|  |
| --- |
| Elmer Lukas, Heidt Christina, Steiner Diego, Treichler Delia, Waltenspül Remo  7. März 2011 |

|  |
| --- |
| SE2 Projekt MRT |
| Projektplan |
|  |

****

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 03.03.2011 | 1.0 | Erste Version des Dokuments | WR |
| 04.03.2011 | 1.1 | Anpassung und Korrektur | TD |
| 06.03.2011 | 1.2 | Anpassung und Korrektur | EL |
| 07.03.2011 | 1.3 | Codeformatierung ergänzt | WR |

## Inhaltsverzeichnis

[1 Dokumentinformationen 1](#_Toc287278452)

[1.1 Änderungsgeschichte 1](#_Toc287278453)

[1.2 Inhaltsverzeichnis 1](#_Toc287278454)

[2 Einführung 3](#_Toc287278455)

[2.1 Zweck 3](#_Toc287278456)

[2.2 Gültigkeitsbereich 3](#_Toc287278457)

[2.3 Definitionen und Abkürzungen 3](#_Toc287278458)

[2.4 Referenzen 3](#_Toc287278459)

[2.5 Übersicht 3](#_Toc287278460)

[2.6 Quellen 3](#_Toc287278461)

[3 Projekt Übersicht 4](#_Toc287278462)

[3.1 MRT (Mobile Reporting Tool) 4](#_Toc287278463)

[3.1.1 Ablauf 4](#_Toc287278464)

[3.1.2 Optionale Features 4](#_Toc287278465)

[3.2 Zweck und Ziel 4](#_Toc287278466)

[3.3 Annahmen und Einschränkungen 5](#_Toc287278467)

[4 Projektorganisation 6](#_Toc287278468)

[4.1 Team 6](#_Toc287278469)

[4.1.1 Lukas Elmer (Abk. EL) 6](#_Toc287278470)

[4.1.2 Christina Heidt (Abk. HC) 6](#_Toc287278471)

[4.1.3 Diego Steiner (Abk. SD) 6](#_Toc287278472)

[4.1.4 Delia Treichler (Abk. TD) 6](#_Toc287278473)

[4.1.5 Remo Waltenspül (Abk. WR) 7](#_Toc287278474)

[4.2 Anmerkung 7](#_Toc287278475)

[4.3 Organisationsstruktur 7](#_Toc287278476)

[4.4 Externe Schnittstellen 7](#_Toc287278477)

[5 Management Abläufe 8](#_Toc287278478)

[5.1 Projekt Kostenvoranschlag 8](#_Toc287278479)

[5.2 Projektplan 9](#_Toc287278480)

[5.2.1 Zeitplan und Zeiterfassung 9](#_Toc287278481)

[5.2.2 Iterationsplanung / Milestones 10](#_Toc287278482)

[5.2.2.1 Inception 10](#_Toc287278483)

[5.2.3 Besprechungen 12](#_Toc287278484)

[5.2.4 Abgabe 12](#_Toc287278485)

[5.2.5 Deployment 12](#_Toc287278486)

[6 Risiko Management 13](#_Toc287278487)

[7 Arbeitspakete 14](#_Toc287278488)

[8 Infrastruktur 16](#_Toc287278489)

[8.1 Software 16](#_Toc287278490)

[8.2 Kommunikation 16](#_Toc287278491)

[9 Qualitätsmassnahmen 17](#_Toc287278492)

[9.1 Allgemein 17](#_Toc287278493)

[9.1.1 Regelmässige Teamsitzungen und teamfördernde Massnahmen 17](#_Toc287278494)

[9.1.2 Sourcecode Management 17](#_Toc287278495)

[9.1.3 Issuetracking 17](#_Toc287278496)

[9.1.4 Austausch 17](#_Toc287278497)

[9.1.5 Documetation Guidelines & Review 17](#_Toc287278498)

[9.2 Codequalität 18](#_Toc287278499)

[9.2.1 Codereview 18](#_Toc287278500)

[9.2.2 Styleguide für Code 18](#_Toc287278501)

[9.2.3 Projektautomation 20](#_Toc287278502)

[9.3 Tests 20](#_Toc287278503)

[9.3.1 Unit Tests 20](#_Toc287278504)

[9.3.2 Systemtests 20](#_Toc287278505)

[9.3.3 Usability Tests 20](#_Toc287278506)

# Einführung

## Zweck

Dieses Dokument beschreibt den Projektplan für das Projekt MRT (Mobile Reporting Tool).

