SEP – Hauptaufgabe

Projektmappe des Projektes
SEP RP

Spezifikation des Projektes

Gruppe Gruppe K:

Katharina Böse
Johannes Grundmann
Sami Khatif
Gizem Gülser
Thorben Friedrichs
Tristan Corzilius
Mark Leifeld

Systemdesign des Projektes

Gruppe Gruppe L:

Angelo Soltner
Bardia Asemi-Soloot
Bijan Shahbaz Nejad
Dilara Güler
Dominikus Häckel
Sabiha Can
Tobias van den Boom

Einleitung

Dieses Dokument enthält alle nötigen Informationen zur Erstellung eines Software-Produktes. Die Erstellung eines Software-Produktes wird im Allgemeinen auch als Programmierung bezeichnet. Programmierung kann man aber auch dahingehend verstehen, dass ein Computers zur Durchführung eines bestimmten Verhaltens konfiguriert werden muss.[1] Vorher müssen jedoch alle nötigen Informationen über das „bestimmte Verhalten" zusammengetragen und dokumentiert werden. Diese Informationen bestehen aus Anforderungen (zu neudeutsch *Requirements*), Architekturbeschreibungen, etc., die im Folgenden in diesem Dokument wiedergegeben werden.
Dieses Dokument soll euch durch den gesamten Verlauf der Hauptaufgabe begleiten und dazu dienen, euer Projekt zu dokumentieren. Jeder Abschnitt beschäftigt sich mit einem Teilaspekt eurer Aufgabenstellung (Anforderungen, Architektur, Projektplanung, Testfälle etc.). An vielen Stellen findet ihr im Dokument folgendes Kästchen:

Dies ist eine Hilfestellung.

Diese Kästen dienen dazu, euch kurze Informationen über Ziele und Inhalte der jeweiligen Abschnitte zu geben. Sie sind spätestens zur finalen Abgabe der Projektmappe vollständig zu entfernen. Betrachtet dieses Dokument bitte nicht als Aufgabe, die man von oben nach unten abarbeiten soll; es soll vielmehr als durchgängige Dokumentation eurer Projektarbeit dienen und fortlaufend erweitert bzw. angepasst werden, sodass am Ende des SEPs eine Software entstanden ist, die sich in diesem Dokument wieder findet.

Eine Anmerkung in eigener Sache

Dieses Dokument soll keinen starren Rahmen vorgeben, sondern vielmehr eine Richtschnur für eure Arbeit sein. Wenn Ihr Abschnitte anders gestalten wollt, so könnt ihr dies gerne tun (grobe Änderungen bitte mit eurem Betreuer absprechen, außerdem nur strukturelle Änderungen auf den Ebenen unter der ersten Strukturierungsebene (1, 2, 3, …) durchführen). Ferner ist dieses Dokument keineswegs vollständig oder erhebt Anspruch auf Perfektion. Wenn ihr Anmerkungen und/oder Verbesserungsvorschläge habt, dann könnt ihr diese gerne an euren Betreuer weitergeben. Wir werden sie dann in das Vorlagedokument übernehmen.

Das SEP-Team wünscht euch

viel Erfolg

bei der Bearbeitung der Hauptaufgabe!!!

[1] vgl. B. Kovitz: Practical Software Requirements: A Manual of Content & Style; Manning 1998

Inhaltsverzeichnis

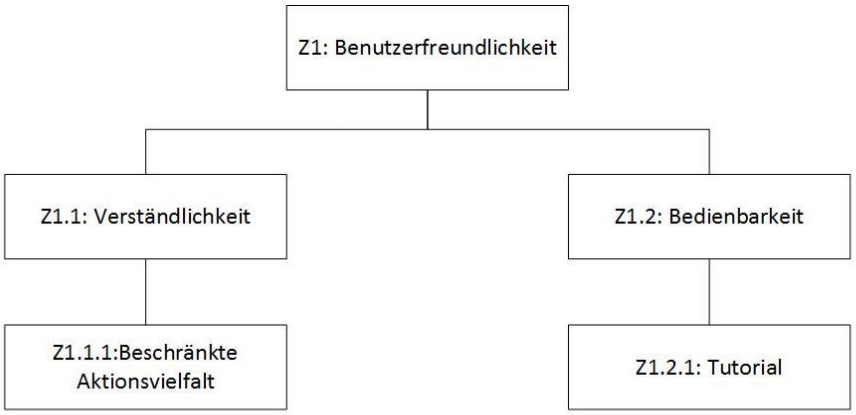
1. Projektbeschreibung SEP RP

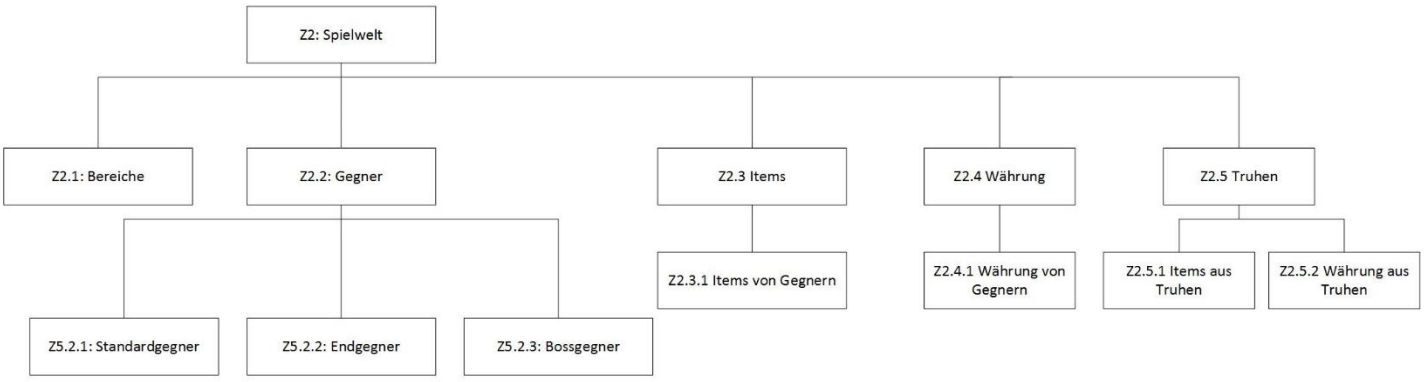
Die Aufgabe besteht in der Entwicklung eines Action Rollenspiels. Der Spieler soll dabei in die Rolle einer fiktiven Person (im Folgenden: Charakter) schlüpfen und Gegner bekämpfen. Der Spieler steuert den Charakter mit Maus und Tastatur aus der Vogelperspektive. Zu Beginn des Spiels soll der Spieler die Möglichkeit haben den Charakter nach seinen Wünschen anzupassen. Dabei hat er die Möglichkeit sich einen Charakter aus mindestens vier Charaktertypen auszuwählen, die jeweils mindestens vier einzigartige Fähigkeiten erlernen können. Diese Fähigkeiten soll der Spieler bewusst einsetzen können, um das Spielgeschehen zu beeinflussen. Im Verlauf des Spiels kann der Spieler seinen Charakter verbessern, um sich in immer schwierigeren Kämpfen behaupten zu können. Gegner gibt es in mindestens drei verschiedenen Typen, die der Spieler bekämpfen kann, und die auf unterschiedliche Arten auf den Spieler reagieren. Die Geschichte des Spiels soll den Charakter des Spielers aktiv einbinden. Das heißt, dass die Entscheidungen, die der Spieler im Verlauf des Spiels trifft, sich auf den Verlauf der Geschichte auswirken. Insgesamt kann der Spieler die Geschichte des Spiels so zu mindestens drei verschiedenen Enden führen.
Der Charakter soll darüber hinaus in der Lage sein während des Spielens Items zu erlangen, die sein Charakter benutzen kann, um Vorteile zu erhalten. Es gibt mindestens fünf verschiedene Kategorien von Items, die sich in ihrer Nutzung unterscheiden, wobei es von jeder Kategorie mindestens vier verschiedene Items geben soll. Dabei können selten auftretende Items dem Charakter einen größeren Vorteil bringen als häufig auftretende Items. Außerdem beinhaltet das Spiel ein Währungssystem, um es dem Spieler zu ermöglichen, Items in einem Auktionshaus anderen Spielern zum Verkauf anzubieten. Neben dem Verkauf im Auktionshaus soll der Spieler auch in der Lage sein durch das Bekämpfen von Gegnern Geld des Währungssystems zu erhalten. Das Auktionshaus soll über eine Netzwerkverbindung erreicht werden können. Das Spiel soll für Bildschirme mit einer Auflösung von 1920x1080 Pixel entwickelt werden und der Charakter darf auf diesen Bildschirmen maximal 150 Pixel hoch und 150 Pixel breit sein. Das Spiel beinhaltet mindestens drei separierte Bereiche, die der Spieler erkunden kann. Der erste dieser Bereiche soll mindestens 15.000.000 Quadratpixel groß sein, der zweite Bereich mindestens 25.000.000 Quadratpixel groß sein und der dritte Bereich mindestens 35.000.000 Quadratpixel groß sein. In jedem dieser drei Bereiche hält sich ein jeweils einzigartiger Gegner auf, der deutlich schwerer zu besiegen ist als die normalen Gegner. Außerdem soll der Charakter, am Ende des Spiels, einen weiteren einzigartigen Gegner besiegen müssen, der schwieriger zu besiegen ist als alle anderen Gegner. Dem Spieler soll mindestens nach dem Durchqueren jeder der drei Bereiche angeboten werden seinen Spielstand zu speichern. Gespeicherte Spielstände sollen auch nach Beenden und Neustarten des Spiels bestehen bleiben. Der Spieler kann so eine beliebige Anzahl Spielstände speichern und laden.

