

Projekt: JBomberman  
  
Projektplan

Pascal Kistler

Silvan Adrian

Fabian Binna

1. Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Version** | **Änderung** | **Autor** |
| **23.02.15** | 1.00 | Erstellung des Dokuments | Gruppe |
| **24.02.15** | 1.01 | Neue Texte eingefügt | Fabian Binna |
| **25.02.15** | 1.02 | Logo + Neue Texte | Silvan Adrian |

1. Inhalt

[Änderungsgeschichte 2](#_Toc317331516)

[Inhalt 3](#_Toc317331517)

[1. Einführung 4](#_Toc317331518)

[1.1 Zweck 4](#_Toc317331519)

[1.2 Gültigkeitsbereich 4](#_Toc317331520)

[1.3 Referenzen 4](#_Toc317331521)

[2. Projekt Übersicht 5](#_Toc317331522)

[2.1 Zweck und Ziel 5](#_Toc317331523)

[2.2 Lieferumfang 5](#_Toc317331524)

[2.3 Annahmen und Einschränkungen 5](#_Toc317331525)

[3. Projektorganisation 6](#_Toc317331526)

[3.1 Organisationsstruktur 6](#_Toc317331527)

[3.2 Externe Schnittstellen 6](#_Toc317331528)

[4. Management Abläufe 7](#_Toc317331529)

[4.1 Kostenvoranschlag 7](#_Toc317331530)

[4.2 Zeitliche Planung 7](#_Toc317331531)

[4.2.1 Phasen / Iterationen 7](#_Toc317331532)

[4.2.2 Meilensteine 7](#_Toc317331533)

[4.3 Besprechungen 7](#_Toc317331534)

[4.4 Abgabe 7](#_Toc317331535)

[5. Risikomanagement 8](#_Toc317331536)

[5.1 Risiken 8](#_Toc317331537)

[5.2 Umgang mit Risiken 8](#_Toc317331538)

[6. Arbeitspakete 9](#_Toc317331539)

[7. Infrastruktur 10](#_Toc317331540)

[8. Qualitätsmassnahmen 11](#_Toc317331541)

[8.1 Dokumentation 11](#_Toc317331542)

[8.2 Projektmanagement 11](#_Toc317331543)

[8.3 Entwicklung 11](#_Toc317331544)

[8.3.1 Vorgehen 11](#_Toc317331545)

[8.3.2 Unit Testing 11](#_Toc317331546)

[8.3.3 Code Reviews 11](#_Toc317331547)

[8.3.4 Code Style Guidelines 11](#_Toc317331548)

[8.4 Testen 11](#_Toc317331549)

[8.4.1 <Bezeichnung des Tests (z.B. Integrationstest oder Systemtest)> 11](#_Toc317331550)

1. Einführung
   1. Zweck

Dieses Dokument beschreibt die Planung des SE2 Projekts.

* 1. Gültigkeitsbereich

Dieses Dokument ist während des ganzen Projekts gültig und wird laufend aktualisiert.

* 1. Referenzen

<Liste aller verwendeten und referenzierten Dokumente, Bücher, Links, usw.>

<Referenz auf ein Glossar Dokument, wo alle Abkürzungen und unklaren Begriffe erklärt werden>

<Die Quellen / Referenzen sollten mit dem Word Tool automatisch erstellt werden>

Spielprinzip/Beschreibung: <http://de.wikipedia.org/wiki/Bomberman>

Konzepte für Games: Spiele entwickeln für iPad, iPhone und iPod touch, Thomas Lucka, 2012, ISBN 978-3-446-43085-3

1. Projekt Übersicht

JBomberman ist ein Geschicklichkeitsspiel, das die gleichen Regeln wie das klassische Bomberman umfasst. Es gibt nur einen Multiplayer Modus, in dem bis zu vier Spieler gegeneinander antreten können. Der Multiplayer Modus ist über ein Netzwerk spielbar.

* 1. Zweck und Ziel

<Sinn und Zweck des Projektes, sowie Zielsetzung und auch persönliche Ziele>

Wir wollen gelerntes aus Prog1, Prog2, Uint1 und SE1 anwenden und erweiterte Programmierkonzepte erlernen. Hinzu kommen Module wie VSS und ParProg, die wir parallel besuchen und auch für die Durchführung des Projekts notwendig sind.

Unser Ziel ist es ein Software Projekt in einem Team erfolgreich durchzuführen.

