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.[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 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!!!

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

Inhaltsverzeichnis

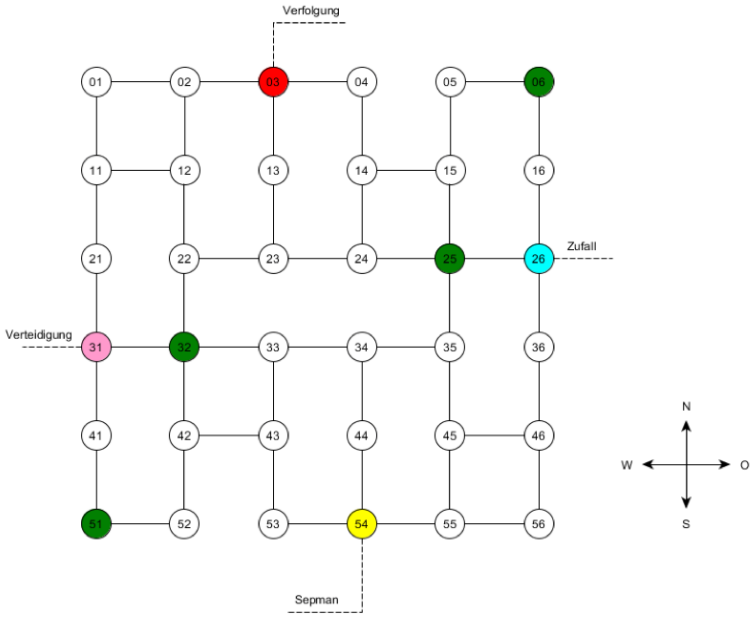
Projektbeschreibung (SEPMAN)

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.

Sepman

„SEPMAN“

Die Aufgabe besteht in der Entwicklung eines Spiels bei dem mehrere Roboter durch ein Labyrinth navigieren. Der Spieler soll hierbei einen der Lego-Mindstorm-Roboter (Sepman) durch ein Labyrinth führen, während er von drei anderen, computergesteuerten Robotern (Geistern) gejagt wird. Zu entwickeln sind zwei identische Roboter die sowohl Geist als auch Sepman sein können. Das variable Labyrinth besteht aus sechs mal sechs Knotenpunkten/Kreuzungen welche teilweise durch Kanten miteinander verbunden sind (siehe Skizze). Die Roboter sollen nur zwischen Knotenpunkten wechseln können die mit einer Kante verbunden sind.



Ein mit den Robotern verbundener Computer soll das Labyrinth, die aktuelle Position aller Roboter im Labyrinth sowie die Position der Powerups (in der Skizze grün) anzeigen. Des Weiteren soll zu erkennen sein welche Kanten des Labyrinths vom Sepman bereits vollständig abgefahren wurden. Über das Programm sollen auch Steuerungsbefehle an die Roboter gesendet werden sowie die Steuerung des Sepman erfolgen.

Die Geister sollen sich autonom durch das Labyrinth bewegen, dabei soll jeder Geist drei unterschiedliche Bewegungs-Modi besitzen. Im Verfolgungsmodus soll der Geist auf direktem Weg zur Position des Sepman fahren. Die Zufalls Einstellung legt fest, dass der Geist immer nur zufällige Bewegungen vollführt. Im Verteidigungsmodus fährt der Geist die im Labyrinth verteilten Powerups ab. Um alle Modi korrekt umzusetzen, dem jeweiligen Roboter seine Rolle zuzuweisen und um Kollisionen zwischen den Geistern zu vermeiden, muss das im Anhang spezifizierte Interface implementiert werden. Die Roboter kommunizieren mit dem Computer ausschließlich über dieses Interface.

