

# SEP – Hauptaufgabe

## PROJEKTMAPPE DES PROJEKTES

### <SEPMAN>

#### Spezifikation des Projektes

*Gruppe: L*

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

#### Systemdesign des Projektes

*Gruppe K:*

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

# 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 Computer zur Durchführung eines bestimmten Verhaltens konfiguriert werden muss.<sup>1</sup> 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:

<i>Dies ist eine Hilfestellung.</i>
-------------------------------------

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 wiederfindet.

## **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!!!

---

<sup>1</sup>vgl. B. Kovitz: Practical Software Requirements: A Manual of Content & Style; Manning 1998

# Inhaltsverzeichnis

<b>PROJEKTBESCHREIBUNG (&lt;PROJEKTNAME&gt;)</b> .....	<b>4</b>
<b>ANFORDERUNGSDEFINITION</b> .....	<b>6</b>
ZIELMODELL .....	6
KONTEXTMODELL / SPIELMODELL .....	9
<Akteur 1/Ext. System 1> .....	9
<Akteur n/Ext. System n> .....	9
SZENARIEN .....	10
<Name Szenario 1> .....	10
<Name Szenario n> .....	10
<b>LOGISCHER ARCHITEKTURENTWURF</b> .....	<b>11</b>
DATENFLUSSDIAGRAMM .....	11
MINI SPEZIFIKATION .....	11
<Name Prozess 1> .....	11
<Name Prozess n> .....	11
DATA DICTIONARY .....	11
MESSAGE SEQUENCE CHARTS .....	11
bMSCs .....	11
bMSC-1: <Name des bMSC> .....	11
bMSC-n: <Name des bMSC> .....	11
Abbildung der Szenarien auf Message Sequence Charts .....	11
hMSC .....	12
<b>TECHNISCHER ARCHITEKTURENTWURF</b> .....	<b>13</b>
GUI-PAPIERPROTOTYP .....	13
Screen „<Name des Screens>“ .....	13
TECHNISCHES KONZEPT .....	13
<Name Komponente 1> .....	13
<Name Komponente n> .....	13
KOMPONENTENDIAGRAMM .....	14
Komponentenbeschreibung .....	14
<Name Komponente 1> .....	14
<Name Komponente n> .....	14
Interfacebeschreibung .....	14
<Name Interface 1> .....	14
<Name Interface n> .....	14
<b>TESTARTEFAKTE</b> .....	<b>15</b>
MODULTEST .....	15
Testspezifikation .....	15
Modultestfall 1: <Kurzbezeichnung MTF-1> .....	15
Modultestfall n: <Kurzbezeichnung MTF-n> .....	15
Testergebnisse .....	15
Testprotokoll Modultestfall 1 (1. Testdurchführung) .....	15
Testprotokoll Modultestfall 1 (n. Testdurchführung) .....	15
Testprotokoll Modultestfall n (1. Testdurchführung) .....	15
Testprotokoll Modultestfall n (n. Testdurchführung) .....	16
SYSTEMTEST .....	16
Testspezifikation .....	16
Systemtestfall 1: <Kurzbezeichnung STF-1> .....	16
Systemtestfall n: <Kurzbezeichnung STF-n> .....	16
Testergebnisse .....	16
Testprotokoll Systemtestfall 1 (<1. Testdurchführung>) .....	16
Testprotokoll Systemtestfall 1 (<n. Testdurchführung>) .....	17
Testprotokoll Systemtestfall n (Version <1. Testdurchführung>) .....	17
Testprotokoll Systemtestfall n (Version <n. Testdurchführung>) .....	17

## **Projektbeschreibung (<Projektname>)**

<p><i>In diesem Abschnitt soll die Projektbeschreibung abgedruckt werden, die ihr als Aufgabenbeschreibung von eurem Betreuer erhalten habt. Sie dient als initiales Anforderungsdokument für eure Spezifikationsaktivitäten.</i></p>
---

<Eure Projektbeschreibung>

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.