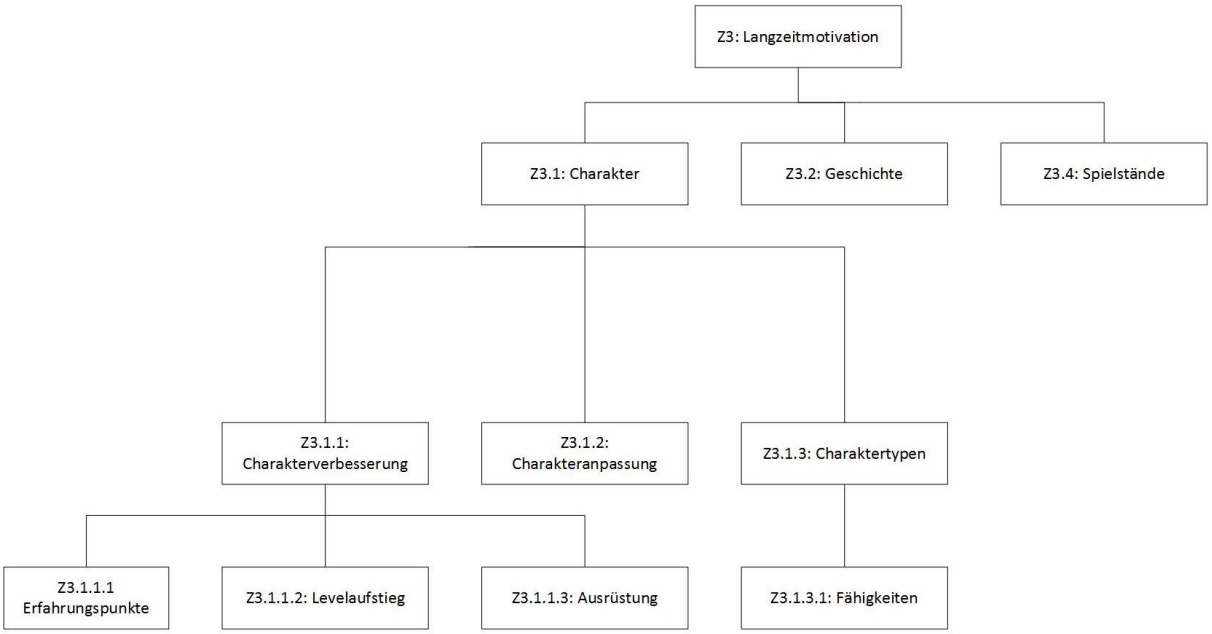
Der Softwareentwicklungsprozess basiert im Rahmen des SEP auf dem angepassten V-Modell. Die Projektmappe ist entsprechend den Phasen des V-Modells aufgebaut. Jede Phase wird Schritt für Schritt im Verlaufe der Veranstaltung bearbeitet und dokumentiert.

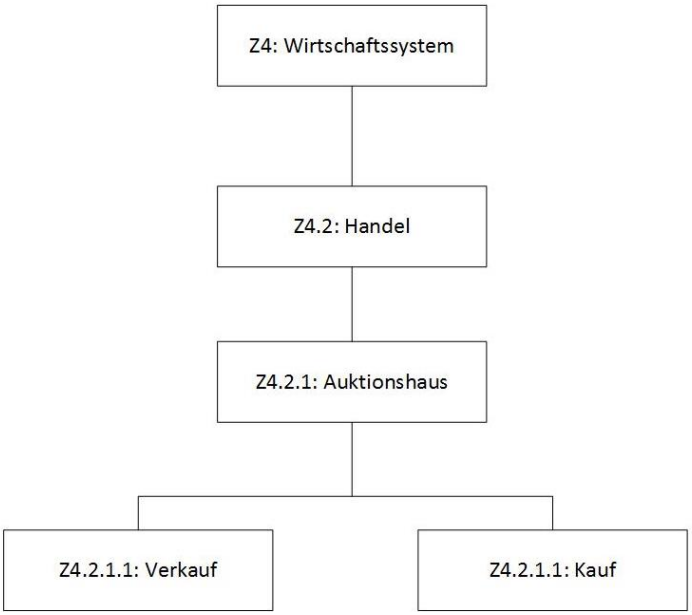
2. Anforderungsdefinition

2.1 Zielmodell






Z3: Langzeitmotivation
Z3.1: Charakter
Z3.2: Geschichte
Z3.4: Spielstände
Z3.1.1: Charakterverbesserung
Z3.1.2: Charakteranpassung
Z3.1.3: Charaktertypen
Z3.1.1.1 Erfahrungspunkte
Z3.1.1.2: Levelaufstieg
Z3.1.1.3: Ausrüstung
Z3.1.3.1: Fähigkeiten


Z4: Wirtschaftssystem
Z4.2: Handel
Z4.2.1: Auktionshaus
Z4.2.1.1: Verkauf
Z4.2.1.1: Kauf

- **Z-1: Benutzerfreundlichkeit**
  Das Spiel soll benutzerfreundlich sein.
  - **Z-<L>-<SG>-1.1: Verständlichkeit**
    Das Spiel soll selbsterklärend sein.
    - **Z-<L>-<HG>1.1.1 Beschränkte Aktionsvielfalt**
      Die Spieler kann 10 verschiedene Aktionen ausführen.
      - **Z-<T>-<HG>1.1.1.1 Eingabegeräte**
        Zur Eingabe werden Tastatur und Maus benutzt.
  - **Z-<L>-<SG>-1.2: Bedienbarkeit**
    Die Bedienung des Spiels soll einfach sein.
    - **Z-<L>-<HG>-1.2.1 Tutorial**
      Der Spieler wird in einem Tutorial mit der Bedienung vertraut gemacht.

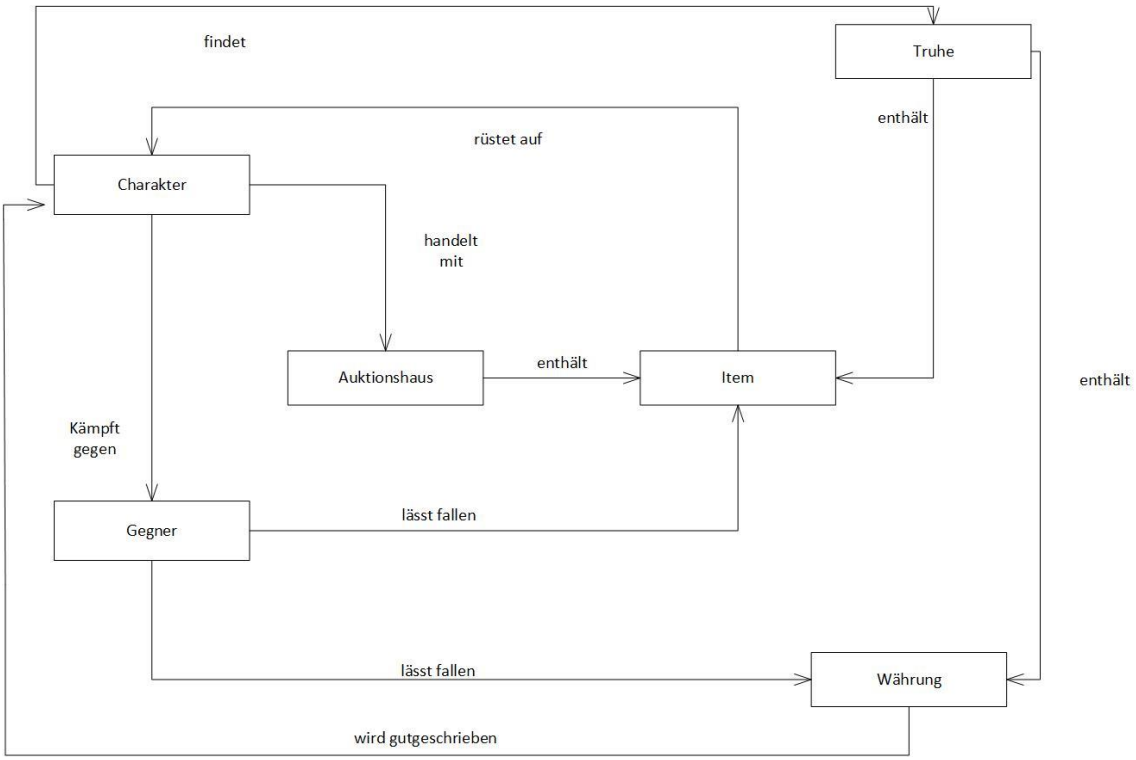
- **Z-2: Spielwelt**
  Das Spiel hat eine Spielwelt, in der der Spieler den Charakter bewegen kann.
  - **Z-<L>-<HG>-2.1: Bereiche**
    Das Spiel hat mindestens drei verschiedene Bereiche, die der Spieler durchqueren muss.
  - **Z-<L>-<HG>-2.2: Gegner**
    Im Bereich gibt es verschiedene, zu bekämpfende Gegner.
    - **Z-<L>-<HG>-2.2.1: Standardgegner**
      Die Bereiche sollen drei verschieden Arten von Gegnern enthalten.
    - **Z-<L>-<HG>-2.2.2: Endgegner**
      Am Ende jedes Bereichs soll ein stärkerer Gegner sein.
    - **Z-<L>-<HG>-2.2.3: Bossgegner**
      Nach Besiegen des dritten Endgegners soll ein besonders starker Gegner erscheinen.
  - **Z-<L>-<HG>-2.3: Items**
    Der Charakter soll Items aufsammeln können.
    - **Z-<L>-<HG>-2.3.1: Items von Gegnern**
      Besiegte Gegner können Items hinterlassen.
  - **Z-<L>-<HG>-2.4: Währung:**
    Das Spiel enthält eine eigene Währung.
    - **Z-<L>-<HG>-2.4.1: Währung von Gegnern**
      Besiegte Gegner können Währung hinterlassen.
  - **Z-<L>-<HG>-2.5: Truhen:**
    Es stehen Truhen in den Bereichen.
    - **Z-<L>-<HG>-2.5.1: Items aus Truhen**
      Der Charakter kann Items aus Truhen erhalten.
    - **Z-<L>-<HG>-2.5.2: Währung aus Truhen**
      Der Charakter kann Währung aus Truhen erhalten.
- **Z-3: Langzeitmotivation**
  Das Spiel soll den Spieler über längere Zeit zum Spielen motivieren.
  - **Z-<L>-<SG>-3.1: Charakter**
    Der Charakter ist die fiktive Spielfigur, die vom Spieler gesteuert wird.