Zu Beginn des Spiels werden sowohl die drei Geister als auch der Sepman auf ihre jeweiligen Startpositionen gesetzt (siehe Rollennamen in der Skizze) und bei Spielstart aktiviert. Nach Spielstart versucht der Spieler alle Kanten des Labyrinths mindestens einmal abzufahren. Sein Fortschritt wird auf dem Computer angezeigt. Wird der Sepman bei seinem Versuch von einem Geist berührt, wird das Spiel pausiert und er verliert ein Leben. Alle Roboter werden wieder manuell auf ihre Startpositionen gesetzt, der Lebensverlust wird auf dem Computer visualisiert und die schon abgefahrenen Kanten bleiben erhalten. Schafft der Sepman es alle Kanten zu befahren, bevor er sein letztes Leben verloren hat, so hat er gewonnen. Sind alle drei Leben verbraucht, verliert der Sepman. Das Spielergebnis soll dem Spieler auf dem Computer angezeigt werden. Fährt der Spieler über ein Powerup laufen alle Geister eine bestimmte Zeit lang vor ihm weg. Berührt der Sepman in dieser Zeit einen der Geister, wird dieser für eine bestimmte Zeit aus dem Spiel entfernt. Der Geist startet nach Ablauf der Deaktivierungszeit an seiner Startposition. Während die Geister/ der Sepman aus dem Spiel genommen oder wiedereingesetzt werden, wird das Spiel angehalten, dies wird auf dem Steuerungscomputer angezeigt. Wenn alle Roboter auf den richtigen Positionen stehen kann der Spieler das Spiel fortsetzten und alle Roboter fahren mit ihren Bewegungen fort.

Anmerkungen:

- Das Programm ist in der Programmiersprache Java zu entwickeln.
- Damit der Roboter mit Java programmiert werden kann, soll LeJOS (Lego-Java-Operating-System) für Mindstorms EV3 verwendet werden.
- Wir veröffentlichen besonders gelungene Software auf unserer SEP-Webseite. Hierzu ist es unbedingt erforderlich, dass ein System keine urheberrechtlich geschützten Inhalte (Bilder, Musik, etc.) enthält.

Anhang

Netzwerkinterface
Das zu implementierende Netzwerkinterface wird auf Port 18415 angeboten. Hierbei übernimmt jeder Roboter die Aufgabe eines Servers und bietet eine Socket-Verbindung an. Der Computer implementiert dementsprechend vier Klienten, welche auf je eine Verbindung zugreifen. Um eine Verbindung zu den Robotern aufbauen zu können, muss es auf dem Computer eine Möglichkeit geben die IP-Adressen der Server einzutragen und ihnen die jeweiligen Rollen Geist-Zufällig, Geist-Verteidigung, Geist-Verfolgung und Sepman zuzuordnen. Übermittelt werden ausschließlich Byte-Arrays der Länge 9, diese Arrays werden im Folgenden Nachrichten genannt.

Nachdem alle Verbindungen erfolgreich aufgebaut wurden, soll der Steuercomputer alle 10 Millisekunden die in der Tabelle detaillierte Nachricht an alle verbundenen Roboter senden.

Nachrichteninhalt

Bytearray: N = Position im Bytearray

N	NACHRICHT	BESCHREIBUNG	KODIERUNG
00	Ausgangsknoten Geist 1	Die Nummer des Knotens von dem der Geist/Sepman kommt.	Siehe Skizze
01	Zielknoten Geist 1	Die Nummer des Knotens zu dem der Geist/Sepman geht.	Siehe Skizze
02	Ausgangsknoten Geist 2	analog	analog
03	Zielknoten Geist 2	analog.	analog
04	Ausgangsknoten Geist 3	analog	analog
05	Zielknoten Geist 3	analog	analog
06	Ausgangsknoten Sepman	analog	analog
07	Zielknoten Sepman	analog	analog
08	Spielinformationen		Siehe Tabelle „Spielinformationen“

Spielinformationen

WERT	NACHRICHT	BESCHREIBUNG
-0XX	Start des Powerup-Effekts	Wird gesendet, wenn Sepman über das Powerup auf Knoten XX fährt.
001	Spielstart	Wird gesendet, sobald das Spiel Startet
002	Anfang der Spielpause	Wird gesendet, wenn das Spiel angehalten wird.
003	Ende der Spielpause	Wird gesendet, wenn das Spiel fortgesetzt wird.
004	Sepman Norden	Bewegungsbefehl nach Norden
005	Sepman Osten	Bewegungsbefehl nach Osten
006	Sepman Süden	Bewegungsbefehl nach Süden
007	Sepman Westen	Bewegungsbefehl nach Westen
01X	Geist X wird deaktiviert	Wird gesendet, wenn Sepman ein Powerup hat und Geist X berührt.
02X	Geist X wird aktiviert	Wird gesendet, wenn Geist X reaktiviert wird.
100	Ende des Powerup-Effekts	Wird gesendet, sobald Sepmans Powerup-Effekt endet.
101	Rolle: Geist-Verfolgung	Wird mit der ersten Nachricht gesendet um dem Roboter eine Rolle zuzuweisen.
102	Rolle: Geist-Verteidigung	s.o.
103	Rolle: Geist-Zufall	s.o.
104	Rolle: Sepman	s.o.
127	Spielende	Wird gesendet, sobald das Spiel endet.