## Gültigkeitsbereich

Dieses Dokument gilt als Grundlage des Projektes und ist daher über die gesamte Projektdauer gültig.

## Definitionen und Abkürzungen

Die Definitionen und Abkürzungen befinden sich in der ausgelagerten Datei doc/01\_Projektplan/glossar.docx.

## Referenzen

* doc/00\_Projektantrag/projektantrag\_mrt.docx
* doc/01\_Projektplan/risikomanagement.xlsx
* doc/01\_Projektplan/glossar.docx
* doc/templates/template.dotx
* doc/media/logo.png

## Übersicht

Im Abschnitt „3. Projekt Übersicht“ wird das Projekt beschrieben und dessen Ziel und Zweck erläutert.

Darauf folgt der Abschnitt “4. Projektorganisation”, in dem näher auf die Organisationsstruktur und externe Schnittstellen eingegangen wird.

Die Planungsübersicht des Projektes wird im Abschnitt “5. Management Abläufe” detailliert dargestellt.

Der Punkt 6. Risiko Management wird in einem separaten Dokument aufgezeigt.

Sämtliche Arbeitspakete des Projektes werden im Abschnitt “7. Arbeitspakete” aufgelistet und beschrieben.

Im Abschnitt “8. Infrastruktur” werden die genutzten Räume, Geräte und Softwareprodukte aufgezeigt.

Abschliessend wird die Qualitätssicherung im Abschnitt “9. Qualitätsmassnahmen” genauer erläutert.

## Quellen

Als Quellen wurden alle Dateien auf dem HSR Skripte Server im Unterordner S:\Informatik\Fachbereich\Software-Engineering\_2\_-\_Projekt\SE2P verwendet. Insbesondere wurden die Beispielprojekte der letzten Jahre als Vorlage und/oder als Inspiration verwendet.

# Projekt Übersicht

## MRT (Mobile Reporting Tool)

Wie der Name „Mobile Reporting Tool“ bereits erahnen lässt, handelt es sich bei dem Produkt um ein Werkzeug, womit Aussendienstmitarbeiter auf unkomplizierte Art und Weiseihre Stundenrapportieren können und zwar unabhängig von ihrem Standort.

### Ablauf

#### Arbeitseinsatz

Der Arbeiter erhält von der Zentrale oder direkt von einem Kunden einen Auftrag. Sobald er bei dieser Adresse angelangt ist, drückt er die „Start“-Schaltfläche. Durch das Drücken der „Start“-Schaltfläche beginnt die Zeitmessung.  
Sobald die Arbeit vollbracht ist, teilt der Arbeiter dies dem System durch das Drücken der „Stopp“-Schaltfläche mit. Nun wird ihm eine Reihe von potentiellen, bereits registrierten Kunden vorgeschlagen. Der Arbeiter kann den passenden Kunden auswählen oder diesen Schritt überspringen. Um die spätere Verwaltung zu erleichtern, kann der Mitarbeiter zudem eine Beschreibung erfassen, dies ist optional.  
Zudem werden bei jedem Drücken der Schaltfläche die GPS-Daten an den Server übermittelt. Dadurch kann auch im Nachhinein der Kunde eingegrenzt und ausgewählt werden, falls es dem Arbeiter beim Drücken der Stopp-Taste nicht möglich sein sollte (z.B. aus Zeitgründen).

#### Verwaltung & Auswertungen

Die Verwaltung der Kundendaten bzw. Mitarbeiter wird über einen dedizierten Server durchgeführt, auf den alle Aussendienstmitarbeiter sowie Büroangestellten über ein Web-Login Zugriff haben. Die für den Zugriff autorisierten Personen haben unterschiedliche Rechte, abhängig von deren Tätigkeiten.  
Es können neue Kunden bzw. Mitarbeiter erfasst, sowie diverse Auswertungen wie z.B. Rapporte, Mitarbeiterlisten etc. generiert werden.

