Mobile Reporting Tool

SE2 Projekt MRT

Projektplan

Elmer Lukas, Heidt Christina, Steiner Diego, Treichler Delia, Waltenspül Remo

8. März 2011

1 Dokumentinformationen

1.1 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
07.03.2011	1.4	Textanpassungen und Arbeitspakete	EL
07.03.2011	1.5	Review	TD
08.03.2011	1.6	Formatierung & Review	SD
08.03.2011	1.7	Änderungsverfahren erstellt	WR
08.03.2011	2.0	Anpassung und Korrektur gemäss Besprechung vom 08.03.2011 mit Prof. Hans Rudin	EL
08.03.2011	2.1	Review	SD

1.2 Inhaltsverzeichnis

2 Einführung

2.1 Zweck

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

2.2 Gültigkeitsbereich

Dieses Dokument gilt als Grundlage des Projektes und ist daher über die gesamte Projektdauer gültig (21.02 bis 03.06.2011).

Damit der Projektplan immer auf dem neusten Stand ist, wird laufend angepasst.

2.3 Definitionen und Abkürzungen

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

2.4 Referenzen

- doc/00_Projektantrag/projektantrag_mrt.docx
- doc/01_Projektplan/risikomanagement.xlsx
- doc/01_Projektplan/glossar.docx
- doc/01_Projektplan/zeiterfassung.xlsx
- doc/02_Protokolle/*
- doc/templates/template.dotx
- doc/templates/java_formatting.xml
- doc/templates/ruby_formatting_settings.zip
- doc/templates/template_protokoll.dotx
- doc/media/logo.png

2.5 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.
Im Abschnitt “8. Infrastruktur” werden die genutzten Räume, Geräte und Softwareprodukte aufgezeigt.
Abschliessend wird die Qualitätssicherung im Abschnitt “9. Qualitätsmassnahmen” genauer erläutert.

2.6 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.

3 Projekt Übersicht

3.1 MRT (Mobile Reporting Tool)

Wie der Name „Mobile Reporting Tool“ bereits erahnen lässt, handelt es sich bei dem Produkt um ein Werkzeug, mit welchem Aussendienstmitarbeiter auf unkomplizierte Art und Weise ihre Stunden rapportieren können und zwar unabhängig von ihrem Standort.

3.1.1 Ablauf

3.1.1.1 Arbeitseinsatz

Der Aussendienstmitarbeiter erhält von der Zentrale oder direkt von einem Kunden einen Auftrag. Sobald er bei dessen Adresse angelangt ist, drückt er die „Start“-Schaltfläche. Durch das Drücken der Schaltfläche beginnt die Zeitmessung.

Sobald die Arbeit vollbracht ist, teilt der Mitarbeiter dies dem System durch das Drücken der „Stopp“-Schaltfläche mit. Nun wird ihm eine Reihe von potentiellen, bereits registrierten Kunden vorgeschlagen. Der Aussendienstmitarbeiter 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 Mitarbeiter beim Drücken der Stopp-Taste nicht möglich sein sollte (z.B. aus Zeitgründen), eine Beschreibung zu erfassen.

3.1.1.2 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.

3.1.2 Optionale Features

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

3.2 Zweck und Ziel

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

Ein weiteres Ziel ist, mittels geeigneten Werkzeugen wie Redmine, Subversion und Skype gut in der Gruppe zu kommunizieren und zu kooperieren. 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 ausgebaut werden könnte. Es soll die Möglichkeit bestehen, das Projekt flexibel zu erweitern, ohne auf entwicklungsbedingte Grenzen zu stossen.

3.3 Annahmen und Einschränkungen

Für dieses Projekt werden keine Annahmen oder Einschränkungen getroffen.

4 Projektorganisation

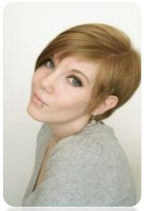
Das Projektteam besteht aus fünf sich gleichgestellten Mitgliedern. In der „Construction“-Phase werden zwei Unterteams gebildet, welche sich jeweils auf einen Tier konzentrieren und das Code-Review für das jeweils andere Team übernehmen.

4.1 Team

4.1.1 Lukas Elmer (Abk. EL)

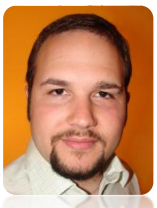
Kenntnisse in:	Ruby on Rails, PHP, Python / Django, Typo3, Wordpress, Java, XHTML, JavaScript, C++, Ubuntu Server
Motivation:	Weitere Erfahrungen im Software-Engineering sammeln. SE1Kenntnisse auf reales Projekt anwenden.
Spezialisierung:	Android-Client
Verantwortlichkeiten:	Coaching bei komplexen Problemen bei Ruby on Rails, Serverunterhalt von Redmine, automatisches Deployment, Design Model, Klassendiagramm(e), Releases, Bedienungsanleitung
Mailadresse :	lelmer@hsr.ch
Skype Adresse:	lukas.elmer

4.1.2 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, Klassenspezifikation, Risikomanagement und Reserveplanung
Mailadresse:	cheidt@hsr.ch
Skype Adresse:	christina_heidt

