

Projekt: Rupert  
  
Projektplan

Daunicht Nils (ndaunich@hsr.ch)

Fröhlich Adrian (a1froehl@hsr.ch)

Hochreutener Joel(jhochreu@hsr.ch)

Peisker Dominic(dpeisk@hsr.ch)

# Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 26.02.2015 | 1.0.0 | Erstellung | Adrian Fröhlich, Joel Hochreutener, Nils Daunicht, Dominic Peisker |
| 27.02.2015 | 1.1.0 | Anpassungen und Konkretisierungen | Dominic Peisker, Nils Daunicht, Adrian Fröhlich, Joel Hochreutener |

# Inhalt

[Änderungsgeschichte 2](#_Toc412810716)

[Inhalt 3](#_Toc412810717)

[1. Einführung 4](#_Toc412810718)

[1.1 Zweck 4](#_Toc412810719)

[1.2 Gültigkeitsbereich 4](#_Toc412810720)

[1.3 Referenzen 4](#_Toc412810721)

[2. Projekt Übersicht 5](#_Toc412810722)

[2.1 Zweck und Ziel 5](#_Toc412810723)

[2.2 Lieferumfang 5](#_Toc412810724)

[2.3 Annahmen und Einschränkungen 5](#_Toc412810725)

[3. Projektorganisation 6](#_Toc412810726)

[3.1 Organisationsstruktur 6](#_Toc412810727)

[3.2 Externe Schnittstellen 6](#_Toc412810728)

[4. Management Abläufe 7](#_Toc412810729)

[4.1 Kostenvoranschlag 7](#_Toc412810730)

[4.2 Zeitliche Planung 7](#_Toc412810731)

[4.2.1 Phasen / Iterationen 7](#_Toc412810732)

[4.2.2 Meilensteine 7](#_Toc412810733)

[4.3 Besprechungen 8](#_Toc412810734)

[5. Risikomanagement 9](#_Toc412810735)

[5.1 Risiken 9](#_Toc412810736)

[5.2 Umgang mit Risiken 9](#_Toc412810737)

[6. Arbeitspakete 10](#_Toc412810738)

[7. Infrastruktur 11](#_Toc412810739)

[7.1 Übersicht der Tools 11](#_Toc412810740)

[8. Qualitätsmassnahmen 12](#_Toc412810741)

[8.1 Allgemeine Massnahmen 12](#_Toc412810742)

[8.1.1 Kommentieren des Codes 12](#_Toc412810743)

[8.1.2 Teamsitzungen 12](#_Toc412810744)

[8.1.3 4-Augen-Prinzip 12](#_Toc412810745)

[8.1.4 Versionskontrolle 12](#_Toc412810746)

[8.1.5 Continuous Integration 12](#_Toc412810747)

[8.2 Dokumentation 12](#_Toc412810748)

[8.3 Projektmanagement 12](#_Toc412810749)

[8.4 Entwicklung 13](#_Toc412810750)

[8.4.1 Vorgehen 13](#_Toc412810751)

[8.4.2 Code Reviews 13](#_Toc412810752)

[8.4.3 Code Style Guidelines 13](#_Toc412810753)

[8.5 Testen 13](#_Toc412810754)

[8.5.1 Unit Test 13](#_Toc412810755)

[8.5.2 Integration Test 13](#_Toc412810756)

[8.5.3 Systemtest 13](#_Toc412810757)

# Einführung

## Zweck

Das Dokument beschreibt die Planung für die Entwicklung eines Spiels das unter dem Projektnamen Rupert Entwickelt wird.

## Gültigkeitsbereich

Das Dokument ist während des SE2 Projektes im Frühlingsemester 2015 und im Zusammenhang mit der Arbeit am Projekt Rupert gültig.