Auf diese Nachricht antworten die Roboter wie folgt:

Antwortnachricht

Bytearray: N = Position im Bytearray

N	NACHRICHT	BESCHREIBUNG	KODIERUNG
00	Ausgangsknoten des Roboters	Die Nummer des Knotens von dem der Roboter kommt.	Siehe Skizze
01	Zielknoten des Roboters	Die Nummer des Knotens zu dem der Roboter geht.	Siehe Skizze
02	Sensorstatus	Beschreibt ob einer der Drucksensoren aktiviert wird.	0 – inaktiv 1 – aktiviert
03	Aktivierungsstatus	Beschreibt ob der Roboter aktiviert ist.	0 – inaktiv 1 – aktiviert
04	Powerup-Status	Beschreibt ob der Powerup-Effekt aktiviert ist.	0 – inaktiv 1 – aktiviert
05	Pausenstatus	Beschreibt ob das Spiel pausiert ist.	0 – nicht pausiert 1 – pausiert
06	Unbenutzt		
07	Unbenutzt		
08	Unbenutzt		

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 Spezifkation 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. Sofgoals 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>

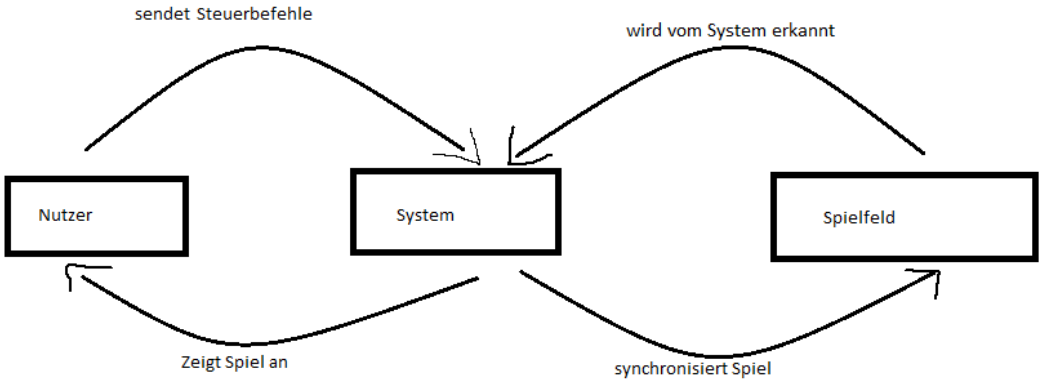
- **Z-1:** Rollenauswahl
  Ein Roboter können sich in einer der 2 Rollen befinden.
  - **Z-L-HG-1.1:** SEPMAN
    Der Roboter kann als SEPMAN fungieren.
    - **Z-L-HG-1.1.1:** Spielersteuerung
      Der Nutzer kann durch Aktionen den SEPMAN steuern.
    - **Z-L-HG-1.1.2:** Leben
      Die Versuche des 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 werde.



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.*

Nutzer:

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

System:

Das entwickelte System zeigt dem Nutzer das Spielgeschehen an. Teil des Systems sind auch Roboter, die als Geist oder SEPMAN über das Spielfeld fahren.

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.*
>
> *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*

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.