4.1.3 Diego Steiner (Abk. SD)



Kenntnisse in:	PHP, XHTML, ASP.net, Linux
Motivation:	vorhanden
Spezialisierung:	Webplattform
Verantwortlichkeit:	Überwachung und Erstellung Projektplan, Qualitätsmanagement, Koordination Domainanalyse, Anforderungsspezifikation, Projekt-Präsentation
Mailadresse:	dsteiner@hsr.ch
Skype Adresse:	diego2alps

4.1.4 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, Systemsequenzdiagramme, Dokumentation Architektur
Mailadresse:	dtreichler@hsr.ch
Skype Adresse:	de-lia

4.1.5 Remo Waltenspül (Abk. WR)

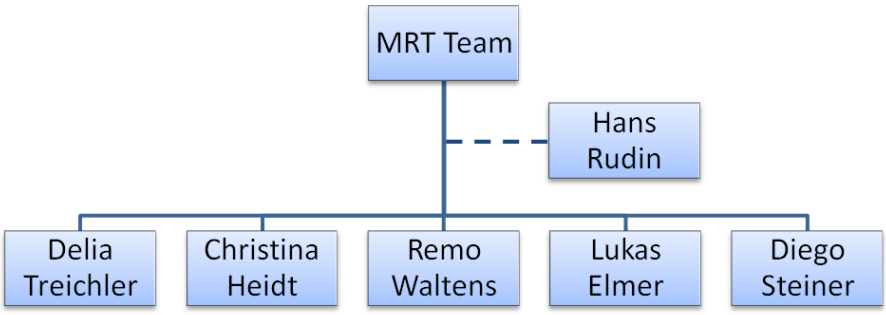
Kenntnisse in:	Java, HTML/CSS, VB, C++
Motivation	Erfahrungen in einem Software-Engineering Projekt sammeln
Spezialisierung:	Webplattform
Verantwortlichkeit:	Operation Contracts, Requirements, System- und Unit Tests, Koordination Architekturprototyp
Mailadresse:	rwaltens@hsr.ch
Skype Adresse:	kobold246

4.2 Anmerkung

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

Die Verantwortlichkeit bedeutet, dass diese Person für diesen Bereich verantwortlich ist. Das bedeutet aber nicht, dass diese Person diesen Posten unbedingt selbst erledigen muss. Sie kann das Arbeitspaket / die Arbeitspakete an andere Teammitglieder delegieren, ist dann aber für diese Arbeit verantwortlich und muss diese überprüfen.

4.3 Organisationsstruktur



4.4 Externe Schnittstellen

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

5 Managementabläufe

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

5.1 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.

5.2 Projektplan

5.2.1 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. Bei verschachtelten Tickets dürfen keine Stunden auf das Parent-Ticket verbucht werden, die erbrachten Stunden sind jeweils auf der zugehörigen Aktivität zu verbuchen.

5.2.2 Fertigstellungsgrad der Arbeiten

Grössere Arbeitspakete werden in Untertickets aufgeteilt. Aufgrund der Schätzungen und der Fertigstellungsgrade der jeweiligen Untertickets wird dann automatisch vom Redmine der Fertigstellungsgrad des grösseren Arbeitspaketes errechnet.

5.2.3 Änderungsverfahren

5.2.3.1 Dokumentationen

Bei Änderungen an einem Dokument schreibt derjenige, der die Anpassung durchgeführt hat, eine kurze Beschreibung, sein Kürzel sowie die neue Version in die vorgesehene Änderungsgeschichte. Anschliessend wird beim Einchecken ins SVN-Repository das passende Ticket referenziert und eine kurze Beschreibung angegeben.

Für weitere Informationen siehe 9.1.2 Subversion / Sourcecode Management und 9.1.5 Dokumentation Guidelines & Review.

5.2.3.2 Code

Falls der Code angepasst wird, wird dies beim Einchecken auf den SVN-Server mit dem gleichen Vorgang wie bei einer Änderung eines Dokuments gehandhabt. Wir verzichten auf Beschreibungen von Änderungen direkt im Code über Kommentarzeilen, da anhand der Revisions-History des Subversions jegliche Änderungen übersichtlich als Differenz dargestellt werden können.

Ebenfalls wichtig beim Code ist, dass beim committen des Codes alle Unit Tests (siehe 9.3.1 Unit Tests) fehlerfrei laufen müssen.

Für weitere Informationen siehe 9.1.2 Subversion / Sourcecode Management.

5.2.4 Iterationsplanung / Milestones

Die detaillierte Iterationsplanung und auch die aktuellen Milestones befinden sich im Redmine. Da die Milestones und die Iterationen sehr wichtig sind, wurden sie nachfolgend detailliert beschrieben.

Für weitere Details zu den Arbeitspaketen, die für die einzelnen Iterationen / Milestones geplant sind, siehe 7 Arbeitspakete.

5.2.4.1 Inception 21.02.-10.03.11 / SW01-SW03

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

5.2.4.2 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.

