Software-Engineering

Alexander Heider, 43167
Constantin Eisinger, 43309
Lutz Bader, 42887

Gruppe 4

Sommersemester 2015
11. November 2015

Inhaltsverzeichnis

1 Analyse

1.1 Use Cases

Im folgenden werden die Use Cases genauer erläutert.

1.1.1 Beschreibung

Beschreiben Sie jeden Use-Case (mindestens 5) mit eigenen Worten.

1.1.1.1 Use Case 1 - Spiel starten

Name Spiel starten

Boundary Spielbrett (Anwendung)

Akteure Spieler (2-4)

Prio 1 (essentiell)

Beschreibung Damit zwei oder mehr Spieler eine Partie spielen können, muss ein Spiel gestartet werden können. Hierzu gehört auch die Auswahl eines Spielers, der den ersten Spielzug machen wird.

Vorbedingungen Das Spielbrett (die Anwendung) muss gestartet sein und es darf kein aktives Spiel laufen. → Wir befinden uns im Hauptmenü.

1.1.1.2 Use Case 2 - Spielzug

Name Spielzug

Boundary Spielbrett (Anwendung)

Akteure Spieler

Prio 1 (essentiell)

Beschreibung Ein Spielzug besteht aus mehreren Aktionen. Der Spieler, der den Spielzug ausführt, würfelt den Würfel und führt seinen Zug entsprechend der Augenzahl aus. Beispielsweise bringt der Spieler einen Wissensstreiter aus seinem Heimatfeld auf seinen Startwert bei einer Augenzahl von 6.

Vorbedingungen Ein Spiel muss zuvor gestartet werden.

1.1.1.3 Use Case 3 - Fragerunde

Name Fragerunde

Boundary Spielbrett (Anwendung)

Akteure Spieler

Prio 2 (wichtig)

Beschreibung Der ausführende Spieler stellt dem geschlagenen Spieler eine Frage aus Kategorie 1-4.

Vorbedingungen Im Spielzug trifft der Wissensstreiter eines Spielers auf ein Feld, welches von einem Wissensstreiter eines anderen Spielers besetzt ist.

1.1.1.4 Use Case 4 - Spiel beenden

Name Spiel beenden

Boundary Spielbrett (Anwendung)

Akteure Spieler

Prio 1 (essentiell)

Beschreibung Sollten die Spieler keine Lust mehr haben oder alle Spielzüge gezogen worden sein, so muss das Spiel beendet werden.

Vorbedingungen Um ein Spiel zu beenden, muss es gestartet worden sein.

1.1.1.5 Use Case 5 - Bestenliste

Name Bestenliste

Boundary Spielbrett (Anwendung)

Akteure Spieler

Prio 3 (unwichtig)

Beschreibung Ein zusätzliches Feature des Spiels ist eine Bestenliste. Diese wird am Ende eines Spiels angezeigt.

Vorbedingungen Damit sich ein Spieler in die Bestenliste eintragen kann, muss ein Spieler der Gewinner einer Partie sein und eine Punktzahl erreicht haben um in die Top 10 der Bestenliste kommen zu können.

1.1.1.6 Use Case 6 - Spiel speichern

Name Spiel speichern

Boundary Spielbrett (Anwendung)

Akteure Spieler

Prio 3 (unwichtig)

Beschreibung Ein zusätzliches Feature des Spiels ist die Möglichkeit, den aktuellen Spielstand zu sichern.

Vorbedingungen Das Spiel muss zum Speichern gestartet worden sein und ein Spielzug durchgeführt worden sein.

1.1.1.7 Use Case 7 - Spiel laden

Name Spiel laden

Boundary Spielbrett (Anwendung)

Akteure Spieler

Prio 3 (unwichtig)

Beschreibung Als Erweiterung des Use Case 6 wird ein Feature zum Laden eines bereits gespeicherten Spielstands angeboten.

Vorbedingungen Damit ein gespeichertes Spiel geladen werden kann, muss ein zuvor gespieltes Spiel gespeichert worden sein.

