|  |  |  |
| --- | --- | --- |
| Machbarkeit | October 28  2014 | |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] | | Jevtic, Kodras, El-FAR & Stokic |

Machbarkeit

[1.1 Projekt-Team](#_Toc402288838)

[1.2 Projektbeschreibung 3](#_Toc402288839)

[2.1 Ist-Erhebung 3](#_Toc402288840)

[2.1.1 Allgemeine Ist-Erhebung 3](#_Toc402288841)

[2.1.2 Ist-Zustand 4](#_Toc402288842)

[2.2 Soll-Zustand 4](#_Toc402288843)

[2.2.1 Allgemeine Beschreibung 4](#_Toc402288844)

[2.2.2 Muss-Ziele 5](#_Toc402288845)

[2.2.3 Kann-Ziele 6](#_Toc402288846)

[2.2.4 Nicht-Ziele 6](#_Toc402288847)

[2.3 Festlegen der Hauptdaten 6](#_Toc402288848)

[2.4 Festlegen der Hauptleistungen 7](#_Toc402288849)

[2.5 Festlegen der wichtigsten Qualitätsmerkmale 7](#_Toc402288850)

[3.1 Prüfen alternativer Lösungsvorschläge 8](#_Toc402288851)

[3.1.1 Programmiersprachen/Skriptsprachen 8](#_Toc402288852)

[4.1 Nutzen für den Kunden 11](#_Toc402288853)

[4.2 Nutzen für das Projektteam 11](#_Toc402288854)

1. Projektdaten

# 1.1 Projekt-Team

**Name:** Boban Jevtic

**E-Mail:** [bjevtic@student.tgm.ac.at](mailto:bjevtic@student.tgm.ac.at)

**Fähigkeiten:** Java, Photoshop, CSS, PHP, MySQL, HTML, JavaScript, C

**Bisherige Projekte:** Subnet-Rechner

**Name:** Dominik Kodras

**E-Mail:** [dkodras@student.tgm.ac.at](mailto:dkodras@student.tgm.ac.at)

**Fähigkeiten:** Java, Photoshop, CSS, PHP, MySQL, HTML, JavaScript, C

**Bisherige Projekte:** JPS-Wecker, The Lightbringer

**Name:** Stefan Stokic(Programmierer)

**E-Mail:** [sstokic@student.tgm.ac.at](mailto:sstokic@student.tgm.ac.at)

**Fähigkeiten:** Java, Photoshop, CSS, PHP, MySQL, HTML, JavaScript, C

**Bisherige Projekte:** Space Warfare, FlashNotify+

**Name:** Matthias El-FAR

**E-Mail:** [melfar@student.tgm.ac.at](mailto:melfar@student.tgm.ac.at)

**Fähigkeiten:** Java, Photoshop, CSS, PHP, MySQL, HTML, JavaScript, C

**Bisherige Projekte:** Vokabeltrainer, NotiZync

# 1.2 Projektbeschreibung

3D-Schach ist ein dreidimensionales Brettspiel, das auf mehreren Ebenen basiert. Es besteht aus 32 Spielfiguren (16 weiße Figuren und 16 schwarze Figuren). Es gibt 6 Arten von Spielfiguren (Bauer, Turm, Pferd, Läufer, Dame und König). Jede Spielfigur hat bestimmte Regel, wie man sie bewegen darf. Die Art des 3D-Schachs heißt Star-Track-Schach. Das ist eine besondere Variante des klassischen Schachspiels, die immer wieder in populären Medien auftaucht. Die Regeln beim 3D-Schach sind ähnlich, wie beim 2D-Schach, werden aber ergänzt, da es mehrere Ebenen gibt.

Außerdem bieten wir den Spieler die Möglichkeit gegen einen anderen Spieler zu spielen, aber auch gegen eine künstliche Intelligenz zu spielen.

Die Software wird später komplett kostenlos online zur Benutzung verfügbar sein.

2. Voruntersuchung des Produkts

# 2.1 Ist-Erhebung

## 2.1.1 Allgemeine Ist-Erhebung

Als Anwendung wird dieses Spiel ein ausführbares JAR-File sein, dass heißt der Benutzer kann dieses Spiel lokal auf dem PC ausführen, jedoch unter der Voraussetzung, dass er eine aktuelle Version von JRE (Java-Laufzeitumgebung) auf seinem PC installiert hat.