### Optionale Features

* Kunden direkt auf dem Android erfassen
* Kundenvorschläge anhand von GPS-Koordinaten
* Zusätzliche Auswertungen (z.B. um (halb-)automatisch Rechnungen zu erzeugen)
* Assoziieren von Kundendaten mit GPS-Koordinaten (mithilfe der Google Maps / Google Places API)

## Zweck und Ziel

Im Modul Software Engineering 1 in der Durchführung HS 2010/2011haben sich alle Teammitglieder fundamentales theoretisches Wissen im Bereich Software Engineering angeeignet. Nun soll dieses Wissen durch das Projekt MRT vertieft werden.

Ein weiteres Ziel ist, mittels geeigneten Werkzeugen wie Redmine, Subversion und Skype gut in der Gruppe zu kooperieren. Denn eine klare und effiziente Kommunikation ist die Grundlage eines jeden erfolgreichen Projektes.

Für das Projekt steht vergleichsweise wenig Zeit zur Verfügung. Aus diesem Grund wird das Produkt nach Abschluss des Projektes noch zu wenig ausgereift sein, um im produktiven Arbeitsumfeld ein­gesetzt werden zu können. Das Augenmerk wird darauf gerichtet, eine saubere und erweiterbare Basis zu entwickeln, welche zu einem späteren Zeitpunkt erweitert werden könnte. Es soll die Möglichkeit bestehen, das Projekt flexibel auszubauen, ohne auf entwicklungsbedingte Grenzen zu stossen.

## Annahmen und Einschränkungen

Für dieses Projekt werden keine Annahmen oder Einschränkungen getroffen.

# Projektorganisation

Das Projektteam besteht aus fünf sich gleichgestellten Mitgliedern. In der „Construction“-Phase werden zwei Unterteams gebildet, welche sich jeweils auf einen Tier konzentrieren und das Code-Review für das jeweils andere Team übernehmen.

## Team

### Lukas Elmer (Abk. EL)

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

Motivation: Weitere Erfahrungen im Software-Engineering sammeln.  
SE1 Kenntnisse auf reales Projekt anwenden.

Spezialisierung: Android-Client

Verantwortlichkeiten: Coaching bei komplexen Problemen bei Ruby on Rails, Serverunterhalt  
von Redmine, automatisches Deployment

Mailadresse : lelmer@hsr.ch

Skype Adresse: lukas.elmer

### Christina Heidt (Abk. HC)

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

Motivation: Grundlagen von SE1 auf Projekt anwenden und Wissen vertiefen

Spezialisierung: Webplattform

Verantwortlichkeiten: Grafisches Design, Sitzungsprotokollierung, Teamleitung

Mailadresse: cheidt@hsr.ch

Skype Adresse: christina\_heidt

### Diego Steiner (Abk. SD)

Kenntnisse in: PHP, XHTML, ASP.net, Linux

Motivation: vorhanden

Spezialisierung: Webplattform

Verantwortlichkeit: Überwachung Projektplan, Anforderungs- und Featurespezifikation

Mailadresse: dsteiner@hsr.ch

Skype Adresse: diego2alps

### Delia Treichler (Abk. TD)

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

Motivation: Anwendung des theoretischen Wissens aus dem Modul SE1

Spezialisierung: Android Client

Verantwortlichkeit: Teamsitzungen, Teamfördernde Massnahmen

Mailadresse: dtreichler@hsr.ch

Skype Adresse: de-lia

### Remo Waltenspül (Abk. WR)

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

Motivation Erfahrungen in einem Software-Engineering Projekt sammeln

Spezialisierung: Webplattform

Verantwortlichkeit: Technisches Design, Domainmodellierung

Mailadresse: rwaltens@hsr.ch

Skype Adresse: kobold246

## Anmerkung

Bei der Spezialisierung handelt es sich um die Unterteilung des Teams während der „Construction“-Phase.

## Organisationsstruktur



## Externe Schnittstellen

Für die Beratung und Benotung ist Prof. Hans Rudin zuständig. Als zusätzlicher Berater steht Daniel Keller zur Verfügung.

# Management Abläufe

Das folgende Kapitel erläutert kurz die essentiellen Management Abläufe. Dies beinhaltet den Projekt Kostenvoranschlag mit den Rahmenbedingungen sowie den Projektplan, welcher die zeitliche Aufteilung des Projekts in diverse Arbeitsschritte enthält.