- **Z-<L>-<HG>-3.1.1: Charakterverbesserung**
  Der Spieler soll die Fähigkeiten seines Charakters verbessern können.
    - **Z-<L>-<HG>3.1.1.1: Erfahrungspunkte**
      Der Charakter erhält für das Besiegen von Gegnern Erfahrungspunkte.
    - **Z-<L>-<HG>3.1.1.2: Levelaufstieg**
      Hat der Charakter ausreichend Erfahrungspunkte gesammelt, steigt er ein Level auf.
    - **Z-<L>-<HG>3.1.1.3: Ausrüstung**
      Der Charakter kann durch Items verbessert werden.
- **Z-<L>-<HG>-3.1.2: Charakteranpassung**
  Der Spieler soll das Aussehen seines Charakters anpassen können.
- **Z-<L>-<HG>-3.1.3: Charaktertypen**
  Der Spieler hat die Wahl zwischen verschiedenen, einzigartigen Charaktertypen.
    - **Z-<L>-<HG>-3.1.3.1: Fähigkeiten**
      Jeder Charaktertyp hat einzigartige, erlernbare Fähigkeiten.
- **Z-<L>-<HG>-3.2: Geschichte**
  Das Spiel beinhaltet eine begleitende Geschichte.
- **Z-<L>-<HG>3.3: Spielstände:**
  Der Spieler soll die Möglichkeit haben, seinen Fortschritt in beliebig vielen Spielständen abzuspeichern.

- **Z-4: Wirtschaftssystem**
  Das Spiel beinhaltet ein eigenes Wirtschaftssystem.
    - **Z-<L>-<HG>-4.1: Handel**
      Der Spieler soll Handel mit anderen Spieler treiben können.
        - **Z-<L>-<HG>-4.1.1: Auktionshaus**
          Der Spieler soll die Möglichkeit haben, Items im Auktionshaus an andere Spieler zu verkaufen.
            - **Z-<T>-<HG>-4.1.1.1 Server**
              Das Auktionshaus läuft auf einem Server.
                - **Z-<T>-<HG>-4.1.1.1.1: Netzwerkverbindung**
                  Die Funktionen des Auktionshauses werden über eine Netzwerkverbindung zum Server bereitgestellt.
                    - **Z-<L>-<HG>-4.1.1.1.1.1: Verkauf**
                      Der Spieler soll die Möglichkeit haben, Items im Auktionshaus an andere Spieler zu verkaufen.
                    - **Z-<L>-<HG>-4.1.1.1.1.2: Kauf**
                      Der Spieler soll die Möglichkeit haben, Items im Auktionshaus von anderen Spielern zu erwerben.
                - **Z-<T>-<HG>-4.1.1.1.2 Datenbank**
                  Die Inhalte des Auktionshauses werden in einer Datenbank gespeichert.
- **Z-5: Effizienz in der Entwicklung**
  In der Entwicklung wird nach Möglichkeit auf vorhandene Techniken zurückgegriffen

- **Z-<T>-<HG>-5.1: Speichermedium**
  Die Spielstände werden lokal in einer Datenbank gespeichert

2.2 **Spielmodell**



Charakter:

Der Charakter wird vom Spieler gesteuert mit Hilfe der Eingabefehle.

Gegner:

Der Gegner wird vom Charakter bekämpft.

Währung:

Währung kann von besiegten Gegnern erhalten, in Truhen gefunden und vom Charakter genutzt werden.

Item:

Items sind Ausrüstungsgegenstände für den Charakter, welche im Autkionshaus gehandelt werden, von besiegten Gegnern erhalten werden und in Truhen gefunden werden können.

Auktionshaus:

Der Spieler handelt im Auktionshaus mit Items des Charakters.

Truhe:

In den Truhen befinden sich Items oder Währung.

2.3 **Szenarien**

Szenario 1 ***„Erfolgreicher Kampf“*** (Erfüllt Ziele 2.3.1, 2.4.1 und 3.1.1.1)

1. Der Nutzer steuert den Charakter auf einen Gegner zu.
2. Das System weist den Gegner an, den Charakter zu attackieren.
3. Der Nutzer weist den Charakter an, Angriff auszuführen.
4. Das System berechnet den Schaden der Angriffe.
5. Das System errechnet, dass der Gegner besiegt wurde.
6. Das System zeigt dem Nutzer erhaltene Erfahrungspunkte an.
7. Das System zeigt dem Nutzer das hinterlassene Item an.
8. Das System zeigt dem Nutzer hinterlassene Währung an.
9. Der Nutzer weist das System an, Währung aufzuheben.
10. Das System weist dem Charakter Währung zu.
11. Der Nutzer weist das System an, das Item aufzuheben.
12. Das System weist dem Charakter das Item zu.

Szenario 2 ***„Charaktererstellung“*** (Erfüllt Ziel 3.1.3)

1. Der Nutzer weist das System an, einen neuen Charakter zu erstellen.
2. Das System fordert den Nutzer auf, eine von vier Klassen zu wählen.
3. Der Nutzer wählt die zweite Klasse.
4. Das System fordert den Nutzer auf, den Charakter zu benennen.
5. Der Nutzer gibt den Namen des Charakters ein.
6. Das System fordert der Nutzer auf, seine Eingabe zu bestätigen.
7. Der Nutzer bestätigt seine Eingabe.
8. Das System erzeugt einen neuen Spielstand und lädt Charakter und Spielwelt.

Szenario 3 ***„Spielstand speichern“*** (Erfüllt Ziel 3.4)

1. Der Nutzer hat den ersten Bereich beendet.
2. Das System speichert den aktuellen Spielstand.
3. Das System fordert den Nutzer auf, weiter zu Spielen oder das Spiel zu beenden.
4. Der Nutzer wählt „Spiel beenden“.
5. Das System beendet das Spiel.

Szenario 4 ***„Auktionshaus“*** (Erfüllt Ziel 4.1.1.1)

1. Der Nutzer weist das System an, das Auktionshaus zu öffnen.
1. Das System zeigt eine Übersicht des Auktionshauses an.
1. Der Nutzer stellt ein Item zum Verkauf bereit.
2. Das System fordert den Nutzer auf, einen Preis festzulegen.
3. Der Nutzer gibt einen Betrag ein.
4. Das System fordert den Nutzer auf, seine Eingabe zu bestätigen.
5. Der Nutzer bestätigt seine Eingabe.
6. Das System stellt das Item im Auktionshaus zum Kauf zur Verfügung

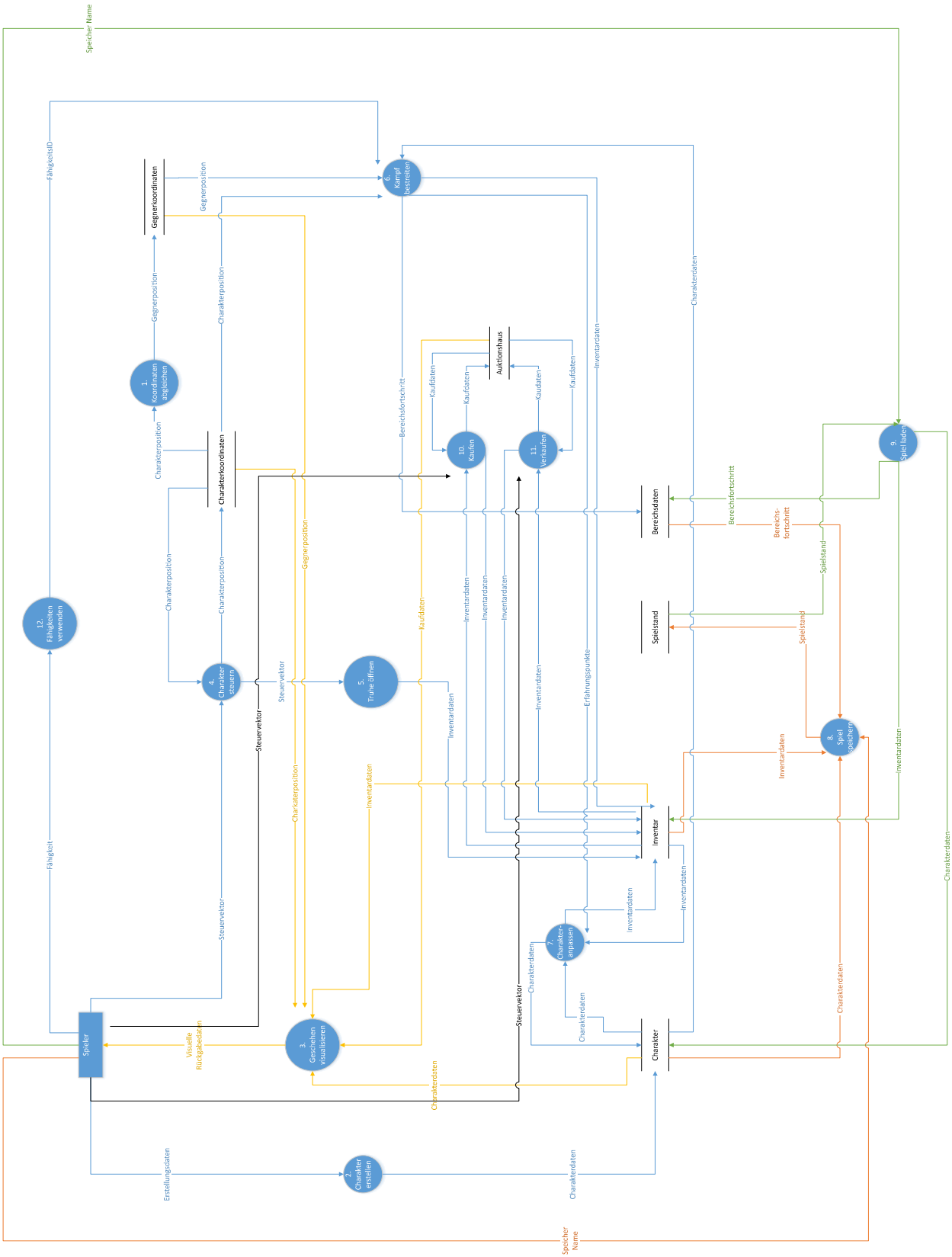
Szenario 5 „Herumstehende Truhen“ (Erfüllt Ziel 2.5)

1. Der Nutzer bewegt den Charakter durch die Spielwelt
2. Das System zeigt dem Nutzer eine in der Spielwelt stehende Truhe an.
3. Der Nutzer bewegt den Charakter neben die Truhe.
4. Der Nutzer weist das System an, die Truhe zu öffnen.