## Referenzen

Redmine Server: <http://sinv-56076.edu.hsr.ch/redmine/login?back_url=http%3A%2F%2Fsinv-56076.edu.hsr.ch%2Fredmine%2F>, Username/Passwort: guest/guest

Google Code Style Guidelines: <https://google-styleguide.googlecode.com/svn/trunk/javaguide.html>

# Projekt Übersicht

Das Ziel unseres Projektes ist das Entwickeln eines rundenbasierten Strategiespiels. Im Spiel werden zwei Spieler über eine Client-Server Architektur gegeneinander antreten. Die Spieler kommen nacheinander an die Reihe und können während ihrem Zug Aktionen für ihre Einheiten festlegen. Wenn ein Spielereine Einheit des Gegners angreift, wählen beide Spieler eine Kampfaktion für Ihre beteiligte(-n) Einheit(-en), wie z.B. Angriff oder Verteidigen. Aufgrund der gewählten Aktionen wird der Ausgang des Kampfes berechnet. Als Spielfeld wollen wir eine schachfeldartige Karte gestalten. Zu Beginn wollen wir drei Einheitentypen erstellen. Diese sollen Fernkampf, Nahkampf und Kavallerie darstellen. Bei der Entwicklung legen wir jedoch Wert auf eine einfache Erweiterbarkeit.

## Zweck und Ziel

Unser SE2 Projekt ist es, ein rundenbasiertes Strategiespiel zu programmieren. Als Team haben wir zum Ziel, ein Spiel zu entwickeln, dass Spass macht und einfach zu bedienen ist.

## Lieferumfang

Im Rahmen des Projektes Rupert wird folgendes erstellt:

* Anwendung
  + Server
  + Client
* Installationsanleitung
* Quellcode
* Javadoc
* Usermanual

## Annahmen und Einschränkungen

Tests werden nur auf Rechnern mit Windows Durchgeführt. Wir nehmen an das jedes Teamitglied pro Woche ca. 8 bis 10 Stunden am Projekt Arbeiten wird, was bis zum Schluss zu einer Gesamtarbeitszeit von 480 bis 600 Stunden führt.

# Projektorganisation

## Organisationsstruktur

|  |  |
| --- | --- |
| Name | Verantwortung |
| Herr Farhad Mehta | Projektbetreuung |
| Nils Daunicht | Projektleitung, Entwickler |
| Joel Hochreutener | Entwickler |
| Dominic Peisker | Entwickler |
| Adrian Fröhlich | Entwickler |

## Externe Schnittstellen

Projektbetreuer: Herr Farhad Mehta, [fmehta@hsr.ch](mailto:fmehta@hsr.ch)

Serverbetreuung: IT Service Desk HSR, [i-support@hsr.ch](mailto:i-support@hsr.ch)

# Management Abläufe

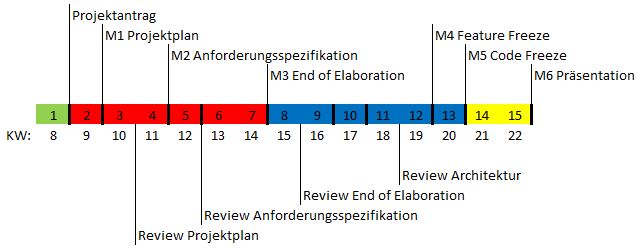
Unsere Planung wird primär im Redmine realisiert und ist unter folgendem Link erreichbar: <http://sinv-56076.edu.hsr.ch/redmine/projects/rupert/issues/gantt>

## Kostenvoranschlag

Das Projekt gibt 4 ECTS Punkte. Bei einem Aufwand von 30h pro ECTS Punkt ergibt dies 120h pro Person. Der Gesamtaufwand des Teams wird somit etwa 480h betragen. Das Projekt läuft über das ganze Semester, was 15 Wochen entspricht. Wir rechnen mit einer Ferienwoche über Ostern. Somit beträgt der Aufwand pro Woche und Person etwa 8.5h.

## Zeitliche Planung

Anhand unserer Planung ergibt sich folgender grober Zeitplan:



Die zeitliche Planung pro Arbeitspaket wird im Redmine erfasst und im Excell-Sheet (Zeitplanung.xlsx) nachgetragen. Die Mitglieder erfassen ihre Stunden im Excell-Sheet (Zeitplanung.xlsx) und führen diese pflichtbewusst nach, da Redmine uns in dieser Hinsicht zu wenig Übersicht bietet.