# Anforderungsdefinition

Dieser Abschnitt soll jeweils von der Gruppe, die für die Spezifikation des Projekts zuständig ist, ausgefüllt werden.

Auf dieser Ebene wird das System als Ganzes betrachtet. Jedoch gibt es kein Wissen über die Abläufe im System oder über die genauen Funktionalitäten.

## Zielmodell

In diesem Abschnitt sollen die Ziele des Systems beschrieben werden. Ein Zielbaum (oder Zielgraph) stellt dabei die geeignetste Methode zur Darstellung dar. Eine textuelle Beschreibung aller Ziele detailliert das Modell entsprechend.

Der Zielbaum stellt ein Artefakt dar, das die Spezifikation des logischen und des technischen Systemdesigns überspannt. Von daher muss er zweimal während der Projektlaufzeit bearbeitet werden. Zuerst wird er während der Spezifikation des logischen Systemdesigns erstellt, und dann während der Spezifikation des technischen Systemdesigns verfeinert, erweitert und aktualisiert. Das Zielmodell enthält damit Informationen unterschiedlicher Detaillierung und stellt die enthaltenen Ziele in ihren Beziehungen dar.

Aufgrund der Trennung von Anforderungsspezifikation und technischem Systemdesign wird folgende Dokumentationsrichtlinie verwendet: Das Zielmodell wird in der Spezifikationsphase in diesem Abschnitt **Fehler! Es wurde kein Textmarkenname vergeben.** erstellt und fortwährend aktualisiert; nach der Übergabe der Anforderungsspezifikation wird im Rahmen des technischen Systemdesigns das Zielmodell in diesem Abschnitt von der Partner-Gruppe, die das System implementiert, vervollständigt.

Zuletzt noch einige inhaltliche und strukturelle Anmerkungen:

- 1) Ziele beschreiben die Intention eines Akteurs mit einem zu bauenden System.
- 2) Die Akteure der Ziele müssen sich im Kontextmodell des Systems wiederfinden.
- 3) Man unterscheidet zwischen logischen und technischen Zielen. Logische Ziele sollen immer lösungsunabhängig sein (z.B. „Der Nutzer möchte seine Termine verwalten“), während technische Ziele explizit lösungsabhängig sein sollen (z.B. „Die Termine des Nutzers werden mit dem Google-Kalender des Nutzer synchronisiert.“).
- 4) Logische Ziele werden nur während der Spezifikation des logischen Systemdesigns erstellt und technische Ziele nur während der Spezifikation des technischen Systemdesigns erstellt. Das bedeutet auch, dass die logischen und technischen Ziele von zwei verschiedenen Gruppen erstellt werden.
- 5) Man unterscheidet außerdem zwischen Softgoals und Hardgoals. Softgoals sind Ziele, die man nicht objektiv überprüfen kann (z.B. „Die Termine des Nutzers werden übersichtlich dargestellt.“), während Hardgoals Ziele sind, die man explizit objektiv überprüfen kann (z.B. „Es können mindestens 10 Termine des Nutzers gleichzeitig dargestellt werden.“).
- 6) Es gibt also vier verschiedene Arten von Zielen. 1. Logische Ziele, die Hardgoals sind; 2. Logische Ziele, die Softgoals sind; 3. Technische Ziele, die Hardgoals sind; und 4. Technische Ziele, die Softgoals sind.
- 7) Ziele der Ebene 1 des Zielbaums werden in der Form **Z-<x>** nummeriert, wobei x die Nummer des Zielbaums ist.
- 8) Ziele aller weiteren Ebenen werden in der Form **Z-<L/T>-<HG/SG>-<x>.<y>** nummeriert. Es wird angegeben ob es sich um ein logisches (L) oder technisches (T) Ziel, ein Hardgoal (HG) oder ein Softgoal (SG) handelt und welche Nummer (x,y) das Ziel hat.