5. Das System zeigt dem Nutzer den Inhalt der Truhe an.
6. Der Nutzer weist das System an, den Truheninhalt aufzuheben.
7. Das System weist dem Charakter des Nutzers den Truheninhalt zu.

3. Logischer Architekturentwurf

3.1 Datenflussdiagramm



3.2 Mini Spezifikation

1. Koordinaten abgleichen

1. Der Prozess erhält vom Speicher *Charakterkoordinaten* die Charakerterposition
2. Der Prozess wandelt diese in Gegnerpositionen um.
3. Der Prozess speichert Gegnerpositionen im Speicher *Gegnerkoordinaten.*

2. Charakter erstellen

1. Der Prozess erhält vom *Spieler* Erstellungsdaten.
2. Der Prozess generiert aus Name, Charakterklasse, Geschlecht und Charakterdaten.
3. Der Prozess speichert Charakterdaten im Charakterdatenspeicher.

3. Geschehen visualisieren

1. Der Prozess erhält vom Speicher *Charakter* Charakterdaten.
2. Der Prozess erhält vom Speicher *Inventar* Truhenrückgabedaten.
3. Der Prozess erhält vom Speicher *Auktionshaus* Handelsrückgabedaten.
4. Der Prozess erhält vom Speicher *Gegnerkoordinaten* die Gegnerposition.
5. Der Prozess erhält vom Speicher *Charakterkoordinaten* die Charakterposition
6. Der Prozess wandelt alle Daten in visuelle Rückgabedaten.
7. Der Prozess sendet visuelle Rückgabedaten an den Spieler.

4. Charakter steuern

1. Der Prozess erhält vom Spieler Steuerbefehle.
2. Der Prozess erhält aus dem Speicher *Charakterkoordinaten* die Charakterposition.
3. Der Prozess wandelt den Steuervektor und Charakterposition in neue Charakterposition um.
4. Der Prozess speichert Charakterposition in *Charakterkoordinaten*
5. Der Prozess sendet einen Steuervektor an *Truhe öffnen.*
6. Der Prozess sendet einen Steuervektor an *Kampf bestreiten.*

5. Truhe öffnen

1. Der Prozess erhält vom Prozess *Charakter steuern* einen Steuervektor.
2. Der Prozess wandelt den Steuervektor in Inventardaten um.
3. Der Prozess speichert die Inventardaten im Speicher *Inventar.*

6. Kampf bestreiten

1. Der Prozess erhält von *Fähigkeiten verwenden* FähigkeitsID.
2. Der Prozess erhält aus dem Speicher *Charakterkoordinaten* die Charakterposition.
3. Der Prozess erhält aus dem Speicher Gegnerkoordinaten die Gegnerposition.
4. Der Prozess erhält aus dem Speicher *Charakter* Charakterdaten.
5. Der Prozess wandelt den Steuervektor, Charakterdaten, Gegnerwerte und FähigkeitsID in Erfahrungspunkte und Inventardaten um.
6. Der Prozess sendet die Erfahrungspunkte an *Charakter anpassen.*

7. Der Prozess speichert die Inventardaten im Speicher *Inventar.*

7. Charakter anpassen

1. Der Prozess lädt aus dem Speicher *Charakter* Charakterdaten.
2. Der Prozess lädt erhält von *Kampf bestreiten* Erfahrungspunkte.
3. Der Prozess lädt aus dem Speicher *Inventar* Inventardaten.
4. Der Prozess erzeugt neue Charakterdaten.
5. Der Prozess speichert diese im Speicher *Charakter.*

8. Spiel speichern

1. Der Prozess lädt aus dem Speicher *Charakter* Charakterdaten
2. Der Prozess lädt aus dem Speicher *Inventar* Inventardaten.
3. Der Prozess lädt aus dem Speicher *Bereichsdaten* Bereichsfortschritt
4. Der Prozess generiert einen neuen Spielstand
5. Der Prozess speichert ihn im Datenspeicher *Spielstand.*

9. Spiel laden

1. Der Prozess lädt einen Spielstand aus dem Speicher *Spielstand.*
2. Der Prozess erzeugt aus dem Spielstand den Bereichsfortschritt.
3. Der Prozess speichert den Bereichsfortschritt im Speicher Bereich.
4. Der Prozess erzeugt aus dem Spielstand die Charakterdaten.
5. Der Prozess speichert die Charakterdaten im Speicher *Charakter.*
6. Der Prozess erzeugt aus dem Spielstand die Inventardaten.
7. Der Prozess speichert die Inventardaten im Speicher *Inventar.*

10. Kaufen

1. Der Prozess erhält vom Speicher *Inventar* Inventardaten.
2. Der Prozess erhält vom Spieler einen Steuervektor.
3. Der Prozess erhält vom Speicher *Auktionshaus* Kaufdaten.
4. Der Prozess erstellt aus den erhaltenen Daten neue Kaufdaten.
5. Der Prozess speichert die neuen Kaufdaten im Speicher *Auktionshaus*.
6. Der Prozess erstellt aus den erhaltenen Daten neue Inventardaten.
7. Der Prozess speichert die neuen Inventardaten im Speicher *Inventar.*

11. Verkaufen

1. Der Prozess erhält vom Speicher *Inventar* Inventardaten.
2. Der Prozess erhält vom Speicher *Auktionshaus* Kaufdaten.
3. Der Prozess erhält vom Spieler einen Steuervektor.
4. Der Prozess erstellt aus den erhaltenen Daten neue Kaufdaten.
5. Der Prozess speichert die neuen Kaufdaten im Speicher *Auktionshaus.*
6. Der Prozess erstellt aus den erhaltenen Daten neue Inventardaten.
7. Der Prozess speichert die neuen Inventardaten im Speicher *Inventar.*

12. Fähigkeiten verwenden

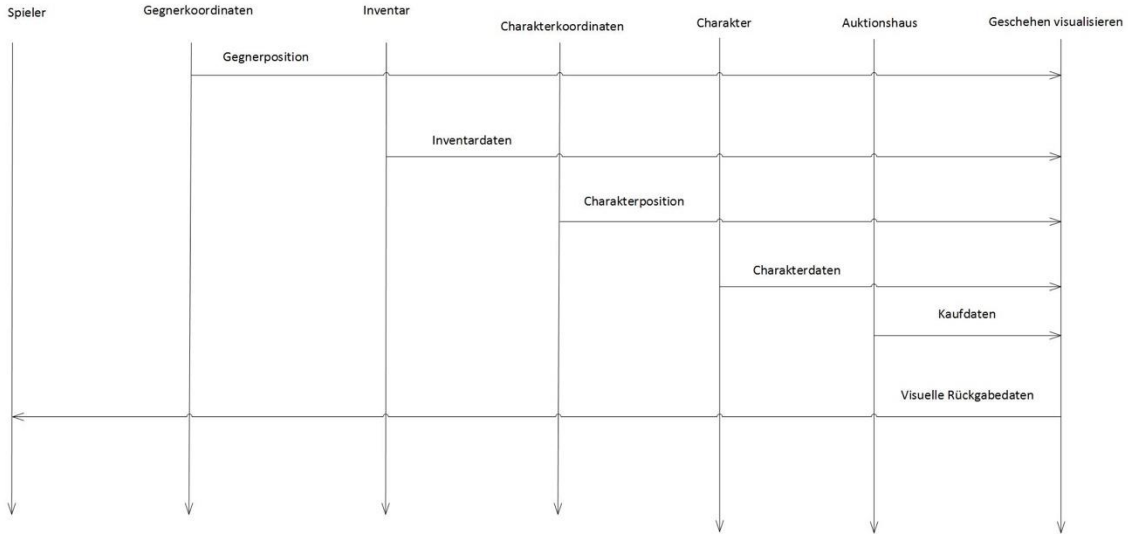
1. Der Prozess erhält vom *Spieler* Fähigkeit.
2. Der Prozess wandelt diese in eine FähigkeitsID um.
3. Der Prozess sendet die FähigkeitsID an *Kampf bestreiten.*