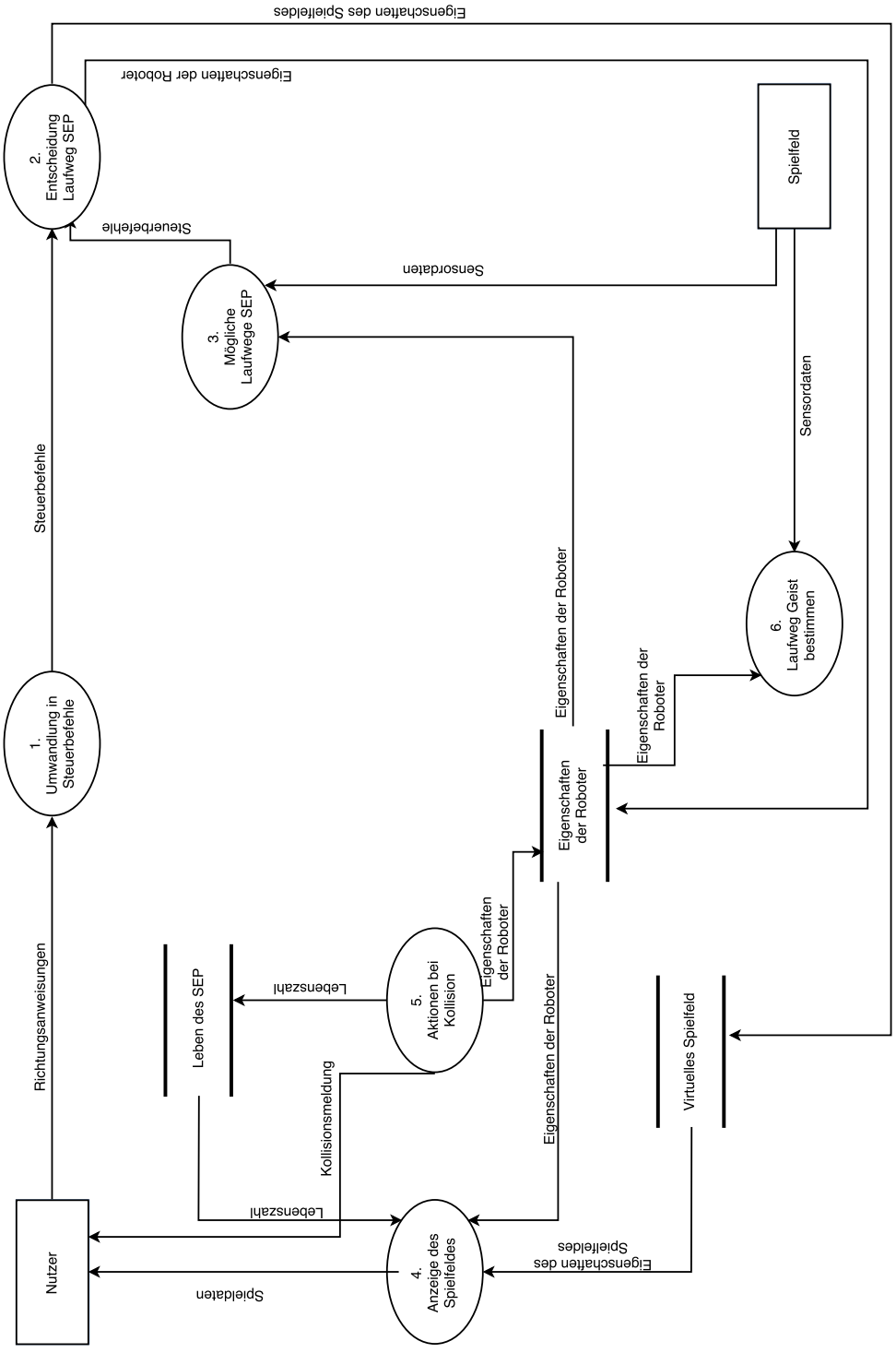
Szenario 5 - Kollision ohne Power-Up (erfüllt 3.2.2)

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.

Logischer Architekturentwurf

Datenflussdiagramm

*Durch ein Datenflussdiagramm werden die Datenflüsse und Verarbeitungsprozesse der Daten innerhalb eines Systems modelliert. Daten kommen von externen Systemen oder Akteuren (Abschnitt **Fehler! Es wurde kein Textmarkenname vergeben.**) in das System und werden verarbeitet.*



Mini Spezifikation

Die Mini Spezifikation gibt Einblick in die Prozesse des DFD. Sie beschreibt wie der Prozess Eingabedaten in die entsprechenden Ausgabedaten transformiert Dabei geht es nicht darum bereits entsprechende Algorithmen zu entwickeln, sondern natürlich sprachlich festzuhalten aus welchen Informationen der Eingabedaten die Ausgabedaten ermittelt werden.