- 9) Die Eltern-Kind-Beziehung zwischen Zielen wird durch die Punktnotation am Ende der Zielnummer angegeben. So ist das Ziel 3.4.2 das zweite Unterziel des Ziels 3.4, das wiederum ein Unterziel von Ziel 3 ist.
- 10) Wichtig: der Zielbaum ist in jedem Ast nicht auf eine Anzahl Ebenen beschränkt. D.h. der Zielbaum kann in jedem Ast beliebig viele Ebenen aufweisen. Die Nummerierung muss dementsprechend angepasst werden.

Der fertige Zielbaum enthält lediglich Hardgoals als Blätter, um die Überprüfbarkeit zu gewährleisten. <ggf. grafische Repräsentation des Zielmodells>

Erinnerung: Struktur der Ziele: Z<X.Y.Z>-<L|T>-<SG|HG>: <Name der Ziels>

- **Z-1: Rollenauswahl**  
Ein Roboter können sich in einer der 2 Rollen befinden.
  - **Z-L-HG-1.1: SEPMAN**  
Der Roboter kann als SEPMAN fungieren.
    - **Z-L-HG-1.1.1: Spielersteuerung**  
Der Nutzer kann durch Aktionen den SEPMAN steuern.
    - **Z-L-HG-1.1.2: Leben**  
Die Versuche des Spielers werden durch Leben des SEPMAN repräsentiert. Der SEPMAN startet mit 3 Leben.
  - **ZL-HG-1.2: Geist**  
Der Roboter kann als Geist fungieren.
    - **Z-L-HG-1.2.1: Autonomie**  
Der Geist kann sich autonom bewegen.
    - **Z-L-HG-1.2.2: Modus Auswahl**  
Der Geist kann sich in einem der 3 Modi befinden.
      - **Z-L-HG-1.2.2.1: Verfolgung**  
Der Geist verfolgt SEPMAN.
      - **Z-L-HG-1.2.2.2: Zufall**  
Der Geist bewegt sich zufällig.
      - **Z-L-HG-1.2.2.3: Verteidigung**  
Der Geist verteidigt die Power-Ups.
- **Z-2: Spielfeld**  
Das System setzt ein Spielfeld um, auf dem sich die Roboter bewegen.
  - **Z-L-HG-2.1: Knotenpunkte**  
Roboter bewegen sich auf und zwischen Knotenpunkten.
    - **Z-L-HG-2.1.1: Position der Roboter**  
Das System zeigt an auf welchem Knotenpunkt sich der Roboter befindet.
    - **Z-L-HG-2.1.2: Zustand**  
Das System zeigt den Zustand des Knotens an (Normal / Power-Up).
  - **Z-L-HG-2.2: Kanten**  
Roboter sollen nur zwischen Knotenpunkten wechseln, die mit Kanten verbunden sind.
    - **Z-L-HG-2.2.1: (Besuch-)Zustand**  
Das System zeigt an ob eine Kante besucht ist oder nicht.