MS2, Elaboration E1 / 11.03.-31.03.11 / SW04-SW06	Termine

- Vision
- Anforderungsspezifikation
  ○ Use Cases
  ○ Nichtfunktionale Anforderungen
- Domainanalyse
  ○ Domain Model
  ○ Klassenspezifikation
  ○ SSDs
  ○ Contracts
	29.03.11: MS2 - Abgabe Analyse

MS3, Elaboration E2 / 01.04.-14.04.11 / SW07-SW08	Termine

- Architekturprototyp (V0.1, siehe 5.2.6 Abgabe)
  ○ Lauffähig
  ○ Über alle Layers und Tiers
  ○ Technische Risiken abgedeckt
  ○ Architektur grob dokumentiert
  ○ Klassendiagramm(e)
  ○ 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

5.2.4.3 Construction 15.04.-12.05.11 / SW09-SW11

In der Construction Phase stehen tatsächlich nur drei und nicht vier Wochen zur Verfügung, da in dieser Zeit eine Woche Ferien sind. Deshalb wird für die Construction Phase auch nur eine Iteration geplant. Für ein grösseres Projekt wäre das natürlich viel zu wenig, aber im SE2 Projekt liegt der Fokus ja primär auf dem Software Engineering und nicht auf der Implementation selbst.

MS4, Construction C1	Termine

- Architektur / Design inhaltlich
  o Physische Architektur
  o Logische Architektur
  o Schnittstellen zw. Packages
  o Persistenz
  o Design Packages
  o Externes Design
  o Besonderheiten des Designs
  o Eingesetzte Technologien
  o Dokumentation der Architektur
- Interner Release der V0.2 (siehe 5.2.6 Abgabe)
	10.05.11: MS4 -Abgabe Construction

5.2.4.4 Transition 13.05.-03.06.11 /SW12-SW14

MS5, Transition T1	Termine

- Release der V1.0 (siehe 5.2.6 Abgabe)
- Deployment V1.0 ausführen
- Präsentation
  o Fertigstellen
  o Präsentieren
  o 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

5.2.5 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 externe Dokumente: Sitzungsprotokolle).

5.2.6 Abgabe

Folgende Versionen sind vorgesehen:

Version	Typ	Beschreibung	Datum
0.1	Architekturprototyp	Basissystem, Datenstrukturen	12.04.11
0.2	Testversion	Businesslogic, UI, Schnittstellen, evt. Feature Freeze	10.05.11
1.0	Finalversion	Bugs gefixt, evt. optionale Features (falls Zeit vorhanden), Fertige Version	31.05.11

5.2.7 Deployment

5.2.7.1 Android

Die Androidapplikation muss schlussendlich auf einem Android Endgerät laufen. Deshalb wird für jede Version ein APK Paket generiert und auf dem Endgerät installiert. Die Tools für das Deployment sind im Android-SDK enthalten.

5.2.7.2 Rails

Da die Webapplikation schliesslich auf einem Server läuft, muss diese, nach der Entwicklung und dem Testen des Quellcodes, deployed 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.

6 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.

Diejenigen Risiken, die eliminiert werden können (R04, R05, R06), werden in bis zu den beschriebenen Phasen (pro Risiko verschieden) überprüft und eliminiert. Dazu befinden sich Tickets im Redmine.

Falls neue Risiken auftauchen sollten werden diese im externen Dokument erfasst. Dafür existiert ein spezielles Arbeitsblatt im Dokument, auf dem die Eingangsdaten der Risiken eingeschrieben werden. Nachdem sie in dem Dokument erfasst worden sind, werden sie dann auch im Redmine als Ticket angelegt und geplant.

7 Arbeitspakete

Die aktuellen Beziehungen zwischen den einzelnen Arbeitspaketen und den dazugehörigen Iterationen sind im Redmine ersichtlich. Alle Arbeitspakete werden im Redmine aktuell gehalten. Untenstehend wurden die einzelnen Gantt Diagramme vom Redmine zur Übersicht dokumentiert. Wir empfehlen aber, die Gantt Diagramme im Redmine anzusehen, da diese dort grafisch noch ein wenig anspruchsvoller angezeigt werden und auch noch über detailliertere Informationen verfügen.

Für weitere Details zu den einzelnen Iterationen / Milestones und deren Abgaben siehe 5.2.4 Iterationsplanung / Milestones.