Umwandlung in Steuerbefehle

1. Der Prozess erhält Richtungsanweisungen vom Nutzer.
2. Der Prozess wandelt die Richtungsanweisungen in Steuerbefehle des SEPMAN um.
3. Der Prozess sendet die Steuerbefehle an den Prozess „Entscheidung Laufweg des SEP“.

Entscheidung Laufweg SEP

1. Der Prozess erhält die Steuerbefehle für den SEPMAN.
2. Der Prozess erhält die möglichen Bewegungsrichtungen für den SEPMAN als Steuerbefehle von dem Prozess „Mögliche Laufwege SEP“.
3. Der Prozess erhält die Eigenschaften des Spielfeldes vom virtuellen Spielfeld.
4. Der Prozess entscheidet, ob der gegebene Steuerbefehl mit einer möglichen Bewegungsrichtung übereinstimmt.
5. Der Prozess bewegt den SEPMAN den Steuerbefehlen entsprechend.
6. Der Prozess aktualisiert die gefahrene Kante in den Eigenschaften des Spielfeldes.
7. Der Prozess sendet die Eigenschaften des Spielfeldes an das virtuelle Spielfeld.
8. Der Prozess sendet Eigenschaften der Roboter mit neuen Positionsdaten des SEPMAN an die Eigenschaften der Roboter.

Mögliche Laufwege SEP

1. Der Prozess erhält Eigenschaften der Roboter vom Speicher „Eigenschaften der Roboter“.
2. Der Prozess extrahiert die Positionsdaten des SEPMAN aus den Eigenschaften der Roboter.
3. Der Prozess erhält Sensordaten vom Spielfeld.
4. Der Prozess wandelt die Positionsdaten in mögliche Bewegungsrichtungen.
5. Der Prozess sendet die möglichen Bewegungsrichtungen als Steuerdaten an den Prozess „Entscheidung Laufweg SEP“.

Anzeige des Spielfeldes

1. Der Prozess erhält Eigenschaften des Spielfeldes vom virtuellen Spielfeld.
2. Der Prozess erhält Eigenschaften der Roboter vom Speicher „Eigenschaften der Roboter“.
3. Der Prozess erhält die Lebenszahl des SEPMAN.
4. Der Prozess wandelt diese Daten in Spieldaten um.
5. Der Prozess sendet die Spieldaten an den Nutzer.

Aktionen bei Kollision

1. Der Prozess erhält Eigenschaften der Roboter vom Speicher „Eigenschaften der Roboter“.
2. Der Prozess erhält die Lebenszahl vom Speicher „Leben des SEP“.
3. Der Prozess entscheidet, ob der SEPMAN ein Leben verliert oder der Geist deaktiviert wird.
4. Der Prozess sendet die Lebenszahl an die Leben des SEP.
5. Der Prozess sendet eine Kollisionsmeldung an den Nutzer.

Laufweg Geist bestimmen

1. Der Prozess erhält Eigenschaften der Roboter vom Speicher „Eigenschaften der Roboter“.
2. Der Prozess extrahiert die Eigenschaften des Geistes aus den Eigenschaften der Roboter.
3. Der Prozess erhält Sensordaten vom Spielfeld.
4. Der Prozess wandelt die Daten in Steuerungsbefehle um.
5. Der Prozess bewegt den Geist den Steuerbefehlen entsprechend.
6. Der Prozess aktualisiert die Positionsdaten des Geistes in den Eigenschaften der Roboter.
7. Der Prozess sendet die Eigenschaften der Roboter an den Speicher „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.