- **Z-3: Spielregeln**  
Das System folgt Regeln.
  - **Z-L-HG-3.1: Startposition**  
Die Roboter starten auf bestimmten Positionen.
  - **Z-L-HG-3.2: Spielgeschehen**  
Die Roboter handeln nach Regeln.
    - **Z-L-HG-3.2.1: Power-Up**  
Power-Ups können vom SEPMAN aufgesammelt werden.
      - **Z-L-HG-3.2.1.1: Flucht**  
Die Geister flüchten vom SEPMAN, wenn ein Power-Up eingesammelt wurde.
      - **Z-L-HG-3.2.1.2: Kollision mit Power-Up**  
Bei Kollision wird der Geist deaktiviert.
        - **Z-L-HG-3.2.1.2.1 Reaktivierung**  
Nach einer bestimmten Zeit wird der Geist auf seiner Startposition wieder aktiviert.
      - **Z-L-HG-3.2.1.3: Zeitliche Beschränkung**  
Das Power-Up läuft nach einer bestimmten Zeit ab.
    - **Z-L-HG-3.2.2: Kollision ohne Power-Up**  
Die Kollision zwischen SEPMAN und Geist folgt bestimmten Regeln.
      - **Z-L-HG-3.2.2.1: Pause**  
Bei einer Kollision wird das Spiel pausiert.
      - **Z-L-HG-3.2.2.2: Lebensverlust**  
Bei einer Kollision verliert der SEPMAN ein Leben.
      - **Z-L-HG-3.2.2.3: Kantenerhaltung**  
Nach einer Kollision bleiben die bereits begangenen Kanten gleich.
      - **Z-L-HG-3.2.2.4: Rücksetzung**  
Nachdem alle Geister und der SEPMAN manuell auf ihre Startposition zurückgesetzt wurden, wird das Spiel fortgesetzt.
  - **Z-L-HG-3.3: Anzeige**  
Das System hat eine Anzeige.
    - **Z-L-HG-3.3.1: Deaktivierte Roboter**  
Das System zeigt an, wenn ein Roboter deaktiviert ist.
    - **Z-L-HG-3.3.2: Leben**  
Das System zeigt die Leben vom SEPMAN an.
    - **Z-L-HG-3.3.3: Pause**  
Das System zeigt an, wenn das Spiel pausiert ist.
    - **Z-L-HG-3.3.4: Spielende**  
Das System zeigt an, wenn das Spiel beendet ist.
      - **Z-L-HG-3.3.4.1: Sieg**  
Das System zeigt den Sieg an.
      - **Z-L-HG-3.3.4.2: Niederlage**  
Das System zeigt die Niederlage an.
  - **Z-L-HG-3.4: Spielende**  
Das Spiel endet unter einer von 2 Bedingungen.
    - **Z-L-HG-3.4.1 Sieg**  
Der Spieler hat gewonnen, wenn alle Kanten mind. 1 Mal begangen wurden.
    - **Z-L-HG-3.4.2 Niederlage**  
Der Spieler hat verloren, wenn der SEPMAN kein Leben mehr hat.



## Kontextmodell / Spielmodell

Durch das Kontextmodell wird die Umgebung des Systems modelliert. Es wird definiert womit das System interagiert. Alle Elemente des Kontextmodells sollen auch beschrieben werden.

Für Gruppen, die ein Spiel implementieren, tritt an die Stelle des Kontextmodells das Spielmodell. Es enthält alle Elemente des Spiels und beschreibt ihre Interaktion mit dem Spieler. Die Elemente des Spiels sollen außerdem beschrieben werden.

<Bild des Kontextmodells>

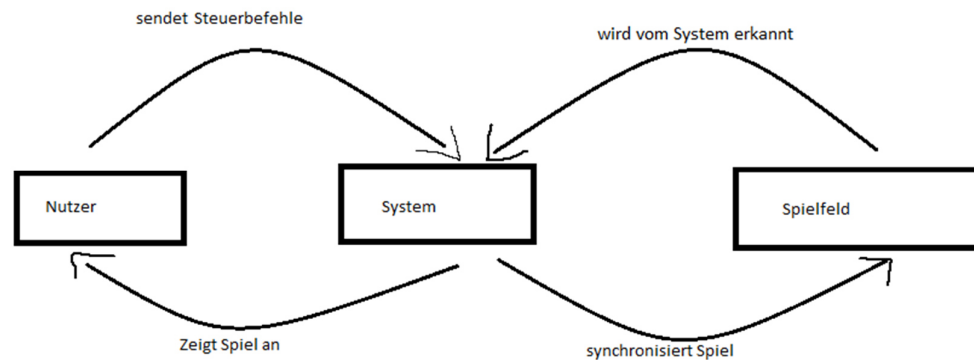
<Akteur 1/Ext. System 1>

Zu jedem Akteur/Ext. System gehört eine kurze Beschreibung. Die Ziele des Akteurs sollten sich in Abschnitt **Fehler! Es wurde kein Textmarkenname vergeben.** wiederfinden und entsprechend verwiesen werden. Zur Beschreibung gehört auch eine textuelle Definition der Daten, die mit dem zu modellierenden System ausgetauscht werden.