### Phasen / Iterationen

Die Entwicklung wird nach RUP in die Phasen Inception, Elaboration, Construction und Transition aufgeteilt. Die Einzelnen Iterationen tragen die Namen der Phasen, wobei die Iterationen innerhalb einer Phase durchnummeriert werden. Die Iterationen werden in Redmine verwaltet. Wir streben ein- bis zweiwöchige Iterationen an. Die konkreten Namen der Iterationen sowie die dazugehörenden Arbeitspakete sind im Abschnitt 6, Arbeitspakete, beschrieben.

### Meilensteine

|  |  |  |
| --- | --- | --- |
| Datum | Meilenstein | Beschreibung |
| 27.02.2015 | Projektplan | Abgabe des Projektplanes |
| 13.03.2015 | Anforderungsspezifikation | Die Anforderungsspezifikationen des Projektes sind Ausgearbeitet. |
| 03.04.2015 | End of Elaboration | Die Planung des Projektes ist abgeschlossen. Alle Planungsdokumente sind erstellt und es kann mit der Implementierung begonnen werden. |
| 08.05.2015 | Feature Freeze | Es werden keine neuen Features mehr ins Projekt aufgenommen. Festlegung, was in den letzten Wochen der Construction-Phase noch erreicht werden soll. |
| 15.05.2015 | Code Freeze | Abschluss der Programmierung. Programm soll lauffähig und getestet sein. |
| 28.05.2015 | Präsentation | Schlusspräsentation ist vorbereitet und gehalten. |

## Besprechungen

|  |  |  |
| --- | --- | --- |
| Datum | Review | Beschreibung |
| 05.03.2015 | Projektplan | Review des Projektplanes durch Herrn Mehta. |
| 19.03.2015 | Anforderungsspezifikation | Review der Anforderungsspezifikationen des Projekts Rupert. |
| 09.04.2015 | End of Elaboration | Zwischenpräsentation mit einem Architekturprototypen sowie ein Review. |
| 30.04.2015 | Architektur | Review der Architektur und des Designs. |

# Risikomanagement

## Risiken

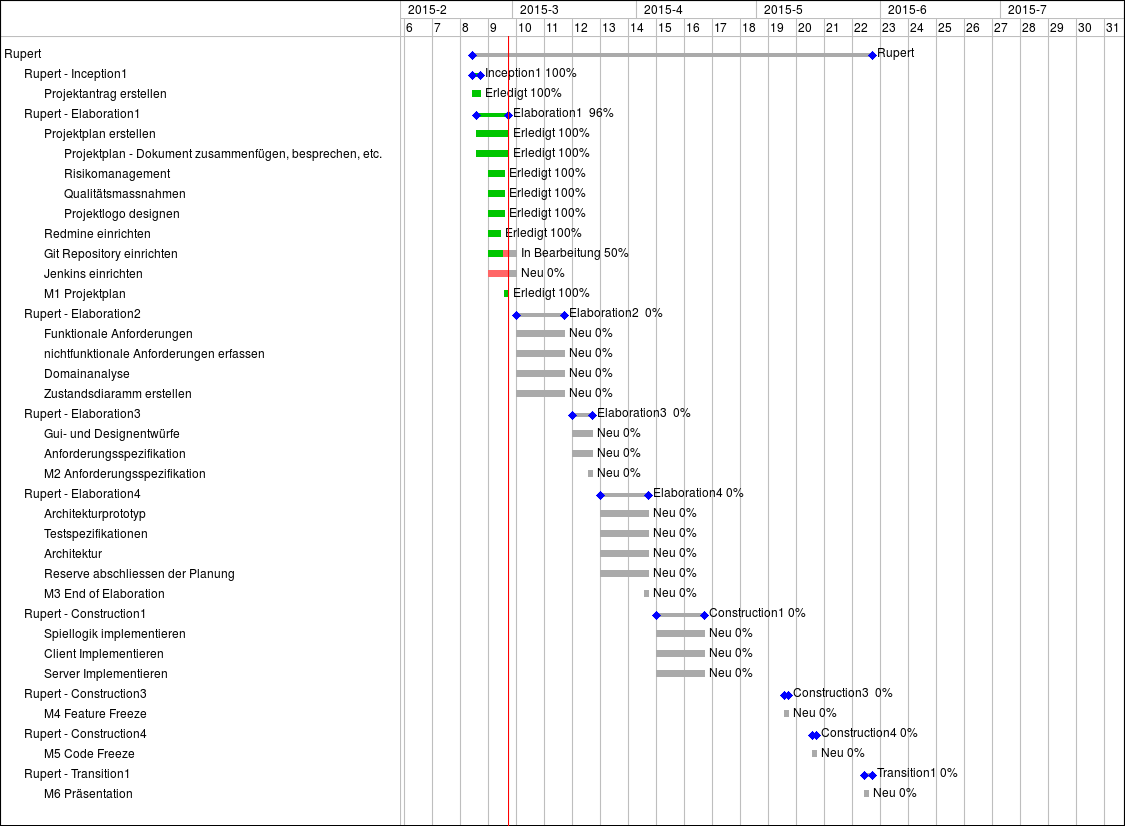
Eine Zusammenstellung der Risiken sowie Massnahmen zur Vermeidung des Eintretens bzw. zum Verhalten beim Eintreten befindet sich im Excel-Sheet Dokument Technische Risiken.