1.1.2 Prioritäten

Prio	Name	Begründung
1	Spiel starten	Um ein Spiel spielen zu können, muss man als erstes ein Spiel starten können.
2	Spielzug	Nachdem das Spiel gestartet wurde, kann man nur weiterspielen, falls man einen Spielzug durchgeführt wurde.
3	Spiel beenden	Wenn alle möglichen Spielzüge durchgeführt wurden, kann das Spiel beende werden.
4	Fragerunde	Die Fragerunde ist Anfangs nicht essentiell, dennoch aber wichtig um das Spiel interessanter zu machen.
5	Bestenliste	Dies ist ein Feature, welches nicht fester Bestandteil des Spiels ist.
6	Spiel speichern	Ein weiteres Feature, welches nicht essentiell für den normalen Spielablauf ist.
7	Spiel laden	Dieses Feature ist nur realisierbar, falls bereits Spiele gespeichert wurden.

1.1.3 Use Case Diagramm

Skizzieren Sie das Use-Case-Diagramm mit allen Akteuren und Abhängigkeiten.
Die Abbildung 1.1 zeigt das Use Case Diagramm für das Lern Quiz.

1.1.4 Umfang erste Iteration

Bestimmen Sie den Umfang der ersten Iteration (3 Use-Cases).
Die wichtigsten 3 Use Cases sind: „Spiel starten“, „Spielzug“ und „Spiel beenden“. Dies sind die 3 Use Cases, welche zum Spielablauf entscheidend sind.

1.2 Use-Cases, Details, Objektmodell und Schnittstellen der ersten Iteration

1.2.1 Activity Diagramme

Erstellen Sie für die Use Cases Beschreibungen in Form von Activity-Diagrammen.
Diese Acitivity Diagramme werden im folgenden dargestellt:

- Spiel starten, Abbildung 1.2
- Startspieler bestimmen, Abbildung 1.3
- Spiel beenden, Abbildung 1.4
- Spielzug, Abbildung 1.5