<Beschreibung des Akteurs>

<Akteur n/Ext. System n>

<Beschreibung des Akteurs>



### Nutzer:

Der Nutzer interagiert mit dem System, indem er Steuerbefehle sendet.

### System:

Das entwickelte System zeigt dem Nutzer das Spielgeschehen an.

### Spielfeld:

Das Spielfeld wird vom System erkannt.

## Szenarien

Szenarien verbinden die Artefakte miteinander. Jedes Szenario beschreibt eine konkrete Interaktion mit dem zu bauenden System. In diesen Szenarien soll beschrieben werden, wie die Ziele der Akteure durch eine Interaktion mit dem System erfüllt werden.

**<Name Szenario 1>**

*Natürlich sprachliche Dokumentation eines Szenarios mit Referenz auf die beteiligten Akteure/Systeme aus Abschnitt Fehler! Es wurde kein Textmarkenname vergeben. und der erreichten Ziele aus Abschnitt 2.1*

<Beschreibung Szenario 1>

**<Name Szenario n>**

<Beschreibung Szenario n>

**< Szenario 1 – Kollision mit Power-up> (erfüllt 3.2.1.2)**

1. Das System zeigt dem Nutzer an, dass er ein Power-Up eingesammelt hat.
2. Der Nutzer steuert den SEPMAN auf Feld 31, auf dem sich ein Geist befindet.
3. Das System zeigt dem Nutzer eine Kollision an.
4. Das System zeigt dem Nutzer die Deaktivierung des kollidierten Geistes an.
5. Das System zeigt dem Nutzer das Stoppen des Spiels an.
6. Der Nutzer entfernt manuell den Geist.
7. Der Nutzer setzt das Spiel fort.

**<Szenario 2 - Niederlage> (erfüllt 3.3.4.2 und 3.4.2)**

1. Das System zeigt dem Nutzer an, dass er nur noch über ein verbleibendes Leben verfügt.
2. Der Nutzer steuert den SEPMAN von Feld 04 auf Feld 14, auf dem sich ein Geist befindet.
3. Das System zeigt dem Nutzer eine Kollision an.
4. Das System zeigt dem Nutzer das Stoppen des Spiels an.
5. Das System zeigt dem Nutzer den Verlust des letzten verbleibenden Lebens an.
6. Das System zeigt dem Nutzer eine Nachricht an, dass das Spiel verloren wurde.

**<Szenario 3 - Start>** (erfüllt 3.1 und 2.2)

1. Der Nutzer weist das System an, das Spiel zu starten.
2. Das System zeigt dem Nutzer die Positionen der Geister auf ihren jeweiligen Startfeldern, sowie die Startposition des SEPMANs auf Feld 54 an.
3. Der Nutzer gibt dem System den Befehl den SEPMAN ein Feld nach oben zu steuern.
4. Der SEPMAN nimmt die Linie nach oben auf dem Spielfeld wahr.
5. Das System bewegt den SEPMAN auf dem Spielfeld auf Feld 44.
6. Das System zeigt dem Nutzer die Bewegung auf dem virtuellen Spielfeld.
7. Das System zeigt dem Nutzer die Kante zwischen Feld 44 und 54 als bereits besucht an.

**<Szenario 4 - Sieg>** (erfüllt 3.3.4.1 und 3.4.1)

1. Der Nutzer gibt den Steuerungsbefehl die letzte Kante zu befahren.
2. Das System nimmt die Kante auf dem Spielfeld wahr.
3. Das System zeigt dem Nutzer das Befahren der letzten Kante an.
4. Auf dem Spielfeld wird die letzte Kante vom Roboter physisch abgefahren.
5. Das System zeigt dem Nutzer den Sieg an.