## Projekt Kostenvoranschlag

Der Projektstart ist am 24. Februar 2011. Während 14 Wochen (exklusive einer Woche Ferien) wird am Projekt gearbeitet. Die Abgabe findet in der 22. Kalenderwoche statt, spätestens am 3. Juni 2011.

Von jedem Mitarbeiter wird eine Mindestarbeitszeit von acht Stunden erwartet, wobei diese acht Stunden bereits die vier offiziellen Lektionen enthalten. Falls zusätzliche nicht geplante Schwierigkeiten auftauchen, kann das Arbeitspensum pro Mitarbeiter auf höchstens 10-12 Stunden pro Woche erhöht werden.

Der Projektumfang erlaubt es uns, die vorgesehene Zeit vollumfänglich auszunutzen. Sofern die Kernfunktionalitäten schneller als geplant fertig gestellt werden können, wird das Projekt um zusätzlich vorerst als optional definierte Module ergänzt.

Es sind gegenwärtig keine geplanten Absenzen von Projektmitarbeitern bekannt, gegebenenfalls werden Abwesenheiten im Projektplan einkalkuliert.

## Projektplan

### Zeitplan und Zeiterfassung

Die Zeiterfassung sowie der komplette Zeitplan werden vollumfänglich über einen dedizierten Redmine-Server durchgeführt. Dazu werden in einem ersten Schritt so genannte Tickets erfasst, welche vordefinierten Milestones zugewiesen werden. Diese Tickets entsprechen den Arbeitspaketen, die in einem nachfolgenden Abschnitt detaillierter spezifiziert werden.

Anhand der erstellten Tickets wird automatisch ein Zeitplan (Gantt-Diagramm) generiert, in dem alle Arbeitspakete übersichtlich dargestellt werden.

Für die einzelnen Reviews an den festgelegten Milestones werden Auszüge (CSV) aus dem Redmine generiert und anschliessend direkt in Excel formatiert. Mithilfe einer Excel-Vorlage können aus diesen generierten Daten automatisch Diagramme erzeugt werden, welche den Arbeitsaufwand der einzelnen Mitarbeiter optisch ansprechend anzeigen.

Zudem kann man zu jeder Zeit direkt im Redmine den Verlauf der verschiedenen Tickets bzw. Arbeitspakete verfolgen. Um einen Soll-Ist-Vergleich zu ermöglichen, wird die geleistete Arbeitszeit von den Projektmitarbeitern direkt auf ein bestimmtes Ticket gebucht.

Ferner ist das Versionsverwaltungssystem SVN mit dem Redmine verbunden, wodurch SVN bedingte Änderungen direkt bei einem Ticket in einer Historie angezeigt werden.

Der gesamte Zeitplan mit allen Arbeitspaketen wird über die Projektmanagementsoftware Redmine verwaltet. Sie ist für alle Projektmitglieder unter <https://redmine.elmermx.ch> erreichbar.

Jeden Abend werden die am Tag gearbeiteten Stunden am Projekt auf das zugehörige Ticket gebucht.

### Iterationsplanung / Milestones

Siehe zeitplan\_gantt.pdf

#### Inception 21.02.-10.03.11 / SW01-SW03

|  |  |
| --- | --- |
| Inception I1 | Termine |
| * Projektantrag * Projektplan * Codestyleguide und Glossar * Iterationsplan und Arbeitspakete | 25.02.11: Abgabe Projektantrag  08.03.11: MS1 - Abgabe Projektplan |

#### Elaboration 11.03.-14.04.11 / SW04-SW08

Da die Elaboration Phase 5 Wochen lang dauert, wurden 2 Iterationen für diese Phase geplant (Elaboration 1 und Elaboration 2). Somit stellt ein abgeschlossener Milestone eine Iteration dar.