3.3 **Data Dictionary**

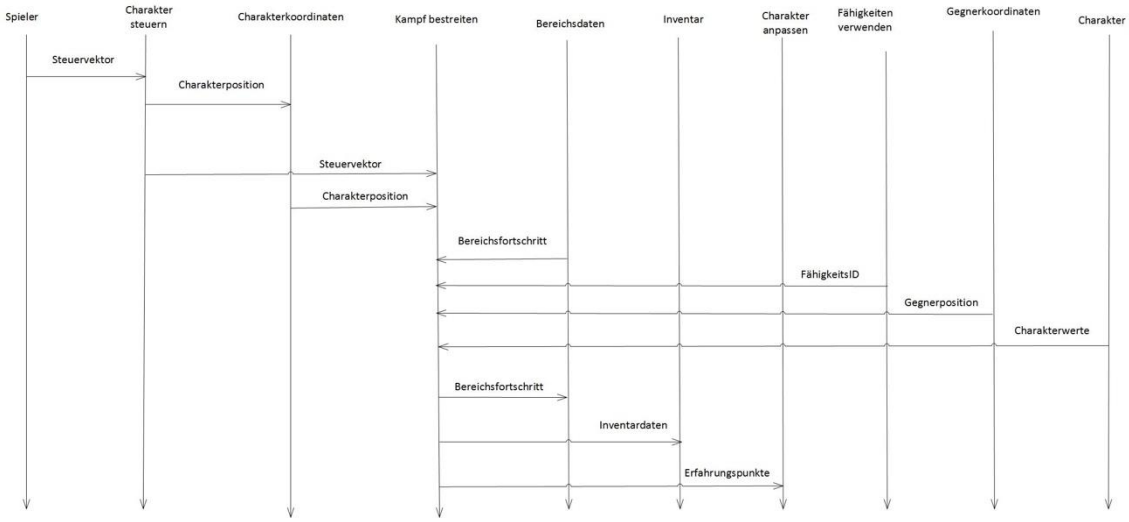
Bereichsfortschritt	=	BereichsID + GegnerID
BereichsID	=	1{Ziffer}1
Charakterdaten	=	Name + Charakterklasse + Geschlecht + Lebenspunkte + Charakterwerte + Erfahrungspunkte
Charakterklasse	=	[Klasse 1 | Klasse 2 | Klasse 3 | Klasse 4]
Charakterposition	=	X-Koordinate + Y-Koordinate
Charakterwerte	=	Stärke + Widerstand
Erfahrungspunkte	=	1{Ziffer}
Erstellungsdaten	=	Name + Charakterklasse + Geschlecht
Fähigkeiten	=	[Fähigkeit 1 | Fähigkeit 2 | Fähigkeit 3 | Fähigkeit 4]
FähigkeitsID	=	1{Ziffer}2
Gegnerposition	=	X-Koordinate + Y-Koordinate
Gegnerwerte	=	Stärke + Widerstand
Geschlecht	=	[Männlich | Weiblich]
Inventardaten	=	{Itemdaten + Währung}
Itemdaten	=	Itemname + ItemID + Itemwerte
ItemID	=	1{Ziffer}3
Itemname	=	1{Zeichen}25
Itemwerte	=	Stärke + Widerstand
Kampfdaten	=	Charakterdaten + Gegnerdaten
Kaufdaten	=	{Itemdaten + Währung}
Lebenspunkte	=	1{Ziffer}
Name	=	2{Zeichen}15
Speichername	=	1{Zeichen}25
Spielstand	=	Inventardaten + Charakterdaten + Bereichsfortschritt
Stärke	=	1{Ziffer}
Steuervektor	=	2{Ziffer}
Truhenrückgabedaten	=	{(Itemdaten) + (Währung)}
Visuelle Rückgabedaten	=	Charakterdaten + Inventardaten + Charakterposition + Gegnerposition + Kaufdaten
Währung	=	1{Ziffer}
Widerstand	=	1{Ziffer}
X-Koordinate	=	1{Ziffer}
Y-Koordinate	=	1{Ziffer}

3.4 Message Sequence Charts

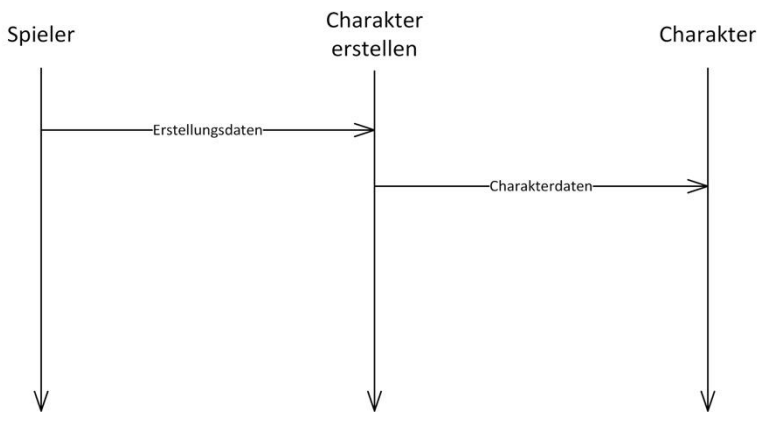
3.4.1 bMSC 0: „Geschehen visualisieren“



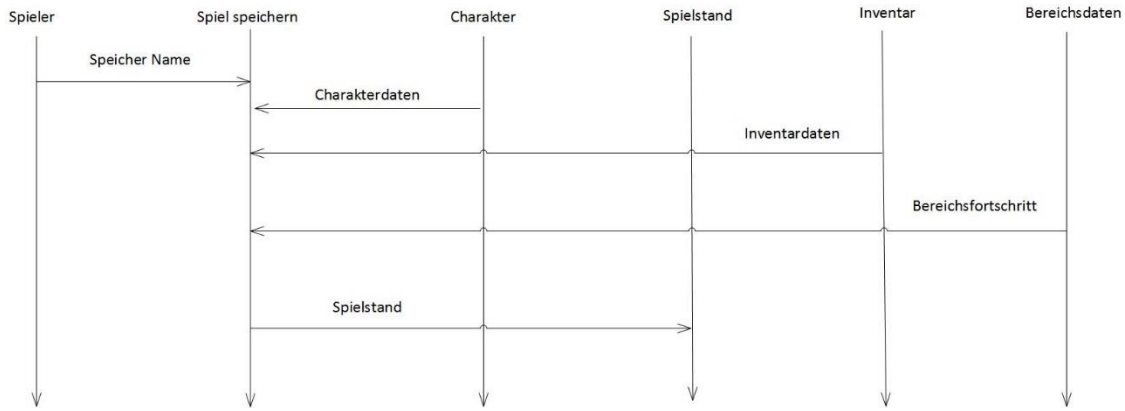
3.4.2 bMSC 1: Szenario 1



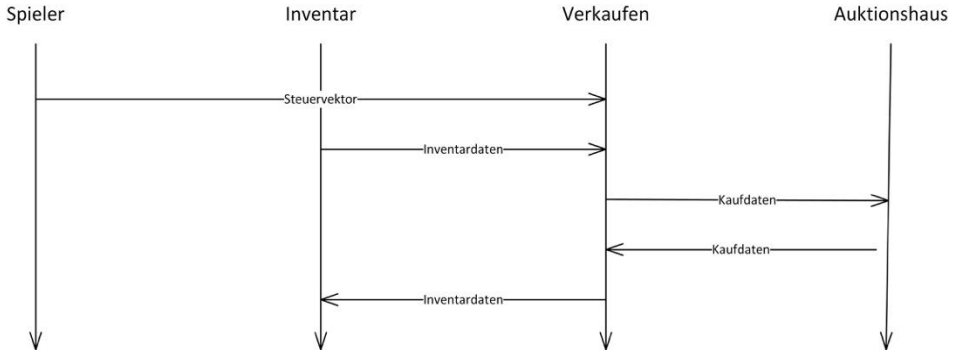
3.4.3 bMSC 2: Szenario 2



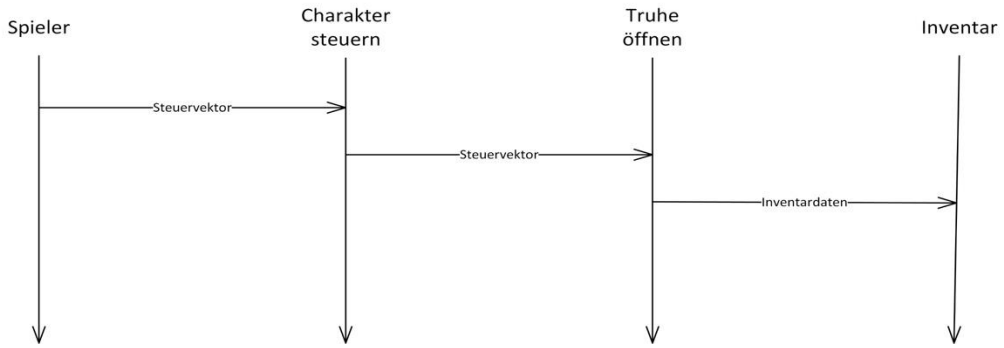
3.4.4 bMSC 3: Szenario 3



3.4.5 bMSC 4: Szenario 4



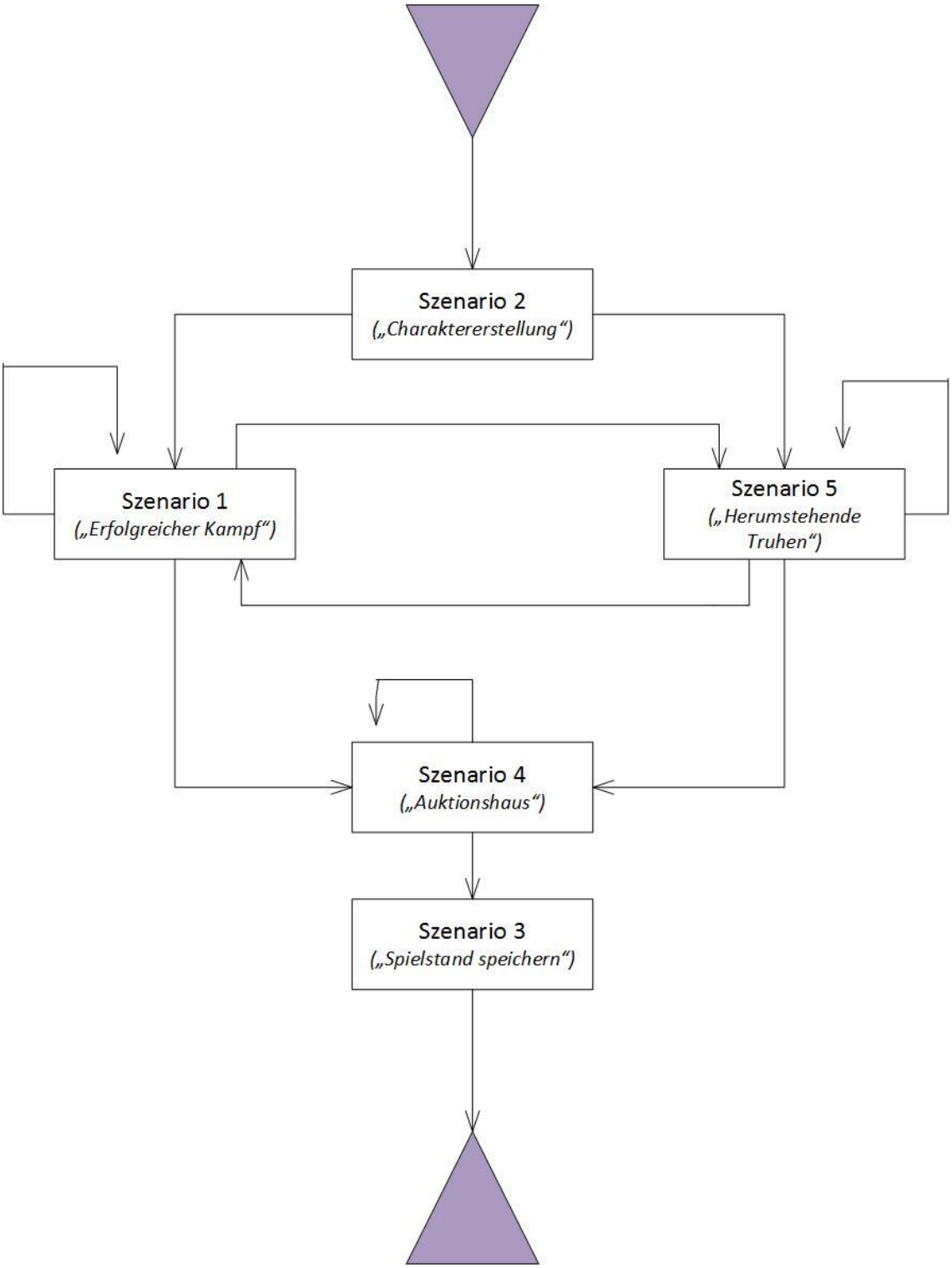
3.4.6 bMSC 5 Szenario 5



3.4.7 Abbildung der Szenarien auf Message Sequence Charts

Szenario 1	BMSC 1
	BMSC 0 "*Geschehen visualisieren*"
Szenario 2	BMSC 2
	BMSC 0 "*Geschehen visualisieren*"
Szenario 3	BMSC 3
Szenario 4	BMSC 4
	BMSC 0 "*Geschehen visualisieren*"
Szenario 5	BMSC 5
	BMSC 0 "*Geschehen visualisieren*"