**<Szenario 5 - Kollision ohne Power-Up>** (erfüllt 3.2.2)

1. Der Nutzer steuert SEPMAN von Knoten 22 auf Knoten 23 und hat noch 3 Leben.
2. Auf dem Spielfeld wird der SEPMAN auf Feld 23 gesteuert.
3. Das System bewegt, mit dem Spielfeld synchronisiert, einen Geist auf Feld 23.
4. Das System zeigt dem Nutzer eine Kollision eines Geistes mit dem SEPMAN an.
5. Das System zeigt dem Nutzer eine Pause an.
6. Auf dem Spielfeld bleiben alle Roboter auf ihrer aktuellen Position stehen.
7. Das System zeigt an, dass der SEPMAN noch 2 Leben hat.
8. Der Nutzer setzt alle Roboter (Geister und SEPMAN) auf ihre Startpositionen.
9. Der Nutzer setzt manuell das Spiel fort.



8. Der Prozess sendet aktualisierte Positions- und Laufdaten an das virtuelle Spielfeld und an die Eigenschaften der Roboter.

#### Mögliche Laufwege SEP

1. Der Prozess erhält existente angrenzende Kanten vom virtuellen Spielfeld.
2. Der Prozess erhält Eigenschaften des SEPMAN von den Eigenschaften der Roboter.
3. Der Prozess erhält die Sensordaten des SEPMAN vom Spielfeld.
4. Der Prozess wandelt die anschließenden Kanten in mögliche Bewegungsrichtungen.
5. Der Prozess sendet die möglichen Bewegungsrichtungen an den Prozess „Entscheidung Laufweg SEP“.

#### Steuerbefehle in physische Aktionen

1. Der Prozess erhält Steuerbefehle für den SEPMAN von dem Prozess „Entscheidung Laufweg SEP“.
2. Der Prozess erhält Steuerbefehle für einen Geist von dem Prozess „Laufweg Geist bestimmen“.
3. Der Prozess wandelt die Steuerbefehle in physische Aktionen um.
4. Der Prozess bewegt den Roboter den Steuerbefehlen entsprechend.
5. Der Prozess sendet die physischen Aktionen an das Spielfeld.

#### Anzeige des Spielfeldes

1. Der Prozess erhält Eigenschaften des Spielfeldes vom virtuellen Spielfeld.
2. Der Prozess erhält Positionsdaten der Roboter von den Eigenschaften der Roboter.
3. Der Prozess erhält eine eventuelle Kollisionsmeldung.
4. Der Prozess erhält die Lebenszahl des SEPMAN.
5. Der Prozess wandelt diese Daten in eine Anzeige des Spielfeldes um.
6. Der Prozess sendet die Anzeige des Spielfeldes an den Nutzer.

#### Aktionen bei Kollision

1. Der Prozess erhält die Sensordaten einer Kollision vom Spielfeld.
2. Der Prozess erhält die Eigenschaften der Roboter.
3. Der Prozess entscheidet, ob der SEPMAN ein Leben verliert oder der Geist deaktiviert wird.
4. Der Prozess sendet die eventuell aktualisierte Lebenszahl an die Leben des SEP.
5. Der Prozess sendet eine Kollisionsmeldung an die Anzeige des Spielfeldes.

#### Laufweg Geist bestimmen

1. Der Prozess erhält existente angrenzende Kanten vom virtuellen Spielfeld.
2. Der Prozess erhält die Eigenschaften des Geistes von den Eigenschaften der Roboter.
3. Der Prozess erhält die Sensordaten des Geistes vom Spielfeld.
4. Der Prozess entscheidet, in welche Richtung sich der Geist bewegt.
5. Der Prozess wandelt die Richtungsanweisungen in physische Aktionen um.
6. Der Prozess bewegt den Geist.
7. Der Prozess sendet die physischen Aktionen an das Spielfeld.
8. Der Prozess sendet aktualisierte Positions- und Laufdaten an das virtuelle Spielfeld und an die Eigenschaften der Roboter.