7.1 MS1

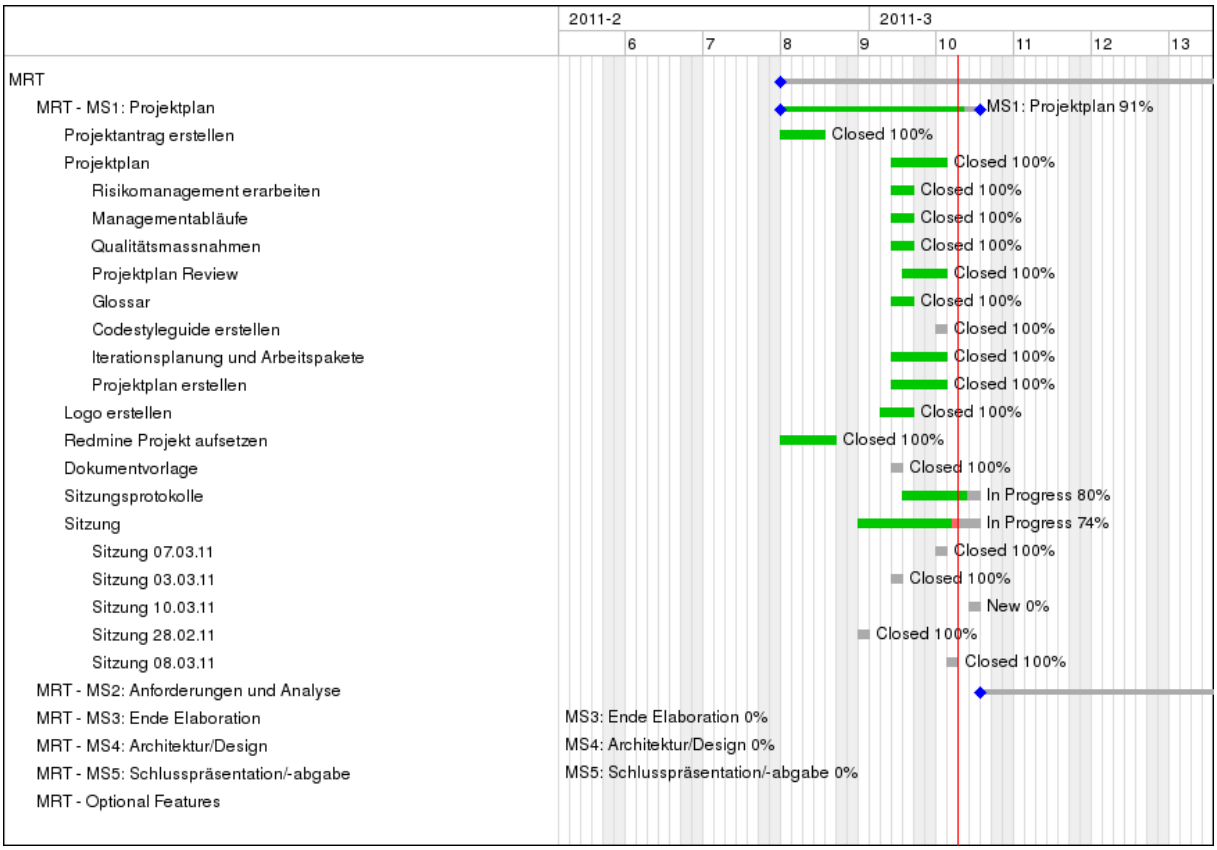


Abbildung 1 Gantt Diagramm MS1

7.2 MS2

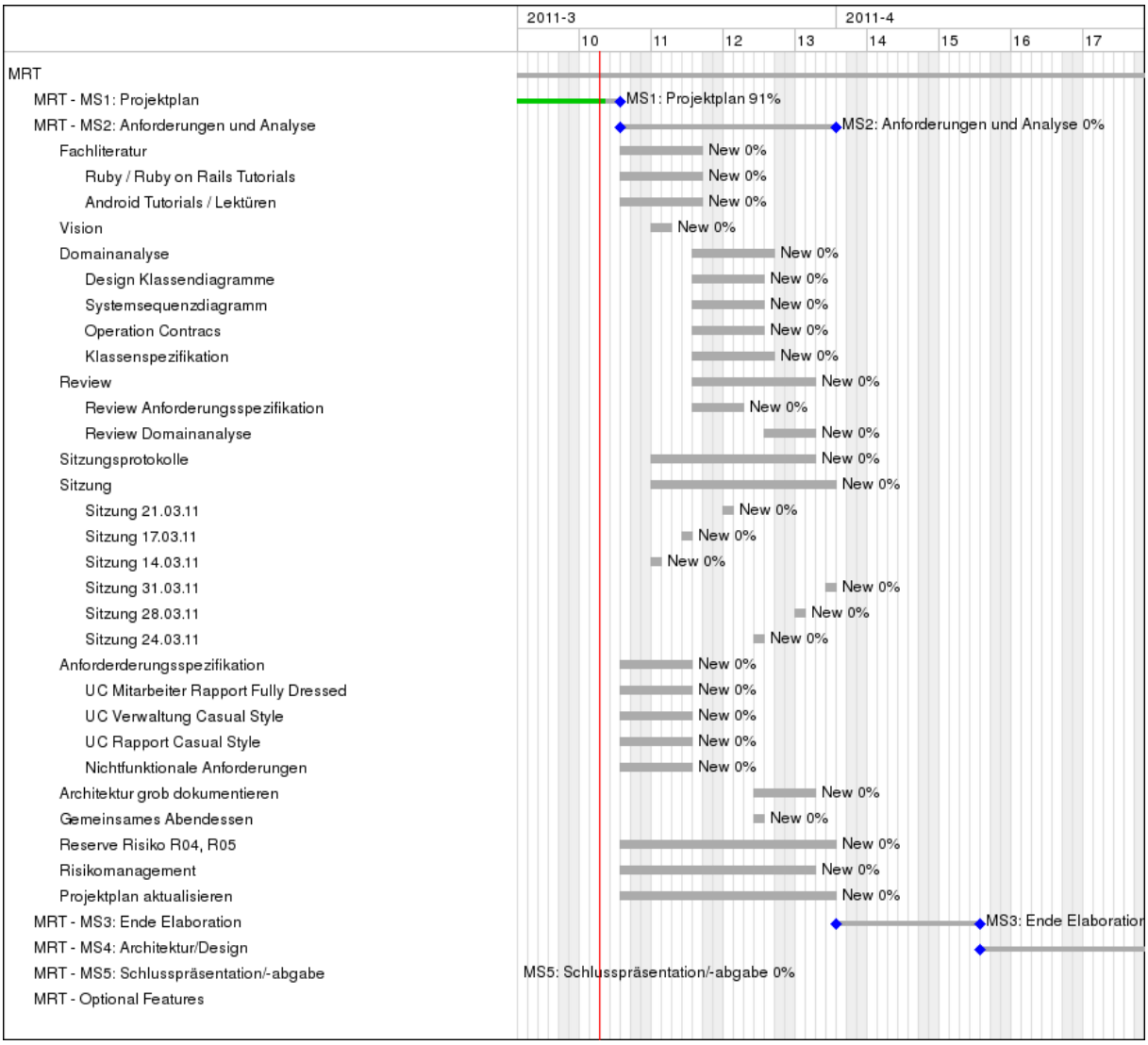


Abbildung 2 Gantt Diagramm MS2

7.3 MS3

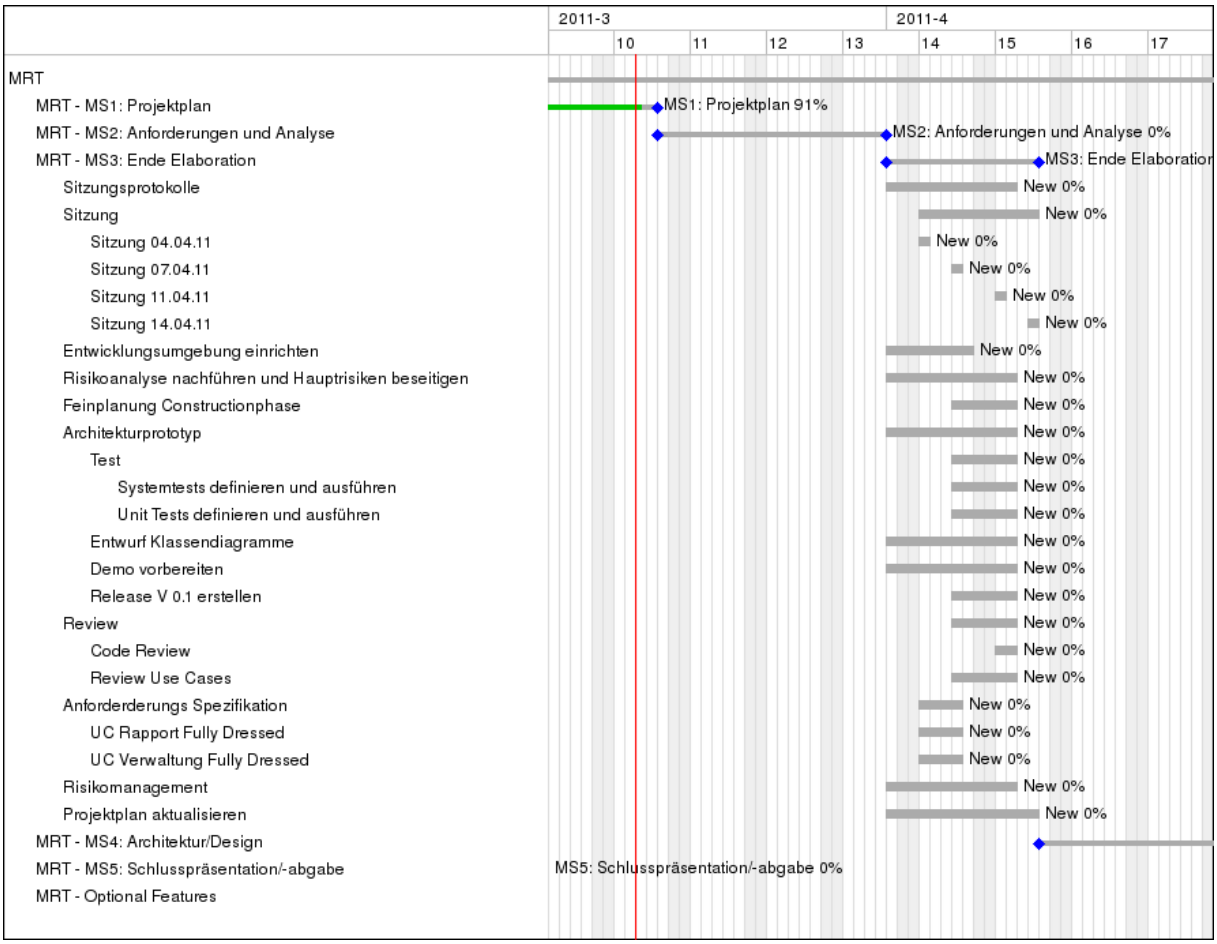


Abbildung 3 Gantt Diagramm MS3

7.4 MS4

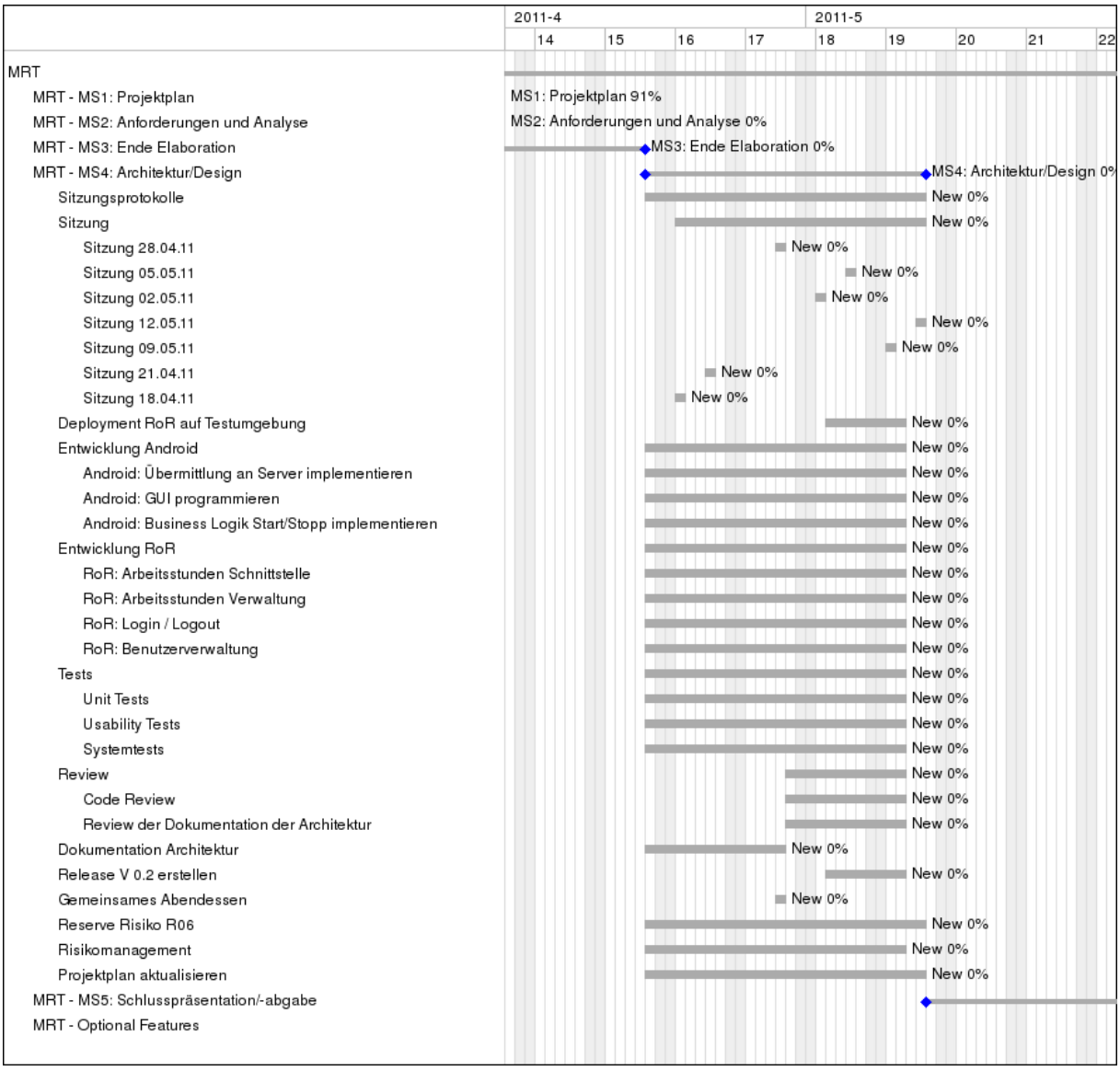


Abbildung 4 Gantt Diagramm MS4

7.5 MS5

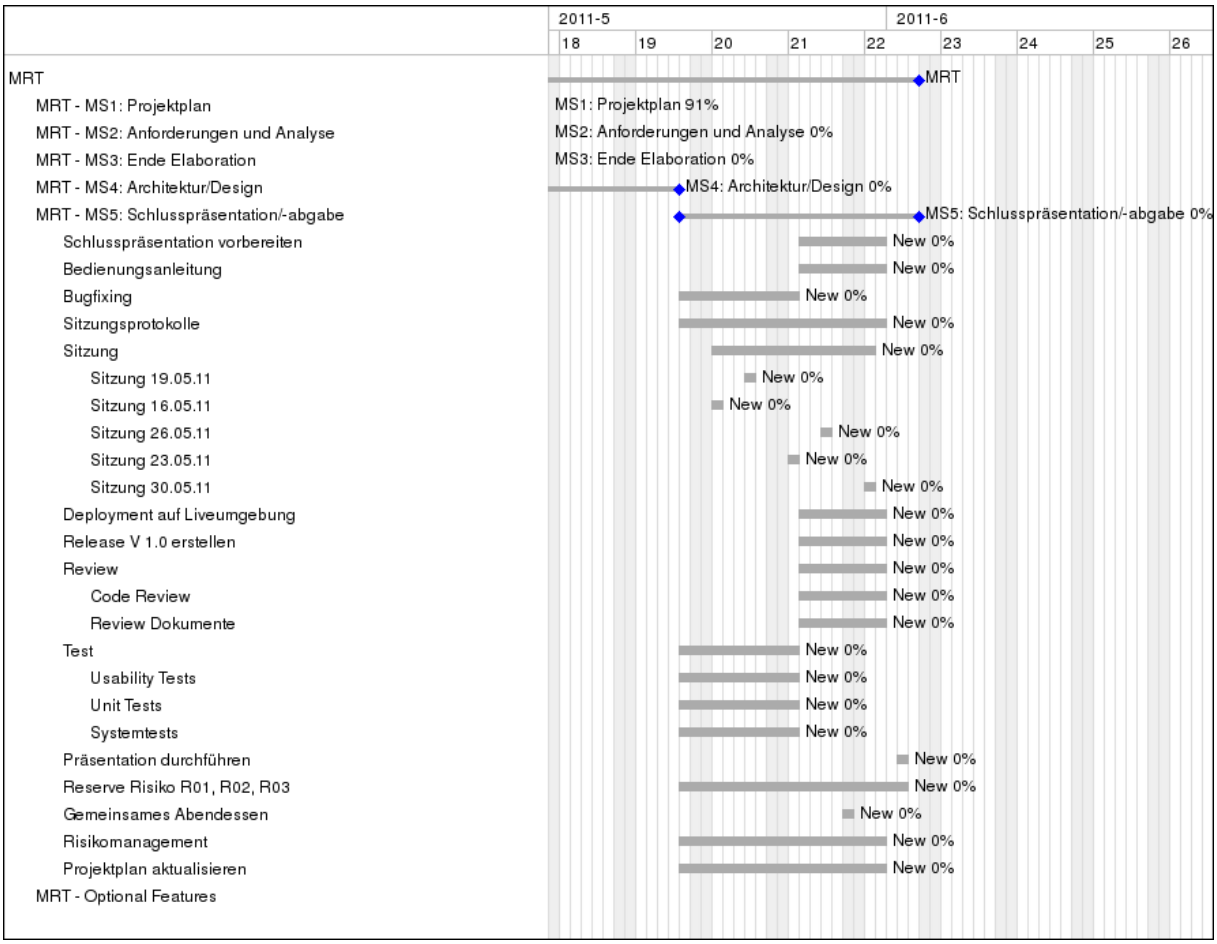


Abbildung 5 Gantt Diagramm MS5

8 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.

8.1 Software

Folgende Software kommt zum Einsatz:

Software	Version	Beschreibung / Einsatzbereich
Eclipse IDE	Helios Service Release 1	Javabasierte IDE zur Entwicklung von Software. Wird für die Entwicklung der Android Applikation verwendet.