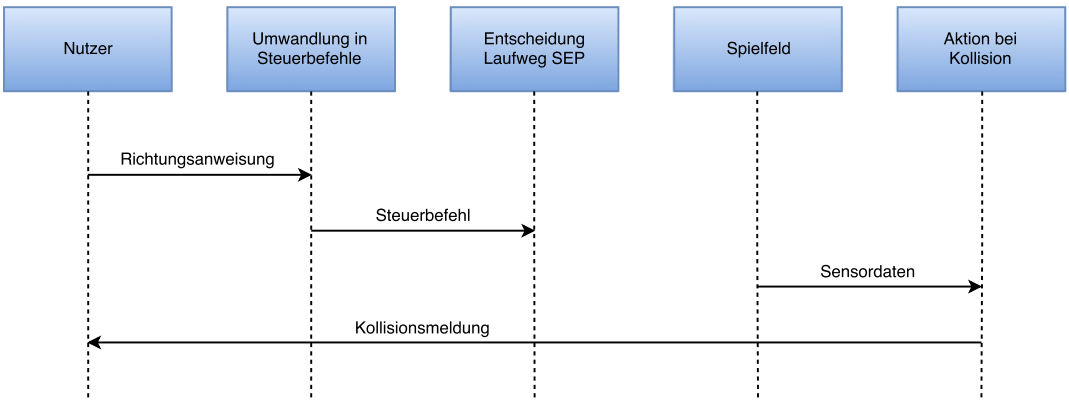
Eigenschaften der Roboter	= Eigenschaften des SEPMAN + 3{ Eigenschaften des Geistes }3
Eigenschaften des Geistes	= Positionsdaten + Geistmodus
Eigenschaften des SEPMAN	= Positionsdaten + [„Power-Up“ | „Kein Power-Up“]
Eigenschaften des Spielfeldes	= Positionsdaten der Power-Ups + { Gefahrene Kante }
Gefahrene Kante	= { Kantennummer }
Geistmodus	= [„Verfolgung“ | „Zufall“ | „Verteidigung“ | „Flucht“]
Kollisionsmeldung	= *Anleitung zum Zurücksetzen der Roboter*
Lebenszahl	= [0 | 1 | 2 | 3]
Positionsdaten	= Zeile + Spalte
Positionsdaten der Power-Ups	= { Zeile + Spalte }
Richtungsanweisungen	= { [Himmelsrichtung „Norden“ | Himmelsrichtung „Süden“ | Himmelsrichtung „Westen“ | Himmelsrichtung „Osten“] }
Sensordaten	= { Ziffer }
Spalte:	= [1 | 2 | 3 | 4 | 5 | 6]
Spieldaten	= Eigenschaften des Spielfeldes + Lebenszahl + Positionsdaten
Steuerbefehle	= { [Bewegung „Vor“ | Bewegung „Zurück“ | Bewegung „Links“ | Bewegung „Rechts“] }
Virtuelles Spielfeld	= Positionsdaten der Power-Ups + { Gefahrene Kante }
Zeile	= [0 | 1 | 2 | 3 | 4 | 5]

Message Sequence Charts

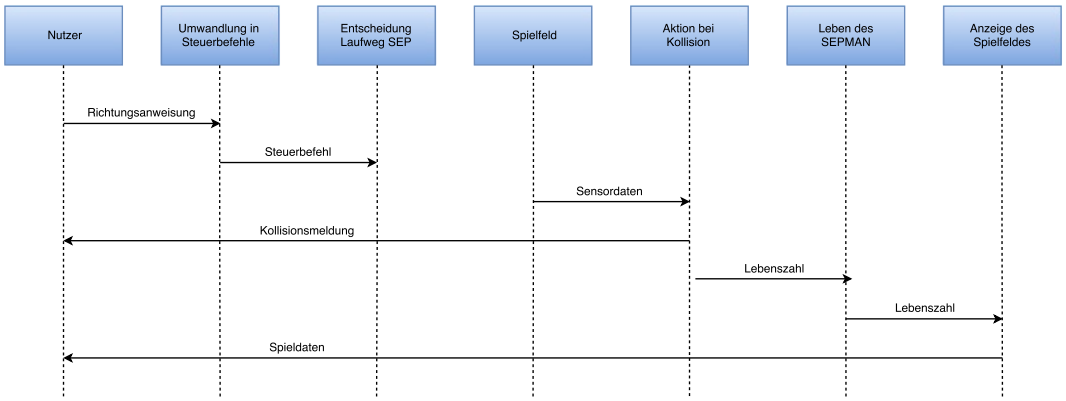
Mit Hilfe von MSC werden Interaktionen zwischen den Elementen des DFDs aus Abschnitt ***Fehler! Es wurde kein Textmarkenname vergeben.*** *modelliert. Zu jedem Szenario aus Abschnitt* ***Fehler! Es wurde kein Textmarkenname vergeben.*** *wird dazu ein oder mehrere zusammenhängende basic MSC (bMSC) modelliert,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