## Data Dictionary

*Das Data Dictionary schlüsselt die Datenflüsse des DFD in atomare Datentypen auf. Jeder Datenfluss muss dabei einem eindeutigen atomaren Datentyp zugeordnet werden. Die Anzahl der Ebenen in die ein Datenfluss zerlegt werden kann, variiert je nach Datentyp. Ein Datentyp gilt als atomar, wenn er sich nicht in weitere Datentypen zerlegen lässt und einem fest definierten Wertebereich zuzuordnen ist.*

Richtungseingaben = { [Taste „Hoch“ | Taste „Runter“ | Taste „Links“ | Taste „Rechts“] }

Steuerbefehle/Mögliche Bewegungsrichtung = { [ Bewegung „Hoch“ | Bewegung „Runter“ | Bewegung „Links“ | Bewegung „Rechts“ ] }

Physische Aktion = { [ „Nach oben gehen“ | „Nach unten gehen“ | „Nach links gehen“ | „Nach rechts gehen“ ] } + { ( Kollision ) }

Positionsdaten der Roboter/Aktualisierte Positions- und Laufdaten = { Roboter Nr. } + { Steuerbefehle } + { Reihe + Spalte }

Sensordaten des SEP/Geister = { Steuerbefehle } + { [ „Bewegung möglich“ | „Bewegung nicht möglich“ ] } + { ( Kollision ) }

Eigenschaften des SEP/Geister/Roboter = Aktualisierte Positions- und Laufdaten + [ „Power-Up“ | „Kein Power-Up“ ]

Kollisionsmeldung = Sensordaten einer Kollision + Roboter Nr. + Aktualisierte Lebenszahl

Sensordaten einer Kollision = [ „Oben“ | „Unten“ | „Links“ | „Rechts“ ]

Aktualisierte Lebenszahl/Lebenszahl = [ 0 | 1 | 2 ]

Positionsdaten der Power-Ups = { Reihe + Spalte }

Eigenschaften des Spielfeldes = Positionsdaten der Roboter + Positionsdaten der Power-Ups

Anzeige des Spielfeldes = Eigenschaften des Spielfeldes + Lebenszahl + [ Kollisionsmeldung ]

Reihe/Spalte = [ 1 | 2 | 3 | 4 | 5 | 6 ]

## Message Sequence Charts

*Mit Hilfe von MSC werden Interaktionen zwischen den Elementen des DFDs aus Abschnitt Fehler! Es wurde kein Textmarkenname vergeben. modelliert. Zu jedem Szenario aus Abschnitt Fehler! Es wurde kein Textmarkenname vergeben. wird dazu ein oder mehrere zusammenhängende basic MSC (bMSC) modelliert, dass den Datenaustausch zwischen den Elementen des DFDs zeigt. Durch das hMSC werden die bMSC in einen Zusammenhang gesetzt.*