|  |  |
| --- | --- |
| Elaboration E1 / 11.03.-31.03.11 / SW04-SW06 | Termine |
| * Vision * Anforderungsspezifikation   + Use Cases   + Nichtfunktionale Anforderungen * Domainanalyse   + Design Klassendiagramme   + Klassenspezifikation   + SSDs   + Contracts | 29.03.11: MS2 - Abgabe Analyse |

|  |  |
| --- | --- |
| Elaboration E2 / 01.04.-14.04.11 / SW07-SW08 | Termine |
| * Architekturprototyp (V0.1, siehe 5.2.4 Abgabe)   + Lauffähig   + Über alle Layers und Tiers   + Technische Risiken abgedeckt   + Architektur grob dokumentiert   + Entwurf Klassendiagramme   + Unit- und Systemtests   + Demo * Anforderungen vollständig und stabil * Test- und Reviewprozeduren definiert * Risikoanalyse nachgeführt, Hauptrisiken beseitigt * Feinplanung Constructionphase * Entwicklunsumgebung komplett eingerichtet * Wireframes (optional)   + Usability Konzept getestet (optional) | 12.04.11: MS3 - Abgabe Elaboration |

#### Construction 15.04.-12.05.11 / SW09-SW11

In der Construction Phase stehen tatsächlich nur 3 Wochen und nicht 4 Wochen zur Verfügung, da sich an diesem Datum noch 1 Woche Ferien befinden. Deshalb wird für die Constructunphase auch nur 1 Iteration geplant. Für ein grössers Projekt wäre das natürlich viel zu wenig, aber im SE2 Projekt geht es ja nicht um das Programmieren sondern viel mehr ums Software Engeneering.

|  |  |
| --- | --- |
| Construction C1 | Termine |
| * Architektur / Design inhaltlich   + Physische Architektur   + Logische Architektur   + Schnittstellen zw. Packages   + Persistenz   + Design Packages   + Externes Design   + Besonderheiten der Designs   + Eingesetzte Technologien   + Dokumentation der Architektur * Interner Release der V0.2 (siehe 5.2.4 Abgabe) | 10.05.11: MS4 - Abgabe Construction |

#### Transition 13.05.-03.06.11 / SW12-SW14

|  |  |
| --- | --- |
| Transition T1 | Termine |
| * Release der V1.0 (siehe 5.2.4 Abgabe) * Deployment V1.0 ausführen * Präsentation   + Fertigstellen   + Präsentieren   + Demo der V1.0 * Testspezifikationen und Protokolle * Installationsanleitung * Screenshots von Teilen der SW * Benutzerhandbuch * Projektmonitoring Auswertung * Erklärung eigenständige Arbeit | 31.05.11 MS5 - Abgabe Transition  02.06.11 MS5 - Schlusspräsentation |

### Besprechungen

Teambesprechungen werden zweimal wöchentlich durchgeführt:

* Montags, 12:15 bis 13:00, kurze Besprechung
* Donnerstags, umfassende Besprechung am Nachmittag (2 Stunden)

Ort: in einem zur Verfügung stehenden Zimmer an der HSR  
Teilnehmer: EL, HC, SD, TD, WR  
Die Meetings dienen zum Austausch in der Gruppe, zur Besprechung anstehender Arbeiten, deren Koordinierung im Team und zum Erarbeiten von Lösungsansätzen für angefallene Probleme.  
Die Besprechungen am Donnerstag beinhalten auch die Meilenstein Reviews und weitere Besprechungen mit dem Betreuer. Zu jeder Besprechung wird ein Sitzungsprotokoll geführt (siehe Sitzungsprotokolle).

### Abgabe

Folgende Versionen sind vorgesehen:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Typ | Beschreibung | Datum |
| 0.1 | Architekturprototyp | Basissystem, Datenstrukturen | 12.04.11 |
| 0.2 | Testversion | Businesslogic, UI, Schnittstellen | 10.05.11 |
| 1.0 | Finalversion | Fertige Version, Bugs gefixt | 31.05.11 |

### Deployment

Da die Webapplikation schlussendlich auf einem Server läuft, muss diese, nach der Entwicklung und dem Testen des Quellcodes, deployed (auf den Server hochgeladen) werden. Um das Deployment zu vereinfachen und zu automatisieren wird Capistrano eingesetzt. Somit kann ein Deployment ganz bequem mit dem Befehl „cap deploy“ hochgeladen werden.

# Risiko Management

Da das Risiko Management komplex ist, wurde es in ein separates Dokument ausgelagert. Sämtliche Angaben zu möglichen Risiken, die während dem Projekt auftauchen könnten, werden in diesem Dokument detailliert analysiert und erläutert.