3.4.8 hMSC


Szenario 2
(„Charaktererstellung“)
Szenario 1
(„Erfolgreicher Kampf“)
Szenario 5
(„Herumstehende Truhen“)
Szenario 4
(„Auktionshaus“)
Szenario 3
(„Spielstand speichern“)

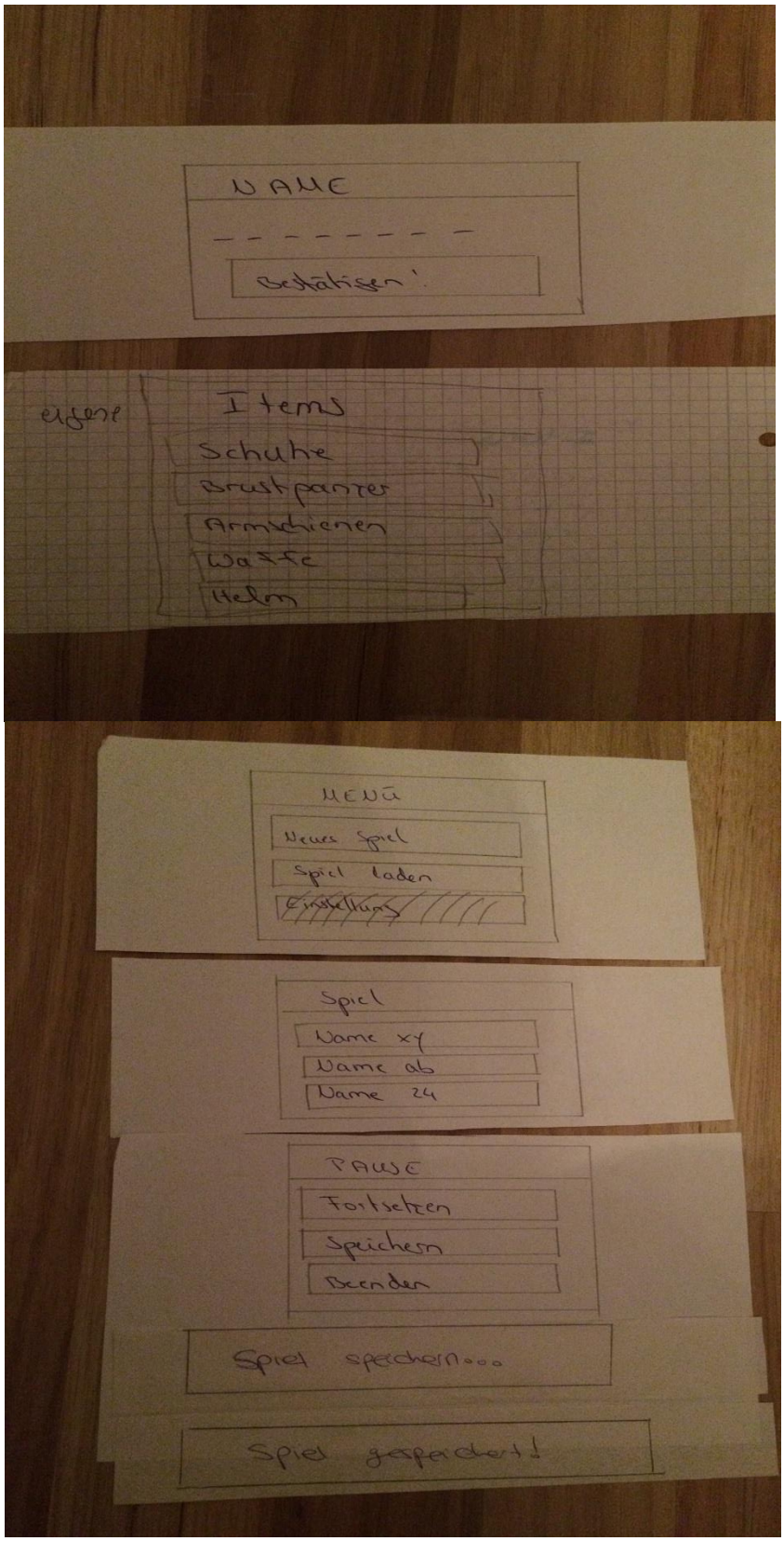
4. Technischer Architekturentwurf

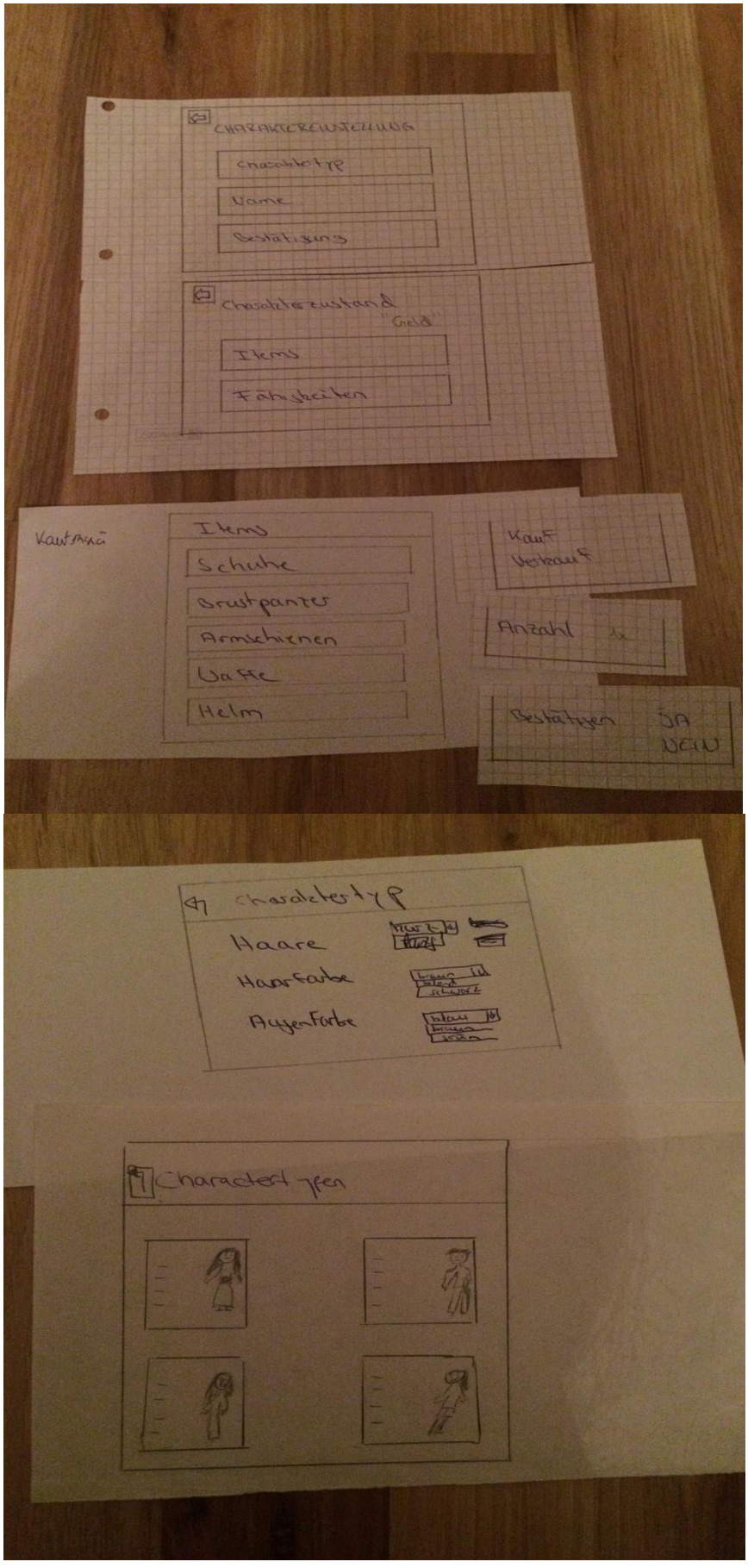
Dieser Abschnitt wird von der Partner-Gruppe ausgefüllt, die das Projekt auch am Ende implementieren wird. Vor der Bearbeitung dieses Abschnitts wird das Dokument an die Partner-Gruppe übergeben.