bMSC

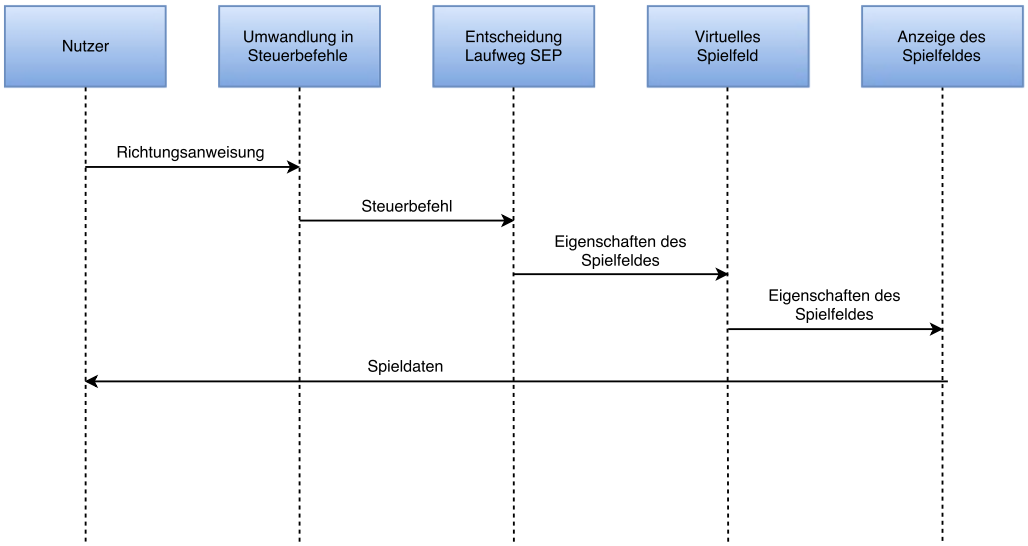
bMSC zu Szenario 1:



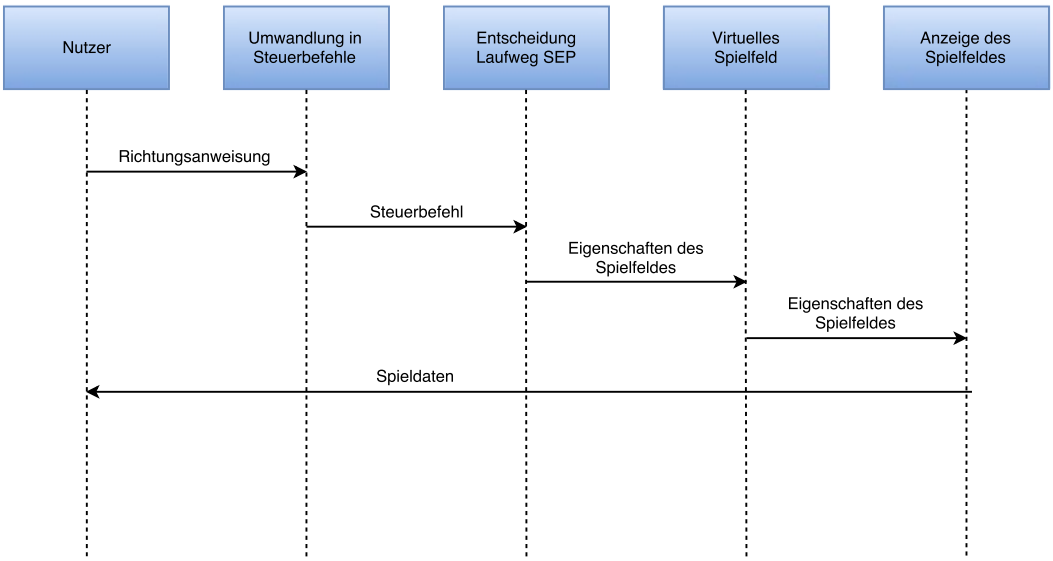
bMSC zu Szenario 2:



bMSC zu Szenario 3:



bMSC zu Szenario 4:



bMSC zu Szenario 5:

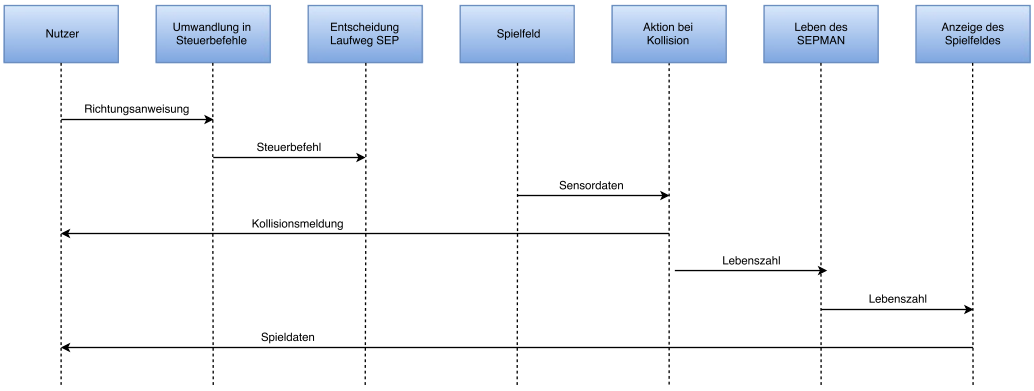
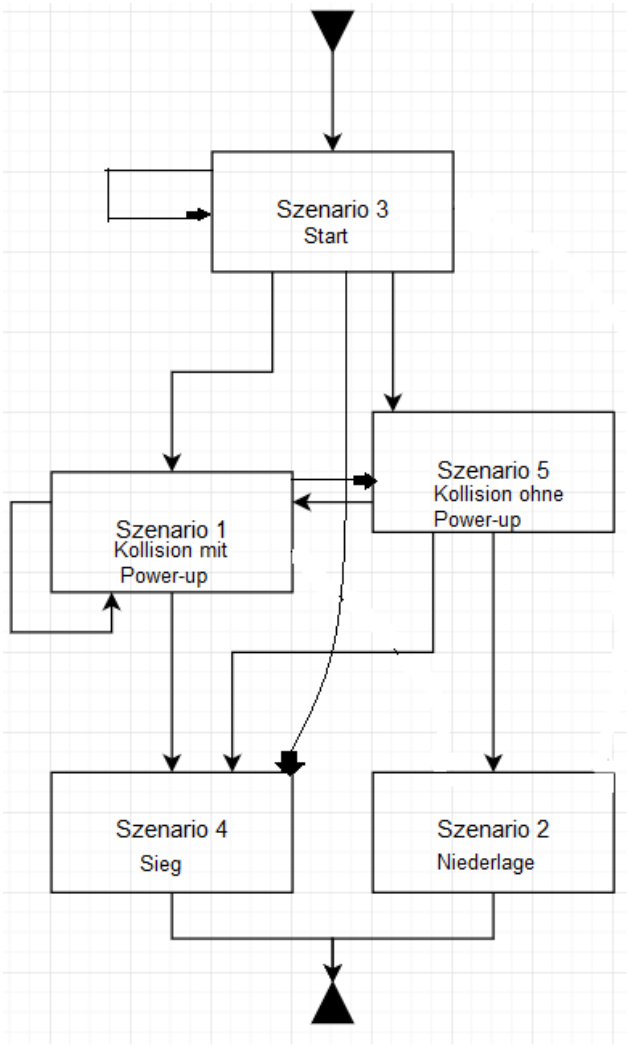


Abbildung der Szenarien auf Message Sequence Charts

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

Szenario 1: Kollision mit Power-Up	bMSC-1
Szenario 2: Niederlage	bMSC-2
Szenario 3: Start	bMSC-3
Szenario 4: Sieg	bMSC-4
Szenario 5: Kollision ohne Power-Up	bMSC-5

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			