Um dynamisch neue Risiken zu erfassen bzw. als beseitigt zu definieren, existiert ein spezielles Arbeitsblatt. Allfällige technische implementationsabhängige Risiken werden als Ticket im Redmine erfasst und sobald wie möglich geklärt bzw. beseitigt.

# Arbeitspakete

Die Beziehung zwischen den einzelnen Arbeitspaketen und den zugehörigen Iterationen ist im Redmine ersichtlich. Die unten dargestellte Tabelle dient nur zum Überblick, folge dessen werden die Pakete komplett über den Redmine-Server verwaltet.

|  |  |  |  |
| --- | --- | --- | --- |
| Arbeitspaket | Inhalt | Abhängigkeiten | Risiken/Probleme |
| Projekt Management |  |  |  |
| Redmine Projekt | Projekt im Redmine aufsetzen |  |  |
| Projektantrag | Konzept in Antrag formulieren |  |  |
| Projektplan | Detaillierter Projektplan erstellen | Iterationsplanung |  |
| Logo | Logo Erstellung für Dokumentvorlagen |  |  |
| Dokumentvorlage | Einheitliche Dokumentvorlagen | Logo |  |
| Codestyleguide | Styleguide für Code definieren |  |  |
| Iterationsplanung | Planung der einzelnen Phasen |  |  |
| Administratives | Zeit rapportieren, Arbeitsumgebung einrichten |  |  |
| Business Modeling |  |  |  |
| DM | Domain Model | Use Cases |  |
| SSD | Systemsequenzdiagramm | Use Cases |  |
| Operation Contracts | Operation Contracts | Domain Model, SSD |  |
| Requirements |  |  |  |
| UC Mitarbeiter Rapport | Usecase des Aussendienstmitarbeiters |  |  |
| UC Rapport | Usecase des Sekretärs |  |  |
| UC Verwaltung | Usercase des Managers |  |  |
| Vision | Vision dokumentieren |  |  |
| Glossar | Glossar |  |  |
| Analyse und Design |  |  |  |
| Paperprototyping | GUI Papierprototyp erarbeiten |  |  |
| Interaktionsdiagramme | Interaktionsdiagramme |  |  |
| Klassendiagramme | Klassendiagramme | Domainmodel |  |
| Implementation |  |  |  |
| Android Client | Implementation des Android Clients | KDs, IDs |  |
| Reporting | Implementation des Reportings aus den gesammelten Daten | KDs, IDs, |  |
| Backoffice Server | Implementation des Serverteils der die Schnittstelle für Android bietet | KDs, IDs |  |
| Tests |  |  |  |
| Systemtests | Testcases für das gesamte System | Use Cases |  |
| Unit Tests | Testcases für einzelne Klassen | Use Cases |  |
| Bugfixing | Fehler die aus dem Test hervorgingen beheben | Systemtests, Unit Tests |  |
| Usability Tests | Testcases mit Benutzern | Paperprototyping |  |
| Deployment |  |  |  |
| Deployment auf Testumgebung | Release auf Testumgebung deployen |  |  |
| Deployment auf Liveumgebung | Release auf Liveumgebung deployen |  |  |
| Dokumentation |  |  |  |
| Sitzungsprotokolle | Protokolle zu den Sitzungen | Sitzungen |  |
| Bedienungsanleitung | Bedienungsanleitung |  |  |
| Schlusspräsentation | Schlusspräsentation |  |  |
| Sitzungen |  |  |  |
| Sitzungen | Sitzungen |  |  |
| Qualitätssicherung |  |  |  |
| Qualitätsmassnahmen | Qualitätsgrundlagen definieren |  |  |
| Code Review | Code reviewen | Implementation |  |
| Review Projektplan | Projektplan reviewen | Projektplan |  |
| Review Dokumente | Dokumente reviewen | Dokumente |  |
| Studium Technologien |  |  |  |
| Fachliteratur | Aneignung neuer Programmiersprachen |  |  |

# Infrastruktur

Für Besprechungen und gemeinsame Arbeiten werden die verfügbaren Räume der HSR genutzt. Die Projektmitglieder arbeiten bevorzugt mit den persönlichen Notebooks (ausgestattet mit Windows 7 und Ubuntu Linux). Bei Ausfall eines dieser Geräte kann gegebenenfalls einer der HSR-Arbeitsrechner verwendet werden. Zur Projektverwaltung wird der persönliche SVN-Server von Lukas Elmer genutzt. Zudem stehen für das Testen der mobilen Applikation drei Android Mobiltelefone zur Verfügung.