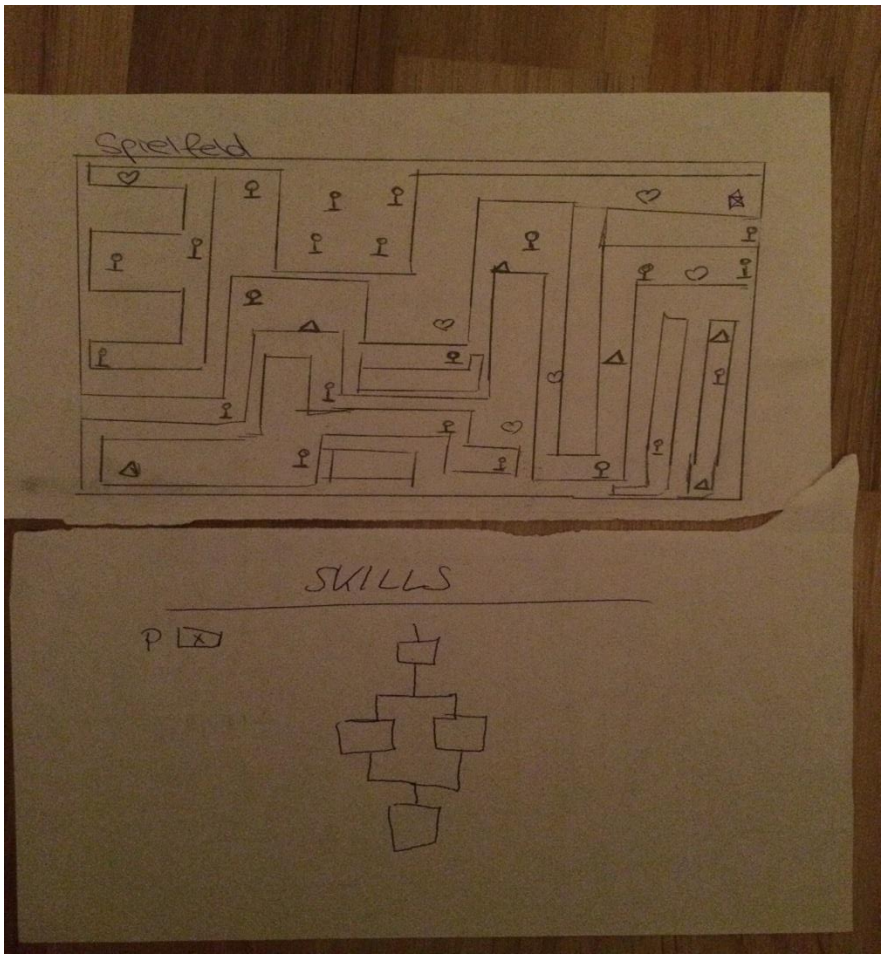
Auf der technischen Ebene erfolgt der kreative Schritt der Konstruktion des technischen Systems. Hierbei liegt der kreative Schritt besonders in der Umsetzung der logischen Architektur der DFDs in ein technisches System mit „echten" Komponenten.

4.1 GUI-Papierprototyp

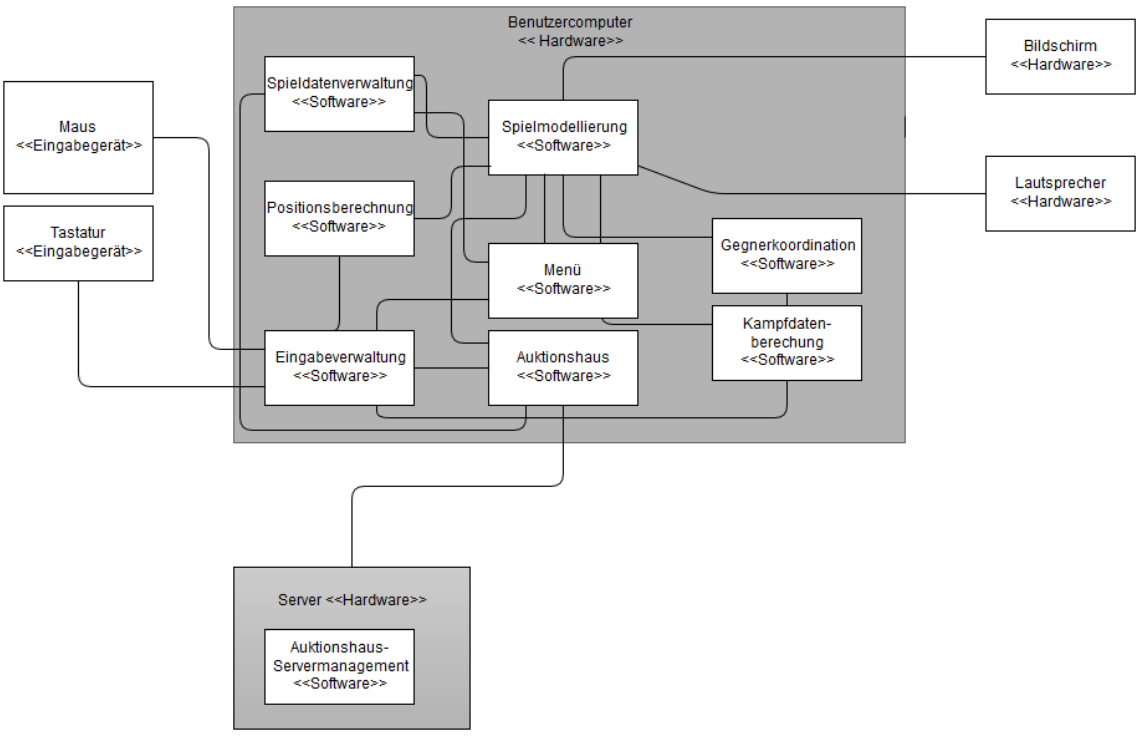
4.1.1 Screen „<Name des Screens>“




Charaktertyp
Haare
Haarfarbe
Augenfarbe
Charakterwahl



4.2 Technisches Konzept



4.2.1 Maus
Die Maus dient als Steuerelement zur Bewegung und Interaktion im Spiel.

4.2.2 Tastatur
Die Tastatur dient primär für die Aktivierung von Fähigkeiten. Außerdem können durch Shortcuts einzelne Spielinhalte geöffnet werden.

4.2.3 Auktionshaus
Verwaltet lokal die Kommunikation zum Server und dem Transfer der Daten.

4.2.4 Spieldatenverwaltung
Die Spieldatenverwaltung speichert alle lokalen Spieldaten (Position des Charakters, Stats, Items, Fortschritt etc.).

4.2.5 Bildschirm
Der Bildschirm ist das Anzeigegerät, welches dem Benutzer alle visuellen Grafiken anzeigt.

4.2.6 Lautsprecher
Die Lautsprecher geben alle spielinternen Soundeffekte ab.

4.2.7 Spielmodellierung
Die Spielmodellierung koordiniert alle systeminternen Prozesse und fügt diese zu einem visuellen Modell zusammen.

4.2.8 Gegnerkoordination
Die Gegnerkoordination dient zur Steuerung der Gegner durch das System.

4.2.9 Kampfdatenberechnung
Diese Softwarekomponente berechnet den Schaden der jeweiligen Charaktere im Kampf.

4.2.10 Positionsberechnung
Diese Softwarekomponente berechnet die Position des Spielers.

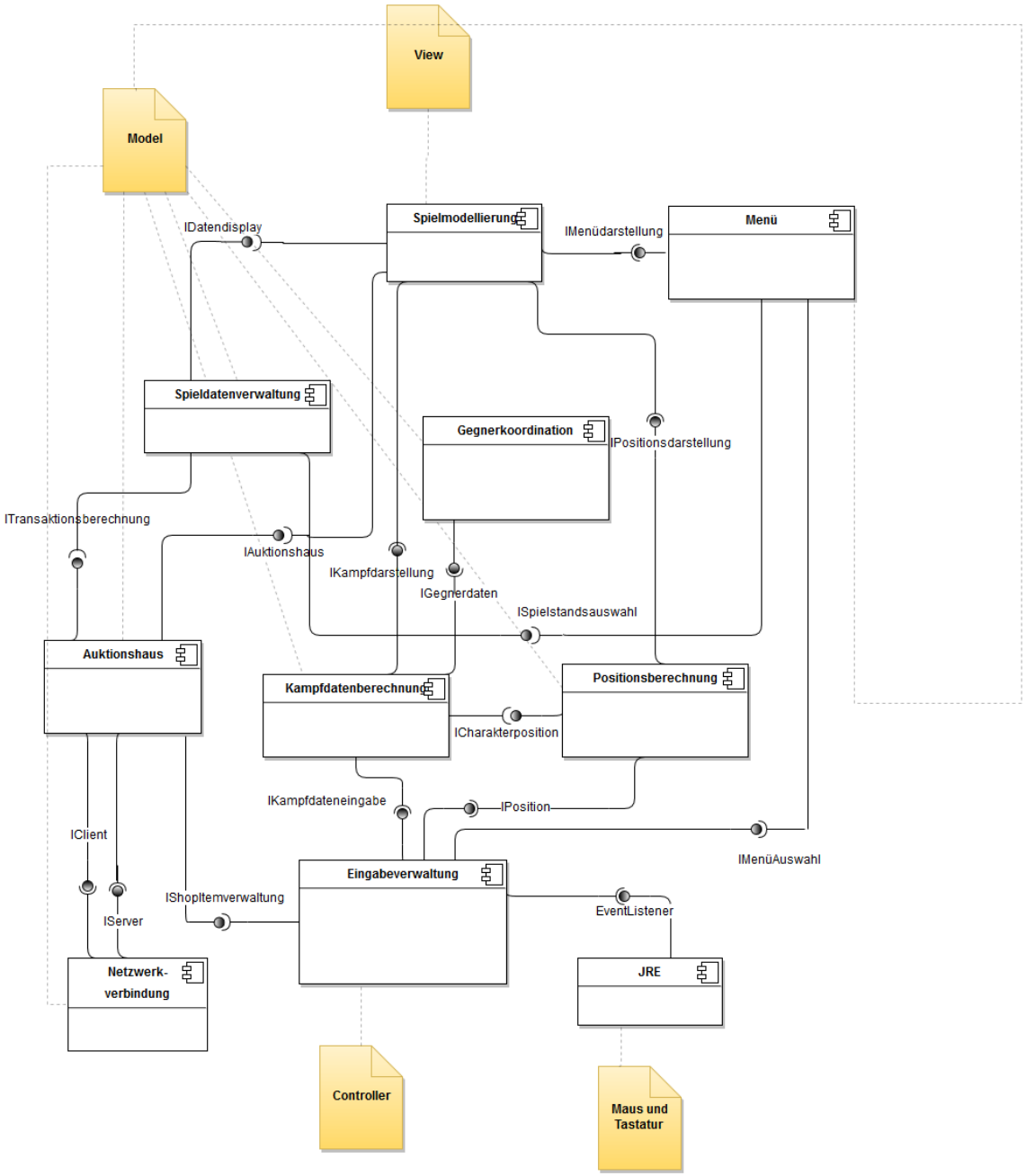
4.2.11 Eingabeverwaltung
Die Eingabeverwaltung wandelt die systemabhängigen Eingabebefehle von Tastatur und Maus zu systemunabhängigen Steuerbefehlen um.