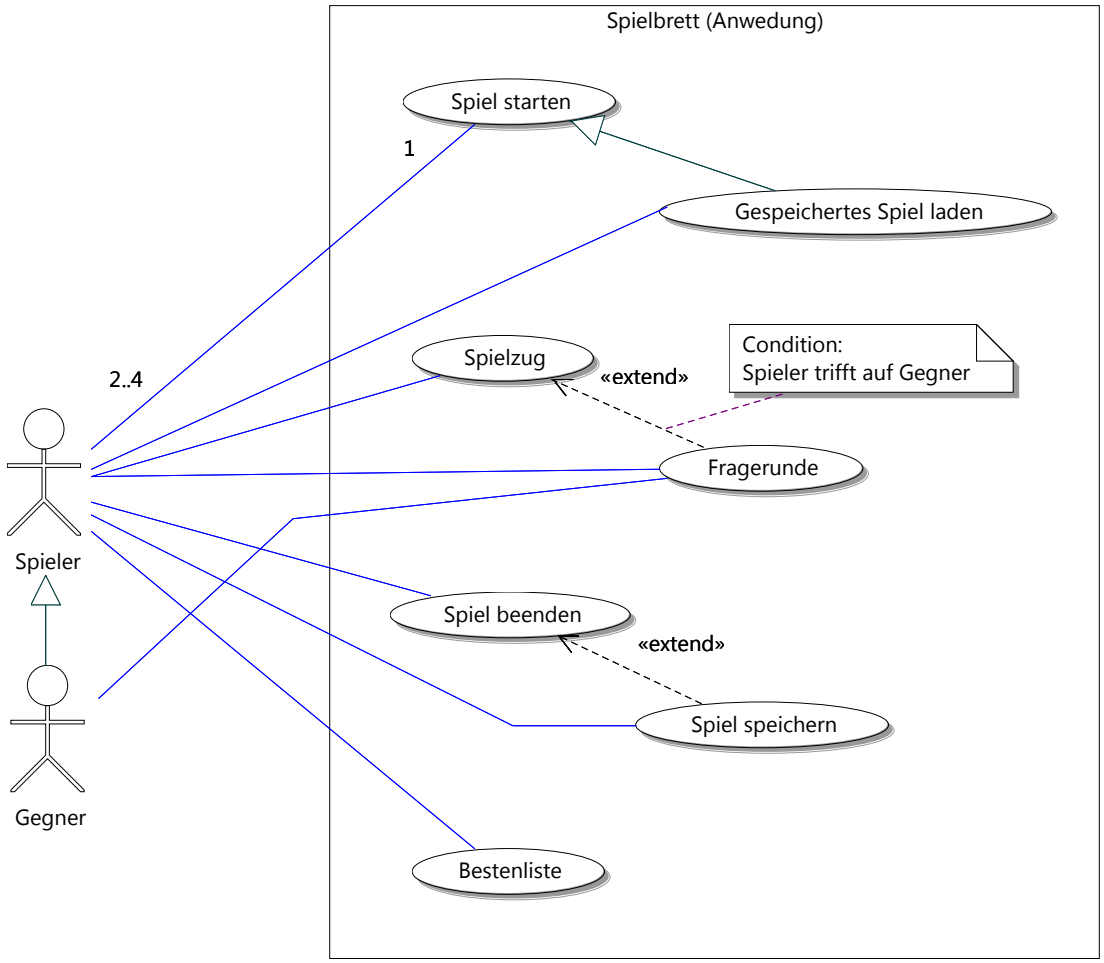


Abbildung 1.1: Use Case Diagramm

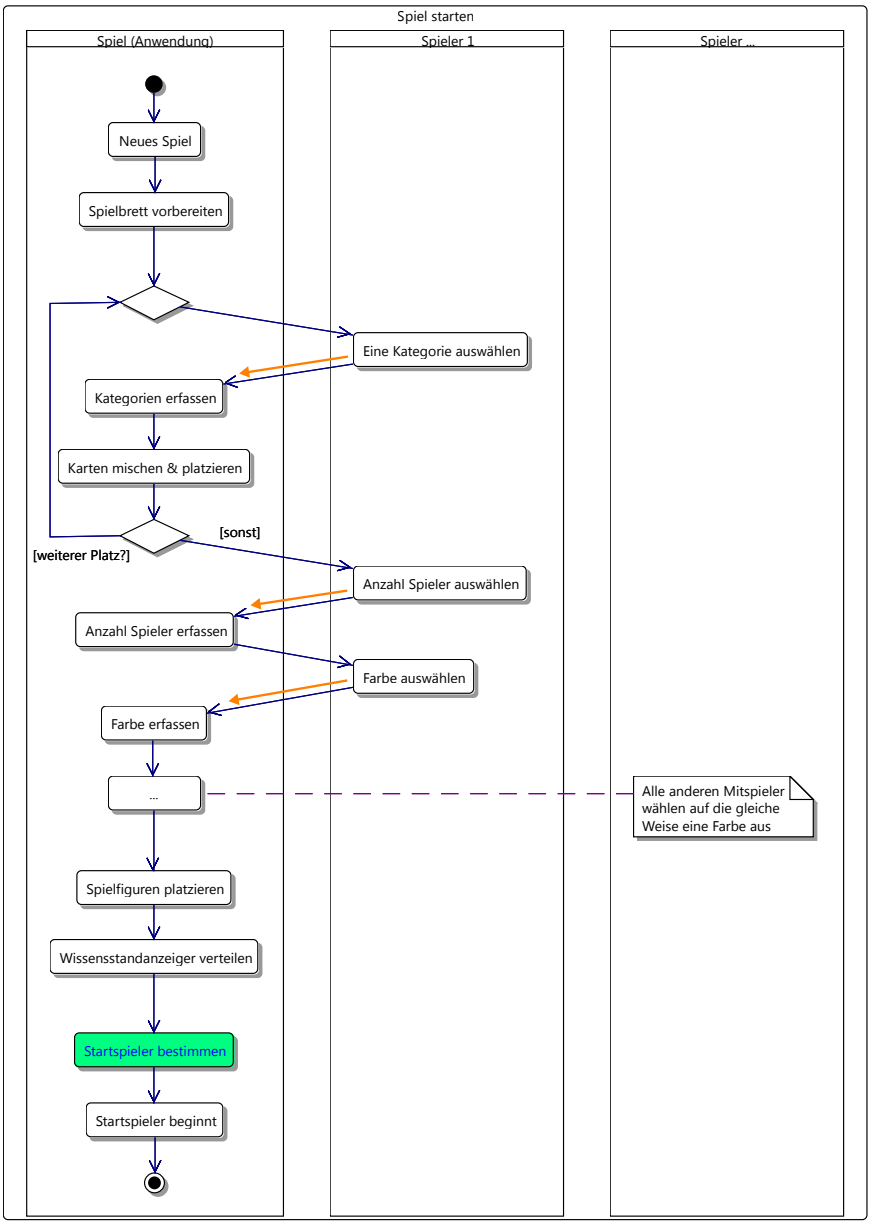


Abbildung 1.2: Spiel starten