* 1. Primäre Features
* Spiellogik
* Netzwerkspiel
* Dedizierter Server
* Desktop Client
  1. Erweiterte Features
* Verschiedene Powerups
  1. Lieferumfang
* Source-Code
* Dokumentation (Projekt und Software)
* Ausführbare Applikation
  1. Annahmen und Einschränkungen

Die Applikation wird lediglich für Desktops zur Verfügung gestellt, dabei sollen folgende Betriebssysteme (Mac OSX , Linux, Windows) unterstützt werden um eine möglichst grosse Userbasis anzusprechen.

Das Projekt soll nach Abschluss Open Source für Interessenten zur Verfügung gestellt werden.

1. Projektorganisation

<Kurze Beschreibung der Projektstruktur>

Jedes Teammitglied konzentriert sich auf seine eigenen Themengebiete, jedoch werden die Entscheidungen im Team besprochen und das Knowhow auf alle verteilt. Entscheidungen die das gesamte Projekt beeinflussen werden im Team besprochen und müssen von jedem akzeptiert werden.

* 1. Organisationsstruktur

<Projektmitglieder nennen und deren Aufgaben und Verantwortungen aufzählen>

**Daniel Keller**

* Projektbetreuer

**Silvan Adrian**

* Serveradministrator
* Build-Management
* Entwicklung

**Pascal Kistler**

* Qualitätsmanagement
* Entwicklung

**Fabian Binna**

* Projektleiter
* Entwicklung
  1. Externe Schnittstellen

<Ansprechpartner, verantwortliche Personen, Betreuer, usw. aufzählen>

Projektbetreuer: Daniel Keller

Es werden regelmässige Reviews mit Daniel Keller durchgeführt und ein Protokoll jener geführt.

1. Management Abläufe
   1. Kostenvoranschlag

<Wie viel Zeit steht zur Verfügung? In welcher Zeitspanne läuft das Projekt? Wird das Projekt früher beendet, dafür wöchentlich mehr gearbeitet?>

* 1. Zeitliche Planung

<Kurze Beschreibung der zeitlichen Planung und mit einer Grafik einen Überblick über die Phasen, Iterationen und Meilensteine geben. Das Datum des Eintreffens der Meilensteine sollte in der Phasenübersicht ersichtlich sein.>

* + 1. Phasen / Iterationen

|  |  |  |
| --- | --- | --- |
| Iteration | Beschreibung | Ende |
| Inception 1 | Projektantrag einreichen | SW01 |
| Elaboration 1 | Projektplan, Domain Model, Requirements (Brief Use Cases), Configuration Management (Redmine, Jenkins, Git, Gerrit lokale Entwicklungsumgebung) | SW03 |
| Elaboration 2 | Externes Design und Mockups, Sprites, Fully dressed Use Cases | SW05 |
| Elaboration 3 | Architektur Prototyp | SW06 |
| Construction 1 | Alphaversion nur lokal spielbar (alleine)  Bugs aus Test Feedback lösen | SW08 |
| Construction 2 | Hauptfeatures (Server, Netzwerkfähigkeit, Sync mit Server). Bugs aus Test-Feedback fixen. | SW10 |
| Construction 3 | Hauptfeatures fertigstellen, erweiterte Features gemäss Priorisierung und Zeitschätzung in Redmine. Bugs aus Test-Feedback fixen. | SW12 |
| Transition | Vorbereitung zur Schlusspräsentation und Abgabe. | SW14 |

* + - 1. <Bezeichnung der einzelnen Phasen>

<Kurze Beschreibung und Dauer der Phase angeben>

* + - * 1. <Bezeichnung der einzelnen Iterationen>

<Kurze Beschreibung und Dauer der Iteration angeben>

<Vorgehen bei Iterationsplanung und Iterationsassessment>

* + 1. Meilensteine
       1. <Bezeichnung der einzelnen Meilensteine>

<Setzen Sie in Ihrem Projekt 6-8 Meilensteine. Kurze Beschreibung der Meilensteine mit genauem Datum. In der Regel auf Ende jeder Iteration einen Meilenstein setzen (diese Faustregel gilt nur für die SE-2Projekte, in realen Projekten haben Sie oft deutlich mehr Iterationen als Meilensteine, weil Meilensteine dort die nach aussen kommunizierten Ereignisse sind). Schreiben Sie zu jedem Meilenstein auf, welche Arbeitsprodukte (work products) Sie dann abliefern werden . Spezifizieren Sie wenn nötig auch den Fertigstellungsgrad der Arbeitsprodukte, z.B. „Zentrale Use Cases ‚fully dressed‘, restliche UCs im ‚brief‘ Format“, oder „Architekturskizze inkl. Definition der Interfaces

zwischen Sub-Systemen und Deployment Diagramm“>

* + - 1. MS1 Projektplan

Termin: 5.03.2015

Beschreibung: Projektplan fertig ausgearbeitet.