## 2.1.2 Ist-Zustand

* Nachdem sich der User registriert hat und er die zugeschickte E-Mail auf seinem E-Mail-Account bestätigt hat, kann er sich seinem Benutzernamen und Passwort einloggen und online gegen andere Spieler antreten.
* Es gibt schon viele Schachspiele die am Computer visualisiert wurden, jedoch gibt es nur wenige andere Arten von Schachspielen, die vom herkömmlichen Schach abweichen.
* Die Idee für ein dreidimensionales Schach ist relativ alt. Es gibt schon sehr viele Schachspiele, die auf dem Markt verkauft werden. Jedoch sind diese vermarkteten 3D-Schach-Spiele ziemlich benachteiligt, weil sie eine Menge Geld kosten.
* Im Internet stehen auch Schach-Spiele in der 3D Umgebung als Download zur Verfügung existieren.
* Im Netz findet man nicht nur Schachspiele, wo man nur gegen eine künstliche Intelligenz spielen kann, sondern man kann sich auch gegen menschliche Gegner messen.
* Als Beispiel nennen wir einen „Star-Trek-Schach“ mit folgenden Eigenschaften
* sehr bekannte 3D-Variante, keine offiziellen Regeln.
* Bezeichnung: Herkunft aus der Fernsehserie „Star-Trek“
* 64 Felder, gleiche Figuren, wie beim 2D-Schach
* Übereinanderliegende Brettern mit vier beweglichen „Angriffsdecks“
* Jede Figur darf über Ebenen ziehen

# 2.2 Soll-Zustand

## 2.2.1 Allgemeine Beschreibung

* Das Ziel des Projektes das Laufen des Spiels unter jedem Betriebssystem. Es soll auch die Möglichkeit haben mit anderen Spielern anzutreten und sich mit Ihnen zu verbinden.
* Jedoch ist die unterscheidet sich schon unser Programm von anderen Spielen, indem dieses 3D-Schach ein OpenSource sein wird.

## 2.2.2 Muss-Ziele

* Benutzerfreundlichkeit

Um die Anwendung benutzerfreundlicher zu machen muss das Spiel plattformunabhängig sein und auf jedem Betriebssystem funktionieren. Um den Gegenspieler erreichen zu können, benötigen wir also einen Web Server, um die Verbindung aufzubauen.

* Ermöglichen eine einfache Registrierung und die Verbindung mit anderen Spieler

Für die Benutzung von dem Spiel wird ein Account benötigt. Dieser Account wird über eine Registrierung erstellt, die am Anfang des Programms angezeigt wird und besitzt Daten über den User wie Username und Passwort.

* Interface für die Kommunikation mit anderen Spieler

Das Programm muss die Möglichkeit besitzen auf irgendeiner Art und Weise mit anderen Menschen (Usern) zu kommunizieren. Dabei wird eine Chat-Funktion in diesem Spiel integriert sein.

* Erweiterung um eine Koordinate (wie zum Beispiel z-Achse)

Ein 3D-Schach-Spiel ist ein dreidimensionales Spiel, das auf mehreren bzw. (mindestens 2 Ebene) basiert. Man kann sich das so vorstellen, wie wenn das Spiel um eine Koordinate erweitert wird.

* Darstellung

Das dreidimensionale Schachspiel, Registrierung und Login muss grafisch dargestellt werden. Nach dem Login sollen alle offenen Spielräume dargestellt werden.

* Validierung der Spielzüge

Die Applikation soll den Spielzug überprüfen, ob der Zug gültig ist oder nicht. Wenn der Spielzug ungültig ist, wird er nicht ausgeführt.

## 2.2.3 Kann-Ziele

* Darstellung des Schachspiels mit mehreren Farben

Dem Benutzer wird die Auswahl von Farben zur Verfügung gestellt, um das Spiel zu personalisieren. Dabei kann er die Farben der Figuren und des Schachbretts verändern.

* Registrierung mit Hilfe von Mail-Verifikation

Bei der Registrierung muss der User zusätzliche eine E-Mail-Adresse angeben. An dieser E-Mail-Adresse wird eine E-Mail verschickt, welche zur zusätzlichen Verifikation dient.

## 2.2.4 Nicht-Ziele

* Grundkosten

Das Spiel wird keine Grundkosten besitzen die man bezahlen muss um das Spiel spielen zu dürfen. Es ist für jeden frei erreichbar und spielbar.

* Keine Ablenkung vom Hauptkonzept

Es wird keine kleineren minispiele im Projekt geben die von dem Grundkonzept ablenken könnten.

# 2.3 Festlegen der Hauptdaten

* ***/LD10/Benutzerdaten:***

Zu jedem Benutzer müssen Login-Daten und Spielergebnisse gespeichert werden. Dazu wird eine Datenbank benutzt.