## Software

Folgende Software kommt zum Einsatz:

|  |  |  |
| --- | --- | --- |
| Entwicklung | Projektmanagement | Dokumentation |
| * Eclipse IDE, Version 3.6.2 * NetBeans IDE, Version 6.9.1 * Subversion, Version 1.6.12 * Android SDK, Version 3.0 * Ruby on Rails, Version 3.0.5 | * Redmine, Version 1.1.1 | * MS Office (2007, 2010) * Enterprise Architect, Version 7.5 * Google Documents |

## Kommunikation

Für die Kommunikation zwischen den Projektmitarbeiter werden folgende Hilfsmittel verwendet:

* Wiki Redmine (<https://redmine.elmermx.ch/projects/mrt/wiki>)
* E-Mail (Mailing Liste an alle Teammitglieder: mrt@elmermx.ch)
* Meetings (siehe 5.2.3 Besprechungen)
* Skype

# Qualitätsmassnahmen

## Allgemein

### Regelmässige Teamsitzungen und teamfördernde Massnahmen

Neben den regelmässigen Teamsitzungen (siehe 5.2.3 Besprechungen) nimmt das gesamte Team einmal pro Monat an einem gemeinsamen Nachtessen Teil. Dabei wird das Projekt auf einer lockeren Basis diskutiert und der Teamgeist gefördert.

### Sourcecode Management

Der gesamte Sourcecode wird von Subversion verwaltet. Diese Software hat sich schon in vielen produktiven Projekten und in einem Projekt während des letztens Semesters bewährt, somit wird auf eine solide Basis gebaut. Das Repository ist mit dem Redmine verlinkt, sodass direkt auf Issues committed werden kann. Die URL des Subversion Repositories lautet: <https://se2p_svn.elmermx.ch/>. Jedes Projektmitglied hat seine persönlichen Zugangsdaten bereits erhalten.

### Issuetracking

Als Issuetracker wird Redmine auf <https://redmine.elmermx.ch> verwendet. Redmine bietet ein ausgereiftes Bug- und Issuemanagement sowie eine SVN Integration und ausgereiftes Reporting. Jedes Projektmitglied hat seine persönlichen Zugangsdaten bereits erhalten.

### Austausch

Als Plattform für Fragen und zum Austausch nutzen wir das bereits vorhandene gegen aussen geschlossene Wiki von Redmine (<https://redmine.elmermx.ch/projects/mrt/wiki>). Die Ergebnisse der Sitzungen werden jeweils in den Sitzungsprotokollen festgehalten.

### Documetation Guidelines & Review

Alle im Rahmen des Projektes erzeugten Word-Dokumente werden auf der Basis eines Word-Templates erstellt. Dadurch werden den über alle dem Projekt zugehörigen Dokumenten eine gewisse Struktur und damit eine Konsistenz gegeben.

Da die Änderungsgeschichte eines Dokumentes wertvoll ist, wird diese in jedem Dokument separat geführt. Die entsprechende Tabelle befindet sich im ersten Abschnitt jedes Dokumentes.

Bevor ein Dokument als Final deklariert werden kann, muss es von mindestens einer weiteren Instanz gegengelesen und freigegeben werden. Dies geschieht, indem im Dokument bei den Dokumentinformationen ein entsprechender Eintrag gemacht wird.

## Codequalität

### Codereview

Mindestens einmal pro Iteration findet ein Codereview durch das jeweils andere Team statt. Damit behalten alle den Überblick über das gesamte Projekt und der Code wurde von mindestens einer anderen Instanz überprüft.

Der Ablauf eines Tickets könnte etwa folgendermassen aussehen:

1. Beginn Implementation Feature XXX (EL) 🡪 Status ändert von „New“ nach „In Progress“
2. Arbeiten fertig für Feature XXX (EL) 🡪 Status ändert zu „Resolved“
3. Review für Feature XXX wird gemacht (TD) 🡪 Status ändert zu „Closed“

Durch diesen Ablauf kann garantiert werden, dass bei jedem Ticket nach den Arbeiten ein Review gemacht wird.