## **bMSCs**

**bMSC-1:** <Name des bMSC>

<Grafik des bMSC>

**bMSC-n:** <Name des bMSC>

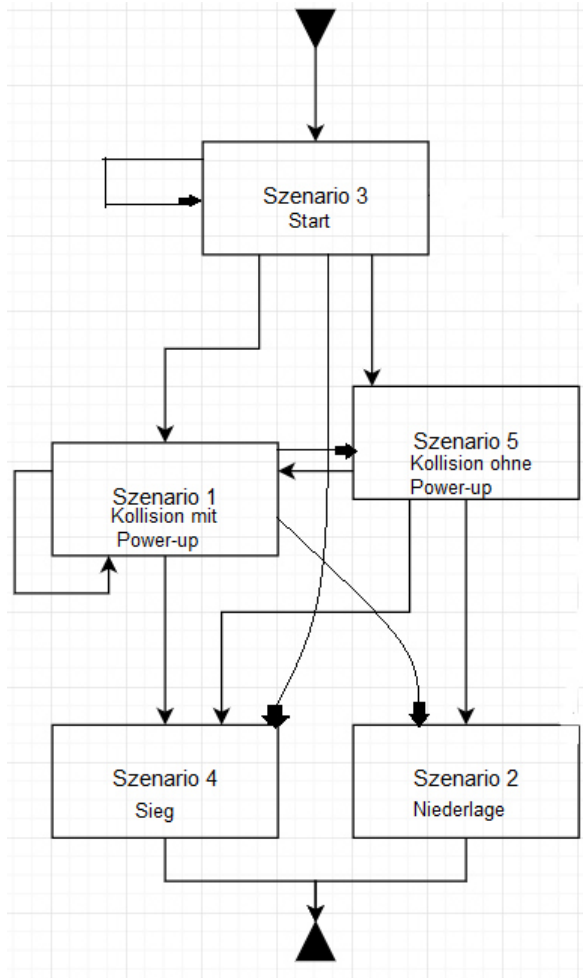
<Grafik des bMSC>

### **Abbildung der Szenarien auf Message Sequence Charts**

*Es muss dokumentiert werden, welche Szenarien in welchen bMSCs (oder in welcher Reihenfolge) umgesetzt wurden.*

<Name Szenario 1>	bMSC-1: <Name des bMSC> bMSC-2: <Name des bMSC> bMSC-1: <Name des bMSC>
...	...
<Name Szenario n>	

**hMSC**





# 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.*

## GUI-Papierprototyp

### Screen „<Name des Screens>“

<Scan des Screen-Papierprototypen>

## Technisches Konzept

<Grafik des technischen Konzepts>

### <Name Komponente 1>

<Beschreibung zu Komponente 1>

### <Name Komponente n>

<Beschreibung zu Komponente n>

## 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.*

<Grafik des Komponentendiagramms>

### Komponentenbeschreibung

<Name Komponente 1>

<Beschreibung zu Komponente 1>

<Name Komponente n>

<Beschreibung zu Komponente n>

### Interfacebeschreibung

<Name Interface 1>

<Beschreibung zu Interface 1>

<Name Interface n>

<Beschreibung zu Interface n>

# Testartefakte

## Modultest

### Testspezifikation

#### Modultestfall 1: <Kurzbezeichnung MTF-1>

Testziel	
Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	
Bestehens Kriterien	

#### Modultestfall n: <Kurzbezeichnung MTF-n>

Testziel	
Schnittstelle/Klasse	
Vorbedingung	
Nachbedingung	
Bestehens Kriterien	

## Testergebnisse

#### Testprotokoll Modultestfall 1 (1. Testdurchführung)

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

#### Testprotokoll Modultestfall 1 (n. Testdurchführung)

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

#### Testprotokoll Modultestfall n (1. Testdurchführung)

Testziel	
Schnittstelle/Klasse	
Vorbedingung	

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

### **Testprotokoll Modultestfall n (n. Testdurchführung)**

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

## **Systemtest**

### **Testspezifikation**

#### **Systemtestfall 1: <Kurzbezeichnung STF-1>**

Szenario	
Schritt	Aktion (User)
1	
2	
3	
4	
...	

#### **Systemtestfall n: <Kurzbezeichnung STF-n>**

Szenario	
Schritt	Aktion (User)
1	
2	
3	
4	
...	

## **Testergebnisse**

### **Testprotokoll Systemtestfall 1 (<1. Testdurchführung>)**

Datum	
Tester	

Version der Software			
Szenario			
Schritt	Aktion (User)	Erwartete Reaktion (System)	Tatsächliche Reaktion (System)
1			
2			
3			
4			
...			
Testurteil			

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

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

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

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

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

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