## Umgang mit Risiken

Wann immer möglich wollen wir durch Qualitätssicherung und sonstige Vorbeugung vermeiden, dass die Risiken überhaupt eintreten können.

# Arbeitspakete

Gantt Stand 27.02.15



# Infrastruktur

Jedes Teammitglied arbeitet an seinem eigenen Laptop oder PC. Dabei ist auf den Geräten jeweils Windows 7 oder 8 Installiert. Als Build- und Testserver wird ein virtueller Server eingesetzt, welcher von der HSR zu Verfügung gestellt wird.

## Übersicht der Tools

Die folgende Tabelle listet die eingesetzten Tools auf. Falls notwendig, sind die Zugangsdaten angegeben.

|  |  |  |  |
| --- | --- | --- | --- |
| Bezeichnung | Beschreibung | Link / Zugangsdaten | Version |
| Git | Versionsverwaltungstool | Offizielle Webseite des Tools: <http://git-scm.com/> |  |
| Eclipse | Entwicklungsumgebung | Offizielle Webseite des Tools:  <http://www.eclipse.org/downloads/> | Luna |
| Gimp 2 | Bildbearbeitungstool | Offizielle Webseite des Tools:  <http://www.gimp.org/> | 2.8 |
| Jenkins | Buildserver | <http://sinv-56076.edu.hsr.ch/jenkins/>  Offizielle Webseite des Tools: <http://jenkins-ci.org/> | 1.599 |
| Vector Magic | Bildbearbeitungstool | Offizielle Webseite des Tools:  <http://vectormagic.com/home> |  |
| Astah Community | UML-Modellierungstool | Offizielle Webseite des Tools: <http://astah.net/editions/uml> | Version 37 |
| Office Word 13 | Textbearbeitungstool | Offizielle Webseite des Tools:  https://products.office.com/de-de/home | 15.xx |
| Redmine | Projektplanungstool | <http://sinv-56076.edu.hsr.ch/redmine>  Zugang:  Benutzer: **guest**  Passwort: **guest**  Offizielle Webseite des Tools: <http://redmine.org> | 2.6.2 |
| Inkscape | Bildbearbeitungstool | Offizielle Webseite des Tools: <https://inkscape.org/de> | 0.91 |

# Qualitätsmassnahmen

## Allgemeine Massnahmen

### Kommentieren des Codes

Durch ein gutes Kommentieren wird gewährleistet, dass alle Teammitglieder schnell die Funktionsweise und Logik des entsprechenden Codeabschnittes erkennen und damit weiterarbeiten können. Diese Massnahme wird während dem gesamten Projekt umgesetzt.

### Teamsitzungen

Mindestens einmal pro Woche sitzen wir als Team zusammen und besprechen die Fortschritte jedes einzelnen. Dabei werden auch gleich Fragen geklärt und einzelnen Teammitgliedern bei Unklarheiten geholfen. Zudem werden die als nächstes fälligen Arbeiten diskutiert und allfällige Probleme behoben. Voraussichtlich werden diese Treffen am Montag ab 15:00 Uhr und am Donnerstag ab 10:00 Uhr stattfinden. Mit dieser Massnahme wollen wir die Zusammenarbeit im Team fördern sowie Probleme frühzeitig erkennen und als Gruppe lösen.