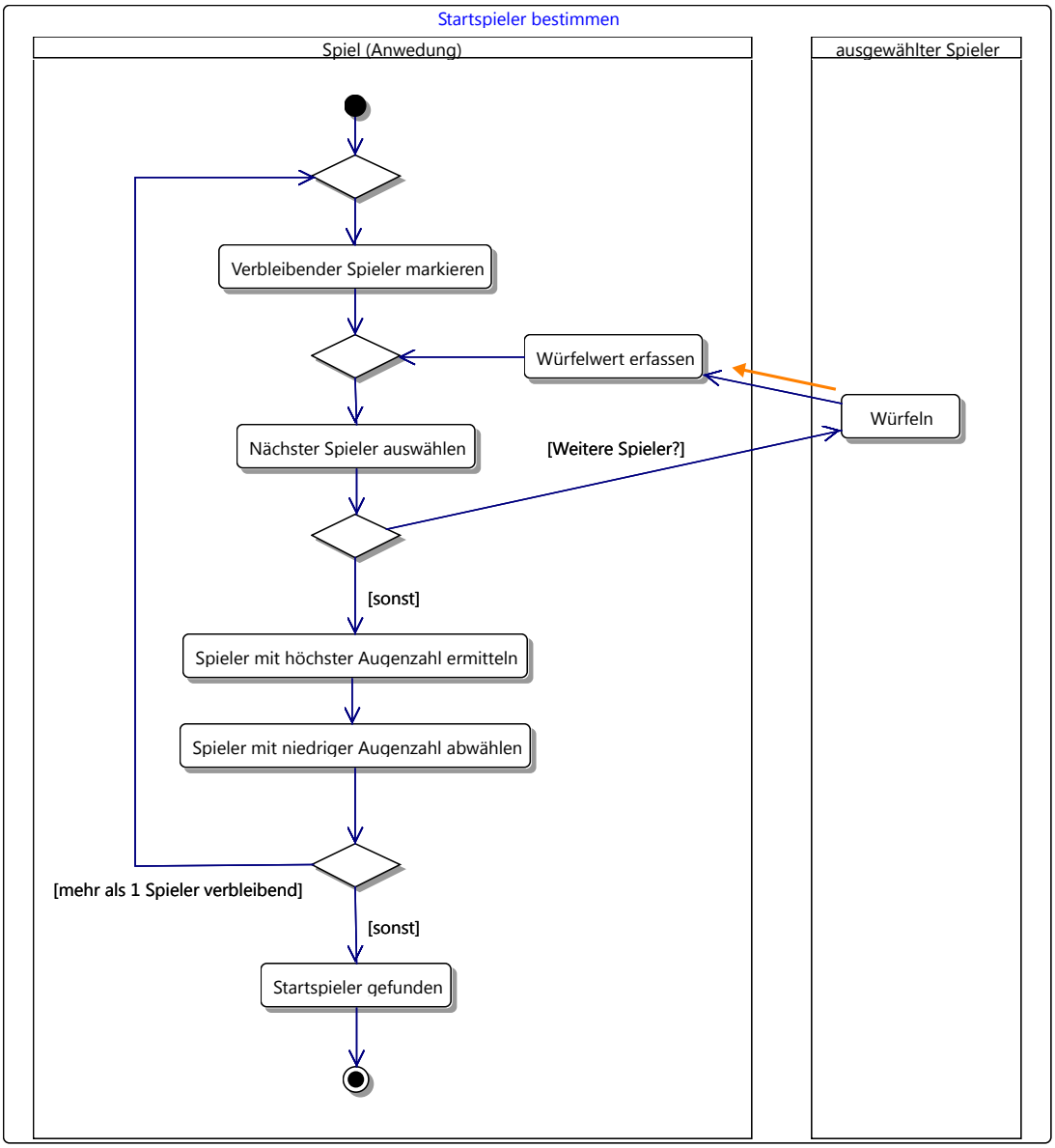


Abbildung 1.3: Startspieler bestimmen

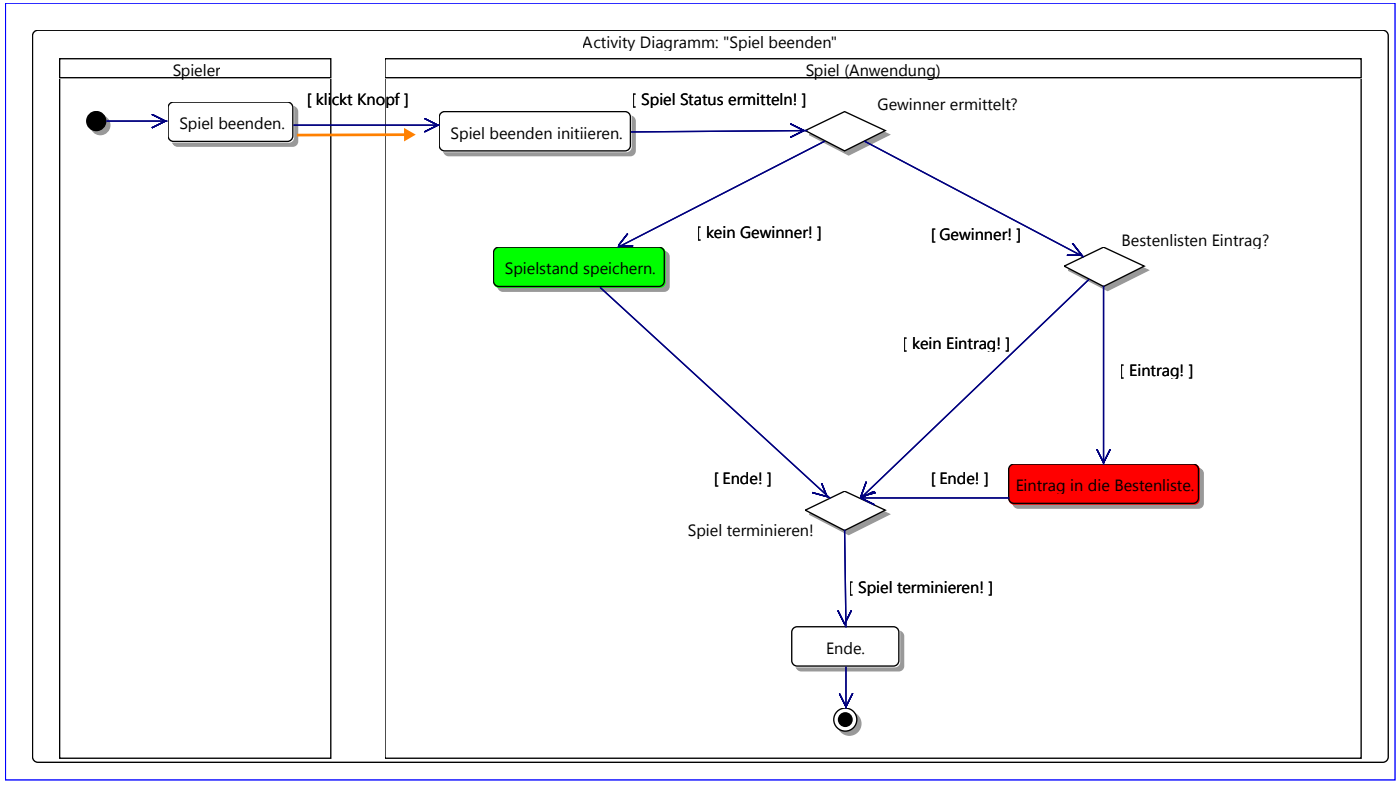


Abbildung 1.4: Spiel beenden