NetBeans IDE	6.9.1	Javabasierte IDE zur Entwicklung von Software. Wird für die Entwicklung der Rails Applikation verwendet.
Java	6 bzw. JRE 1.6.x	
Subversion	1.6.x	Sourcecontroll Management System für die Versionierung von Dokumenten und Code. Wird zur Versionierung von Sourcecode und Dokumentation verwendet.
Android SDK	3.0	Toolkit zur Entwicklung einer Android Applikation.
Ruby	1.8.7	
Ruby on Rails	3.0.x	Rubybasiertes Framework zur Entwicklung von Web-Plattformen. Wird für die Entwicklung der Web-Plattfrom verwendet.
Apache	2.2.16	Etablierter Webserver mit diversen Erweiterungsmöglichkeiten. Ermöglicht Mod Passanger und wird als Web-Server verwendet für SVN, Redmine und die Rails Applikation.
Mod Passanger	2.2.15	Modul für Apache zur Anbindung an Rails-Applikationen.
Redmine	1.1.2	Railsbasiertes Projektmanagementtool. Wird für das Projektmanagement, das Wiki und den Stundenrapport verwendet.
Skype	Aktuellste Version	VoIP Programm mit Gruppenchat Funktion zur ortsunabhängigen Kommunikation innerhalb des Teams.
MS Office	2007, 2010	Suite zur Erstellung von Dokumenten wie Textdokumenten oder Tabellenkalkulationen. Wird fürs Projektmanagement und jegliche Art von Dokumenten verwendet (z.B. Projektplan).
Enterprise Architect	7.5	Modellierungstool, wird für die Erstellung diverser Artifacts (z.B. Domain Model, SSD, etc.) verwendet.
Google Documents	Aktuellste Version	Online Office, wird verwendet, falls Dokumente gemeinsam und gleichzeitig erarbeitet werden müssen.

8.2 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 5.2.5 Besprechungen)
- Skype

8.2.1 Wiki Redmine

Das Wiki wird für folgende Dinge verwendet:

- Wichtige Informationen persistent speichern (Bsp. URL für SVN Server)
- Austausch zwischen Teammitgliedern
- Fragen, die abgeklärt / besprochen werden müssen
- Fragen an den Betreuer
- Kleinere Checkliste für die Abgabe

9 Qualitätsmassnahmen

9.1 Allgemein

9.1.1 Regelmässige Teamsitzungen und teamfördernde Massnahmen

Neben den regelmässigen Teamsitzungen (siehe 5.2.5 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.

9.1.2 Subversion / Sourcecode Management

Der gesamte Sourcecode wie auch die Dokumentation wird im 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 Tickets committed werden kann. Die URL des Subversion Repositories lautet: https://se2p_svn.elmermx.ch/. Jedes Projektmitglied hat seine persönlichen Zugangsdaten bereits erhalten.