Work Products: Projektplan

* + - 1. MS2 Anforderungen und Analyse

4.2.2.4 MS3 Ende Elaboration/Architekturprototyp

4.2.2.5 MS4 Architektur/Design

4.2.2.6 MS5 Ende Construction

4.2.2.7 MS6 Schlusspräsentation und Abgabe

Termin:28.05.15

Beschreibung: Schlussabgabe mit Demo und Präsentation

Work Products: Schlusspräsentation, fertige Dokumentation, fertiges Spiel (Client/Server Applikation)

* 1. Besprechungen

<regelmässige Besprechungen nennen (wann, wo, wer, Ziel und/oder Grund)>

Es werden regelmässige Besprechungen am Donnerstag morgen eingeplant um den Stand der jeweiligen arbeiten zu erfahren und gegebenenfalls Hilfestellung anzubieten ,falls etwas nicht funktionieren sollte.

1. Risikomanagement
   1. Risiken

<Verweis auf Dokument TechnischeRisiken.xlsx>

Risiken werden im Dokument TechnischeRisiken-JBomberman.xslx beschrieben

* 1. Umgang mit Risiken

<Begründungen zur Tabelle. Weitere Beschreibungen zu Massnahmen und Vorbeugungen. Werden Reserven /Rückstellungen eingeplant? Wieso und wie viele? Wann werden Risiken qualitätssichernd überprüft (Vorgehen und Zeitpunkt(e) zur Neubeurteilung der Risiken)?>

Für die Risiken werden Reserven eingeplant. Die Reserven werden direkt in die einzelnen Tickets eingerechnet. Falls Risiken eintreffen werden diese sofort kommuniziert. Jede Woche werden eingetroffene Risiken und potenzielle Risikogefahren diskutiert. Eingetroffene Risiken werden im Team besprochen und möglichen Lösungen evaluiert.

1. Arbeitspakete

<Definieren Sie in einem separaten Tool (Redmine oder XLS, o.a.) diejenigen Arbeitspakete, die Ihnen zu Beginn des Projektes schon mal einfallen. Zu Beginn können Pakete eher generisch ausfallen (z.B. ‚Domainmodell erstellen‘ oder ‚GUI Programmieren‘, oder ‚ Usability Testing‘).

Denken Sie daran, dass es in jedem Projekt auch eine ganze Reihe von ‚overhead‘-Tätigkeiten gibt: Projektleitung und -Sitzungen, Aufsetzen der Server und Werkzeuge, Qualitätsmassnahmen, Schlusspräsentation erstellen und anderes mehr – alles Tätigkeiten, die nicht direkt zum Software-Produkt führen. Planen Sie genug Zeit auch für diese Tätigkeiten ein.

Dann sollte es in jeder Projektplanung einige projektspezifische Arbeitspakete geben, wie sie so nicht in anderen Projekten vorkommen (z.B. ‚ Level-Editor entwerfen‘, oder ‚Verifikation der Zahlungsangaben programmieren‘). Wenn es keine solchen Arbeitspakete gibt, ist der Projektplan zu generisch. Das heisst auch, dass man sich zu wenig Gedanken über die anfallenden Arbeiten gemacht hat.

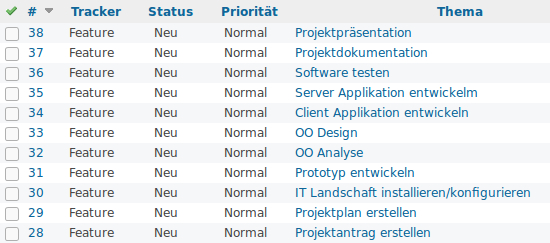
Später im Projekt werden Sie die Arbeitspakete noch verfeinern, verschieben, jemandem zuordnen, neu schätzen, etc. Deswegen empfiehlt sich der Einsatz eines Werkzeuges wie Redmine.

Dokumentieren Sie URL und Logins auf das Projektmanagement Tool, in welchem die Meilensteine und die dazugehörigen Arbeitspakete erfasst sind.>

Die Arbeitspackete werden in Redmine erstellt und gepflegt (siehe Screenshot).

Lesender Zugriff ist anonym möglich, schreibender nur eingeloggt (Projekt ist öffentlich).