### 4-Augen-Prinzip

Bei allen kritischen Arbeiten, wie zum Beispiel das Erstellen eines Dokumentes oder dem Schreiben von Code, kontrolliert mindestens ein weiteres Gruppenmitglied die Arbeit auf Korrektheit und Vollständigkeit. Dies geschieht durch das erfassen entsprechender Tickets im Redmine.

### Versionskontrolle

Durch den Einsatz von Git als Tool zur Versionenkontrolle stellen wir sicher, dass jedes Teammitglied Zugriff auf den aktuellen Code beziehungsweise auf die aktuellen Dokumente hat. Auch können so Änderungen zurückverfolgt werden und bei technischen Problemen eines Gruppenmitgliedes gibt es keinen Datenverlust.

### Continuous Integration

Durch den Einsatz von Jenkins als Build Server und der Bugtrackingfunktion von Redmine stellen wir sicher, dass wir zu jeder Zeit wissen, welche Version lauffähig ist beziehungsweise welche Bugs noch zu beheben sind.

## Dokumentation

Die Dokumente befinden sich auf dem Git Server. Zudem wird periodisch ein Backup erstellt, um auch bei einem Ausfall des Servers keinen Datenverlust zu erleiden. Die Code-Dokumentation erfolgt über JavaDoc.

## Projektmanagement

Für das Projektmanagement setzen wir auf das in SE1 vorgestellte Redmine. Sämtliche Arbeitspakete werden dort erfasst, priorisiert und einem Teammitglied zugewiesen. Der Server ist unter der Adresse <http://sinv-56076.edu.hsr.ch/redmine/login?back_url=http%3A%2F%2Fsinv-56076.edu.hsr.ch%2Fredmine%2F> erreichbar. Das Login für Gastbenutzer ist Username/Passwort: guest/guest.

## Entwicklung

Der Source Code wird auf einem Git Repository gespeichert. Die Qualität garantieren wir, indem jedes Teammitglied seinen neu geschriebenen Code zuerst lokal Testet und nur Commited, wenn keine Fehler mehr Auftauchen. Zudem planen wir auch beim Code ein 4-Augen-Prinzip, siehe Punkt 8.1.3.

### Vorgehen

Sobald ein Gruppenmitglied relevanter Code oder Dokumente auf das Git Repo commited, erfasst es im Redmine ein Arbeitspaket mit der Aufgabe, den gepushten Code bzw. Dokument gegenzulesen. Zudem ist bei jedem Commit eine aussagekräftige Commit-Message anzugeben.

### Code Reviews

Wie bereits unter 8.1.3 geschrieben wird jeder Relevante Code von einem Teammitglied gegengelesen.

### Code Style Guidelines

Als Code Style Guideline werden wir uns an die von Google definierten Richtlinien halten (<https://google-styleguide.googlecode.com/svn/trunk/javaguide.html>). Namen werden grundsätzlich in Englisch geschrieben, Kommentare auf Deutsch.

## Testen

### Unit Test

Für die Unit Tests verwenden wir das Framework JUnit. Grundsätzlich ist jedes Teammitglied verantwortlich, dass zu jeder Klasse ausreichend relevante Unit Tests geschrieben werden. Die Test-Coverage werden durch das Coverage Tool EclEmma sichergestellt.

### Integration Test

Integration Tests werden regelmässig auf dem Build Server durchgeführt.

### Systemtest

Tests werden voraussichtlich nach jeder Iteration in der Construction Phase durchgeführt. Zuerst werden wir in der Gruppe den Code Testen. Dabei achten wir auf die Benutzbarkeit sowie auf das Erfüllen der Anforderungen. Gefundene Bugs werden zum Beheben im Redmine erfasst.

Zudem planen wir Mitstudenten unser Programm testen zu lassen. Dadurch erhalten wir eine unabhängige Meinung zum Projekt und es tauchen evtl. Fehler auf, die wir selbst nicht bemerkt hätten.