Grundsätzlich wird bei jedem Commit, wo es sinnvoll ist, die Ticketnummer gleich im Kommentar angegeben, damit der Commit im Redmine einem Ticket zugeordnet werden kann. Das Format für den Kommentar ist Folgendes: *refs #<TicketID><Kommentar>.*

9.1.3 Issuetracking

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

9.1.4 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.

9.1.5 Dokumentation 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.

9.2 Codequalität

9.2.1 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.

9.2.2 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 der IDE benutzt, abgesehen von der Anpassung der Breite (200 statt 80 Zeichen).

9.2.2.1 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

9.2.2.2 Android

- Paketnamen werden klein geschrieben
- Methoden werden klein und camelCase geschrieben
- Klassen werden gross und CamelCase geschrieben
- Konstanten / StaticVariabeln 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.

9.2.2.2.1 Codeformatierung

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

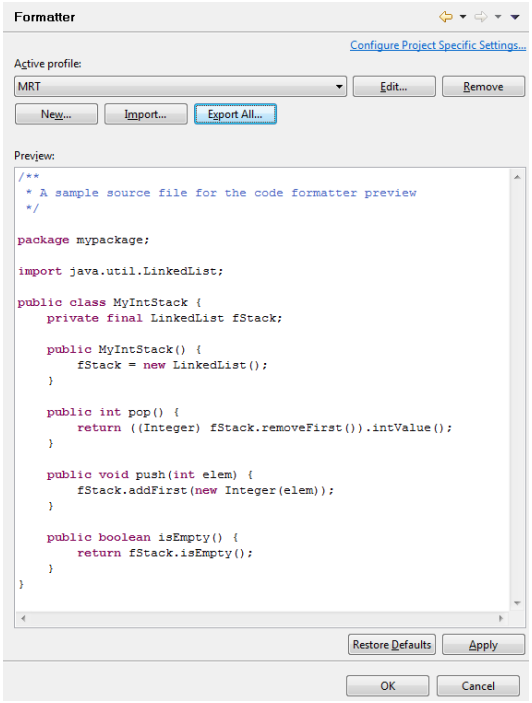


Abbildung 6 Codeformatierung Android

9.2.2.3 Webplattform (RoR)

- Methoden werden klein_und_mit_unterstrich geschrieben
- Klassen werden gross und CamelCase geschrieben
- Konstanten / StaticVariabeln 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.

9.2.2.3.1 Codeformatierung

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

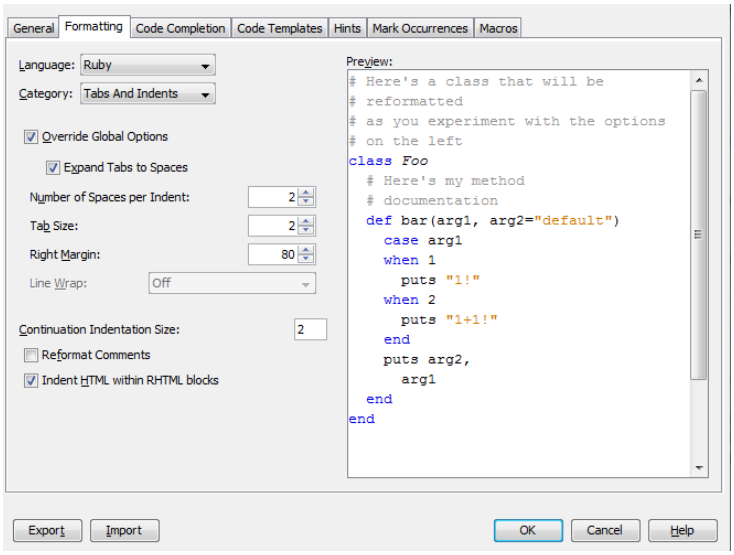


Abbildung 7 Codeformatierung Ruby on Rails

9.2.3 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.

9.3 Tests

9.3.1 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 „raketest“).

9.3.1.1 Android

Auf Android wird JUnit 3 eingesetz. Es ist jedoch nicht direkt TestCase (JUnit Standard) sondern AndroidTestCase (Android spezifisch, erbt von TestCase) zu verwenden.

9.3.1.2 Rails

Das Ruby on Rails Framework unterstützt Unit Tests für Models. Deshalb werden diese mit Unit Test getestet, das Entsprechende Modul heisst test_helper.

Da im Ruby on Rails Teil die Controller (Klassen) per Functional Tests getestet werden, werden statt Unit Tests im Rails Teil für die Controller Functional Tests geschrieben. Da die Functional praktisch den Unit Tests für die Controller entsprechen, fallen die Functional Tests in die Arbeitspakete / Kategorie der Unit Tests.

9.3.2 Systemtests

Am Ende der Milestones 3, 4 und 5 werden umfassende Systemtests durchgeführt. Dazu wird im Vorfeld ein Testskript entworfen, für das alle Ergebnisse in einem Testprotokoll festgehalten werden. Dieses Protokoll bildet dann die Grundlage für die „Bugfixing“-Arbeitspakete

9.3.3 Usability Tests

In der Elaboration Phase werden Paper-Prototyping Methoden eingesetzt, falls in dieser Phase genügend Zeit bleibt (optional, die Iteration ist noch nicht genau definiert). Im diesem Fall wird das entsprechende Ticket einer entsprechenden Iteration zugewiesen.

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