Link zum redmine: <http://se2p.zonk.io/redmine>



1. Infrastruktur

<Benötigte Infrastruktur aufzählen. Spezielle Geräte, Laptop , Tools usw. und nötigenfalls aufzeigen für welche Bereiche diese verwendet werden. Eventuell auch Verfahren beschreiben (auf Tools bezogen).>

Entwicklungsinfrastruktur

Aus persönlicher Präferenz ist die Infrastruktur breit gesät. Beim IDE konnten wir uns auf einen gemeinsamen Nenner bringen.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Hardware | Betriebssystem | IDE |
| Pascal Kistler | ASUS | Windows 8.1 | Eclipse Luna |
| Silvan Adrian | Mac Book Pro | OSX | Eclipse Luna |
| Fabian Binna | Lenovo T530s | Ubuntu 14.04.1 | Eclipse Luna |

Neben den persönlichen Notebooks steht uns der beantragte SE2BuildServer als Entwicklungsinfrastruktur zur Verfügung

Tools/Software

* BuildServer: Jenkins
* CodeReview: Gerrit
* Konfigurtaionsmanagment: Git
* Projektmanagment: Redmine

Kommunikationsmittel

Ausserhalb des Reviewtermins wird mit folgenden Mitteln kommuniziert:

* E-Mail
* Skype
* Redmine (Wiki Kommentare)
* Gerrit (Code Kommentare)
* Whatsapp

1. Qualitätsmassnahmen

<Was wird unternommen damit das Produkt des Projektes, sowie dessen gesamter Verlauf eine hohe Qualität erreicht? Übersicht in einer Tabelle geben mit Massnahmen, Zeitraum und Ziel der Massnahme>

* 1. Dokumentation

Sämtliche Dokumente werden in Git gespeichert. So sind diese immer für alle verfügbar und können einfach versioniert und Änderungen nachvollzogen werden.

* 1. Projektmanagement

Als Projektmanagementtool wird Redmine verwendet. Hier werden alle ausstehenden, laufenden und erledigen Task verwaltet.

<Welches Tool wird für Projektmanagement eingesetzt (z.B. Redmine oder Trac) und wie erfolgt dieser Einsatz? Dazugehörige Links und Logins (Gastbenutzer).>

* 1. Entwicklung

Der Source Code befindet sich ebenfallt in einem Git Repository und wird durch regelmässige Code Reviews überprüft.

<Wo befindet sich der Source Code (z.B. SVN oder Git) und wie wird dessen Qualität sichergestellt?>

* + 1. Vorgehen

<Vorgehen in der Entwicklung>

* + 1. Unit Testing

Es sollten für jede Klasse / jedes Codemodul Junit tests geschriben und durchgeführt werden. Vor jedem push müssen alle Test erfolgreich durchlaufen werden.

<Wo werden welche Unit Tests geschrieben um die Qualität sicherzustellen? Wie wird die Testabdeckung sichergestellt (z.B. durch EclEmma)?>

* + 1. Code Reviews

Durch Code Reviews wird sicher gestellt, dass der Code den Style Guidelines entspricht und wartbar bleibt. Durch die Kontrolle durch ein oder zwei weitere Teammitglieder wird sichergestellt, dass die Codequalität stets auf dem hohem Niveau ist.

<Werden Code Reviews gemacht und wie werden diese gemacht?>

* + 1. Code Style Guidelines

Es werden die Code Style Guidelines aus den Modulen Prog1 ung SE1 verwendet. Namen im Code (wie Variablen, Methoden, etc.) werden in Englisch benannt.

<Welche Code Style Guidelines werden angewendet? Sie brauchen keine eigenen Guidelines zu erfinden. Am besten referenzieren Sie existierende Guidelines, mit denen Sie einverstanden sind. Evtl. noch Abweichungen dazu dokumentieren>

* 1. Testen
     1. Komponententest

Zu jeder Komponente werden mehrere Test geschrieben. Diese stellen sicher, dass alle Komponenten ordnungsgemäss funktionieren und kein nicht funktionierender Code gepusht wird.

* + 1. Integrationstest

Für jede implemetierte Funktion werden wiederum Tests geschrieben, um die korrekte Funktionsweise und die Zusammenarbeit der einzelnen Module sicherzustellen.

* + 1. Systemtest

Systemtest werden gegen Ende des Projekt erstellt. Sie garantieren, dass das gesamte System ordnungsgemäss funktioniert und alle Komponenten richtig zusammen arbeiten.

<Beschreibung der Durchführung, Umsetzung und Umfang der Tests. Wenn möglich mit Mengenangaben (wieviele Tests)>