# 2.4 Festlegen der Hauptleistungen

* ***/LL10/Serverleistung:***

Der Server muss mehrere hundert Personen gleichzeitig verkraften können. Die Mindestanzahl die verkraftbar sein muss ist 100.

* ***/LL20/Plattformunabhängigkeit:***

Das Spiel muss von allen gängigen Betriebssystemen spielbar sein.

# 2.5 Festlegen der wichtigsten Qualitätsmerkmale

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Produktqualität** | **Sehr Gut** | **Gut** | **Normal** | **Irrelevant** |
| Funktionalität | X |  |  |  |
| Zuverlässigkeit |  |  | x |  |
| Benutzbarkeit |  |  | x |  |
| Effizienz |  |  | x |  |
| Änderbarkeit |  |  |  | x |
| Übertragbarkeit | X |  |  |  |

* **Funktionalität**

Die Applikation soll eine sehr gute Funktionalität besitzen, damit es den Benutzer anspricht, sodass der Benutzer das Spiel gerne spielt.

* **Benutzbarkeit**

Die Benutzbarkeit ist neben der Funktionalität das wichtigste bei einer Applikation da, ein kompliziertes und schwer zu spielendes Spiel schnell den Benutzer die Lust zum spielen raubt.

3. Durchführbarkeitsuntersuchung

# 3.1 Prüfen alternativer Lösungsvorschläge

## 3.1.1 Programmiersprachen/Skriptsprachen

**JAVA**

Java ist eine objektorientierte Programmiersprache.

Als IDE verwenden wir Eclipse.

Folgende Tools werden benötigt:

* Entwicklungsumgebung JDK (Java Development Kit)
* Laufzeitumgebung JRE (Java Runtime Environment)

JDK wird für das Coden von Prgrammen verwendet. JRE wird für das Ausführen von Programmen verwendet.

**Eigenschaften von Java:**

* objektorientiert
* dynamisch
* übertragbar
* robust und sicher
* einfach

**Robustheit:**

Viele der Designentscheidungen bei der Definition von Java reduzieren die Wahrscheinlichkeit ungewollter Systemfehler; zu nennen sind die starke Typisierung, Garbage Collection, Ausnahmebehandlung sowie Verzicht auf Zeigerarithmetik.

**Einfachheit:**

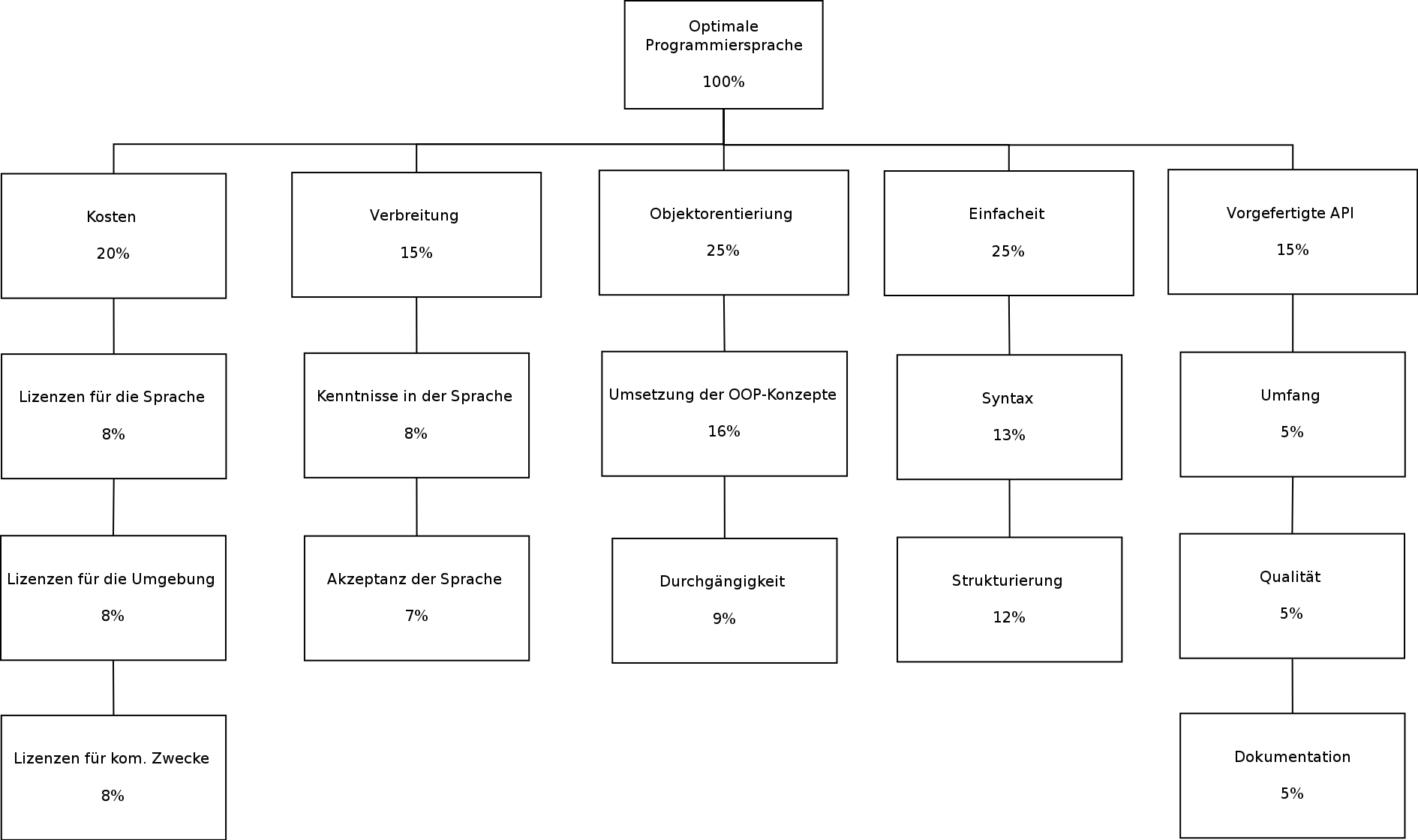
Java ist im Vergleich zu anderen objektorientierten Programmiersprachen wie C++ oder C# einfach, da es einen reduzierten Sprachumfang besitzt und beispielsweise Operatorüberladung und Mehrfachvererbung nicht unterstützt.