4.2.12 Menü
Das Menü verwaltet die Eingaben des Spielers innerhalb aller spielinternen Menüs und stellt vom Spieler ausgewählte Funktionen bereit.

4.2.13 Auktionshaus Servermanagement
Das Auktionshaus koordiniert den Kauf und Verkauf von spielinternen Gütern zwischen den Charakteren über eine Netzwerkverbindung.

4.3 Komponentendiagramm

Die technischen Komponenten zeigen die Realisierung des Systems. Dazu wird hier nun beschrieben, welche echten Komponenten später im System zu finden sind und damit implementiert werden. Sowohl zu jeder technischen Komponente als auch zu jedem Interface soll es eine kurze Beschreibung geben. Zu jeder Komponente soll angegeben werden, welche Funktionen umgesetzt werden. Zur Beschreibung eines Interfaces gehören die Zuordnung zu anbietenden und nutzenden Komponenten sowie die Auflistung aller Methodenköpfe inklusive ihrer Übergabeparameter und Rückgabewerte.



4.3.1 Komponentenbeschreibung

1. **Spielmodellierung**

   Stellt das Spielgeschehen graphisch dar.

2. **Spieldatenverwaltung**

   Speichert alle nötigen Spieldaten, um das Spiel zu einem späteren Zeitpunkt an der gleichen Stelle fortzusetzen.

3. **JRE**

   Stellt Maus- und Tastaturlistener zur Verfügung

4. **Auktionshaus**

   Ermöglicht den Handel mit Items über eine Netzwerkanbindung

5. **Eingabeverwaltung**

   Verwaltet die Eingaben des Nutzers und wandelt sie in vom Spielsystem nutzbare Informationen um.

6. **Positionsberechnung**

   Berechnet die Position der Spielfigur in ihrer Spielwelt.

7. **Menü**

   Stellt das Spielmenü mit allen Einstellungs-, Lade- und Speichermöglichkeiten dar.

8. **Kampfdatenberechnung**

   Berechnet die Handlungen von Gegnern und vom Charakter in Kämpfen

9. **Gegnerkoordination**

   Verwaltet alle Gegner des Spielgeschehens

10. **Netzwerkverbindung**

    Stellt die Verbindung zwischen Client und Server her.

4.3.2 Interfacebeschreibung

1. **IMenüDarstellung**

   Liefert Daten, um das Spielmenü grafisch darzustellen.

   Anbietende Komponente: Menü

   Nutzende Komponente: Spielmodellierung

   Sprite getSprite();

2. **IDatenDisplay**

   Liefert Daten, um ein gespeichert Spiel neu zu laden und grafisch darzustellen.

   Anbietende Komponente: Spieldatenverwaltung

   Nutzende Komponente: Spielmodellierung

   Spielstand getSpielstand();

   Spielstand setSpielstand();

3. **IAuktionshaus**

   Liefert Daten, um das Auktionshaus und seine Inhalte grafisch darzustellen.

   Anbietende Komponente: Auktionshaus

   Nutzende Komponente: Spielmodellierung

   Item getItem();

   Item setItem();

   int getWährung();

   int setWährung();

4. **IPositionsDarstellung**

   Liefert Daten, um die Spielfigur an der richtigen Stelle in der Spielwelt zu bewegen und darzustellen.

   Anbietende Komponente: Positionsberechnung

   Nutzende Komponente: Spielmodellierung

   int[] getPosition();

5. **ISpielstandauswahl**

   Liefert Daten, um zu ladende Spiele im Menü anzeigen zu können.

   Anbietende Komponente: Spieldatenverwaltung

   Nutzende Komponente: Menü

   Spielstand getSpielstand();

6. **ITransaktionsBerechnung**

   Liefert Daten, um vom Spieler nutzbare Items im Auktionshaus kaufen zu können.

   Anbietende Komponente: Auktionshaus

Nutzende Komponente: Spieldatenverwaltung

Item getItem();

7. **IEventListener**

   Liefert Eingabedaten des Nutzers.

   Anbietende Komponente: JRE

   Nutzende Komponente: Eingabeverwaltung

   boolean isKeyPressed(int a);

   int getX();

   int getY();

8. **IShopItemverwaltung**

   Liefert Daten, um im Auktionshaus die vom Spieler ausgewählten Items zu verkaufen, bzw. zu kaufen.

   Anbietende Komponente: Eingabeverwaltung

   Nutzende Komponente: Auktionshaus

   int[] getMousevector();

   boolean isMouseClicked();

   int getInput();

9. **IPosition**

   Liefert Daten, um die neue Position der Spielfigur auf Grund der Eingabedaten des Nutzers berechnen zu können.

   Anbietende Komponente: Eingabeverwaltung

   Nutzende Komponente: Positionsberechnung

   int getInput();

10. **IMenüAuswahl**

    Liefert Inhalte des Menüs, die der Nutzer auswählen kann.

    Anbietende Komponente: Eingabeverwaltung

    Nutzende Komponente: Menü

    int[] getMousevector();

    boolean isMouseClicked();

    int getInput();

11. **IKampfdateneingabe**

    Liefert die Kampfdaten des Nutzers, auf Grund derer die Kampfaktionen der Gegner berechnet werden.

    Anbietende Komponente: Eingabeverwaltung

    Nutzende Komponente: Kampfdatenberechnung

int getInput();

12. **IGegnerdaten**

Liefert Daten der Gegner (Stärke, Position, Spezialfähigkeiten, ...) auf Grund derer die Kampfaktionen der Gegner berechnet werden.

Anbietende Komponente: Kampfdatenberechnung

Nutzende Komponente: Gegnerkoordination

int[] getGegnerkoordinaten(Gegner a);

Gegner[] getGegner();

13. **IClient**

Liefert lokale Auktionshausdaten, die an den Server geschickt werden.

Anbietende Komponente: Auktionshaus

Nutzende Komponente: Netzwerkverbindung

14. **IServer**

Liefert Auktionshausdaten vom Server, die vom lokalen Nutzer benutzt werden.

Anbietende Komponente: Netzwerkverbindung

Nutzende Komponente: Auktionshaus

15. **IKampfdarstellung**

Liefert Grafiken von Gegnern, benutzen Angriffe und Fähigkeiten.
Anbietende Komponente: Kampfdatenberechnung
Nutzende Komponente: Spielmodellierung
Sprite[] getAngriff();
Sprite[] getFähigkeit();
Sprite[] getGegner();

16. **ICharakterposition**
Liefert die Position des Charakters.
Anbietende Komponente: Positionsberechnung
Nutzende Komponente: Kampfdatenberechnung
int[] getPosition;

5.

6. Testartefakte

6.1 Modultest

6.1.1 Testspezifikation

7. Modultestfall 1: <Kurzbezeichnung MTF-1>

Testziel	

Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	
Bestehens Kriterien	

8. Modultestfall n: <Kurzbezeichnung MTF-n>

Testziel	
Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	
Bestehens Kriterien	

8.1.1 Testergebnisse

9. Testprotokoll Modultestfall 1 (1. Testdurchführung)

Testziel	
Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	
Bestehens Kriterien	
Datum	
Tester	
Version der Software	
Testtreiber	
Testsystem & -umgebung	
Testurteil	

10. Testprotokoll Modultestfall 1 (n. Testdurchführung)

Testziel	
Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	
Bestehens Kriterien	
Datum	
Tester	
Version der Software	
Testtreiber	
Testsystem & -umgebung	
Testurteil	

11. Testprotokoll Modultestfall n (1. Testdurchführung)

Testziel	
Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	

Bestehens Kriterien	
Datum	
Tester	
Version der Software	
Testtreiber	
Testsystem & -umgebung	
Testurteil	

12. Testprotokoll Modultestfall n (n. Testdurchführung)

Testziel	
Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	
Bestehens Kriterien	
Datum	
Tester	
Version der Software	
Testtreiber	
Testsystem & -umgebung	
Testurteil	

12.1 Systemtest

12.1.1 Testspezifikation

13. Systemtestfall 1: <Kurzbezeichnung STF-1>

Szenario		
Schritt	Aktion (User)	Erwartete Reaktion (System)
1		
2		
3		
4		
…		

14. Systemtestfall n: <Kurzbezeichnung STF-n>

Szenario		
Schritt	Aktion (User)	Erwartete Reaktion (System)
1		
2		
3		
4		
…		

14.1.1 Testergebnisse

15. Testprotokoll Systemtestfall 1 (<1. Testdurchführung>)

Datum				
Tester				
Version der Software				
Szenario				
Schritt	Aktion (User)	Erwartete Reaktion (System)	Tatsächliche Reaktion (System)	√ / X
1				
2				
3				
4				
…				
Testurteil				

16. Testprotokoll Systemtestfall 1 (<n. Testdurchführung>)

Datum				
Tester				
Version der Software				
Szenario				
Schritt	Aktion (User)	Erwartete Reaktion (System)	Tatsächliche Reaktion (System)	√ / X
1				
2				
3				
4				
…				
Testurteil				

17. Testprotokoll Systemtestfall n (Version <1. Testdurchführung>)

Datum				
Tester				
Version der Software				
Szenario				
Schritt	Aktion (User)	Erwartete Reaktion (System)	Tatsächliche Reaktion (System)	√ / X
1				
2				
3				
4				
…				
Testurteil				

18. Testprotokoll Systemtestfall n (Version <n. Testdurchführung>)

Datum	
Tester	

Version der Software				
Szenario				
Schritt	Aktion (User)	Erwartete Reaktion (System)	Tatsächliche Reaktion (System)	√ / X
1				
2				
3				
4				
…				
Testurteil				