit ein einfacher Architekturprototyp erstellt, der alle Layer und Tiers abdeckt, um die grössten technischen und architektonischen Risiken abzudecken.

#### UCD

Eine hohe Usability ist für die Vieowall aus verschiedenen Gründen besonders wichtig. Einerseits muss die Videowall ohne die Hilfe eines Benutzerhandbuches, also einfach, zu bedienen sein. Anderseits soll die Videowall die HSR präsentieren und einen möglichst positiven Eindruck bei den Besuchern hinterlassen.

Um eine hohe Usability zu erreichen werden für die Videowall auch Aspekte des User Centered Design (UCD) beachtet. Dazu zählen:

* Analyse des Nutzungskontextes durch Benutzerbeobachtung und Erstellen von Personas und Szenarien
* Benutzerumfragen durchführen und auswerten
* Anforderungen an die Applikation gemäss den Personas und Szenarien definieren
* Aufstellen von Thesen (z.B. „Meine Hand ist die Maus“) und diese Thesen durch Usability Tests validieren
* Implementationen oder Arbeiten mithilfe von Usability Tests validieren

Durch das Vorgehen nach UCD ist es möglich, Benutzer früh miteinzubeziehen und die Anforderungen so zu definieren, dass die Nutzer ihre Ziele auf möglichst einfache Weise erreichen können. Es ist auch möglich, durch Usability Tests die Qualität einer Lösung zu beurteilen und zu verbessern.

1. <https://redmine.elmermx.ch/> [↑](#footnote-ref-1)