### Styleguide für Code

Alle Entwickler haben sich beim Schreiben von Code an die in dem Styleguide für Sourcecode definierten Richtlinien zu halten.

Für beide Technologien (Ruby on Rails & Android) werden die Standardvorlagen benutzt, abgesehen von der Anpassung der Breite (200 statt 80 Zeichen).

#### Grundsätzliches

* Einfache Klassen oder Methoden dürfen, müssen aber nicht kommentiert werden
* Komplexe und schwierig zu verstehende Klassen und Methoden sollen kurz kommentiert werden
* Jegliche Art von Code oder Kommentar wird in englischer Sprache geschrieben

#### Android

* Paketnamen werden klein geschrieben
* Methoden werden klein und camelCase geschrieben
* Klassen werden gross und CamelCase geschrieben
* Konstanten / Static Variabeln werden GROSS\_UND\_MIT\_UNTERSTRICH geschrieben
* Für die Formatierung wird bei dem Eclipse-Projekt in den Projekteinstellungen ein Code Style Formatting File ins SVN eingecheckt, damit die automatische Formatierung einheitlich bleibt.

##### Codeformatierung

Auf der nachfolgenden Abbildung (Abb. 1 Codeformatierung Android) sind sämtliche wichtigen Codeformatierungen ersichtlich, die für unser Projekt verwendet werden.



*Abb. 1 Codeformatierung Android*

#### Webplattform (RoR)

* Methoden werden klein\_und\_mit\_unterstrich geschrieben
* Klassen werden gross und CamelCase geschrieben
* Konstanten / Static Variabeln werden GROSS\_UND\_MIT\_UNTERSTRICH geschrieben
* Für die Formatierung wird bei dem Netbeans-Projekt in den Projekteinstellungen ein Code Style Formatting File ins SVN eingecheckt, damit die automatische Formatierung einheitlich bleibt.

##### Codeformatierung

Auf der nachfolgenden Abbildung (Abb. 2 Codeformatierung Ruby on Rails) sind sämtliche wichtigen Codeformatierungen ersichtlich, die für unser Projekt verwendet werden.



*Abb. 2 Codeformatierung Ruby on Rails*

### Projektautomation

* Im Android-Teil sorgt ein Ant-Skript für regelmässiges kompilieren und automatisiertes Testen des Projektes.
* Für den Rails-Teil sind keine automatischen Builds vorgesehen, da Ruby eine Skriptsprache ist und dementsprechend nicht kompiliert werden muss. Das Framework sieht automatische Tests bereits vor, die vor jedem Commit vom Entwickler manuell ausgelöst werden müssen (Befehl: rake test). Falls sich wiederholende Tasks anfallen, wie z.B. das Simulieren eines neuen Stundeneintrags mit GPS Koordinaten, so wird dafür ein eigener Rake-Task geschrieben, der dann über die Konsole (rake TASK\_NAME) ausgelöst werden kann.

Der Einsatz eines Jenkins (Hudson)-Servers hat sich nach einer Teambesprechung als nicht sinnvoll erwiesen.

## Tests

### Unit Tests

Für alle wichtigen Klassen werden Unit Tests geschrieben. Dadurch kann eine hohe Qualität der einzelnen Komponenten in sich selbst gewährleistet werden. Um die Tests durchzuführen, werden Skripts geschrieben, damit die Tests automatisiert ausgeführt werden können (z.B. mit dem Befehl „rake test“).

### Systemtests

Um die Qualität der Gesamtsoftware sicherzustellen, wird am Ende der Milestones 3, 4 und 5 durch Systemtests die Software vollständig getestet. Dafür wird ein Systemtestprotokoll erarbeitet und den Test entsprechend ausgefüllt.

### Usability Tests

In der Elaboration Phase werden Paper-Prototyping Methoden eingesetzt.

Später, am Ende der Construction Phase, werden Usability Tests mit echten Benutzern durchgeführt, um Schwächen im Bereich Usability zu erkennen und zu beseitigen.