**Sicherheit:**

Dafür stehen Konzepte wie der Class-Loader, der die sichere Zuführung von Klasseninformationen zur Java Virtual Machine steuert, und Security-Manager, die sicherstellen, dass nur Zugriff auf Programmobjekte erlaubt wird, für die entsprechende Rechte vorhanden sind.

***Nutzwertanalyse***

Zielhierarchie:

****

Erläuterung der Kriterien:

**• Kosten**

- Lizenzen für die Sprache:

Muss für den Einsatz der Sprache eine Gebühr bezahlt werden?

- Lizenzen für die Entwicklungsumgebung:

Muss für den Erwerb bzw. Einsatz einer Entwicklungsumgebung für die jeweilige Sprache eine Gebühr bezahlt werden?

- Lizenzen für kommerzielle Zwecke:

Muss für den kommerziellen Einsatz eines in einer bestimmte Programmiersprache entwickelten Softwareprodukts eine Gebühr bezahlt werden?

**• Verbreitung**

- Kenntnisse in der Sprache

Beschreibt inwiefern unsere Entwickler bereits mit der jeweiligen Programmiersprache vertraut sind.

- Akzeptanz der Sprache

Beschreibt inwiefern unsere Entwickler die jeweilige Sprache gutheißen.

**• Objektorientierung**

- Umsetzung der OOP-Konzepte

Wurden alle Konzepte der OOP (Abstraktion, Datenkapselung, Vererbung, Polymorphie, etc.)

vollständig umgesetzt?

- Durchgängigkeit

Ist es möglich bzw. wie weit ist es möglich vom Konzept der OOP abzuweichen?

**• Einfachheit**

- Syntax

Ist die Syntax einfach, intuitiv und logisch aufgebaut?

- Strukturierung

Bietet die Programmiersprache die Möglichkeit einer einfachen und übersichtlichen Strukturierung?

**• Vorgefertigte API**

- Umfang

Wie viele vorgefertigte Klassen bzw. Programmteile stehen zur Verfügung?

- Qualität

Ist die API von einer für uns brauchbaren Qualität (d.h. fehlerfrei, effizient, etc.)?

- Dokumentation

Sind alle Inhalte der API vollständig dokumentiert?

4.Nutzenanalyse

# 4.1 Nutzen für den Kunden

Dem Kunden wird ein schnelles und einfaches Schachspiel angeboten durch die Verwendung von JAR-File, wenn er das Programm nur per Doppelklick starten kann.

4.2 Nutzen für das Projektteam

Das Projekt wird auf eindrucksvoller Weise auf die Fähigkeiten des Teams aufmerksam machen. Außerdem wird es dem Projektteam den späteren Einstieg in andere Firmen erleichtern, da das Projektteam bei der Bewerbung das Projekt präsentieren kann.