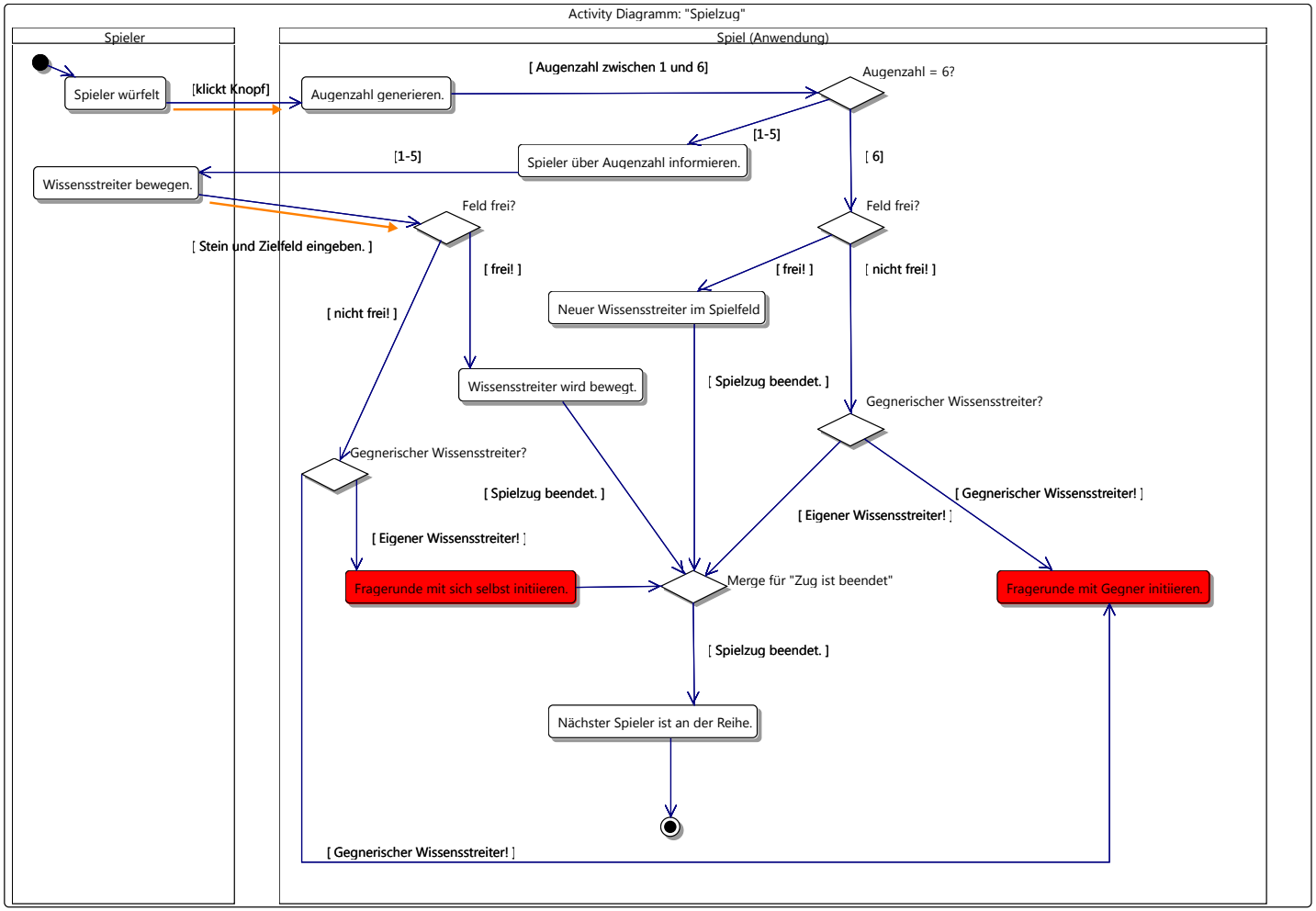


Abbildung 1.5: Spielzug

1.2.2 Klassendiagramm

Extrahieren Sie aus den erstellten Diagrammen die Konzepte des Lern-Quiz-Computer-Spiels und ihre Beziehungen. Stellen Sie diese in Form eines Klassendiagramms dar (Objektmodell).
Die Abbildung 1.6 zeigt das Klassendiagramm des Spiels.

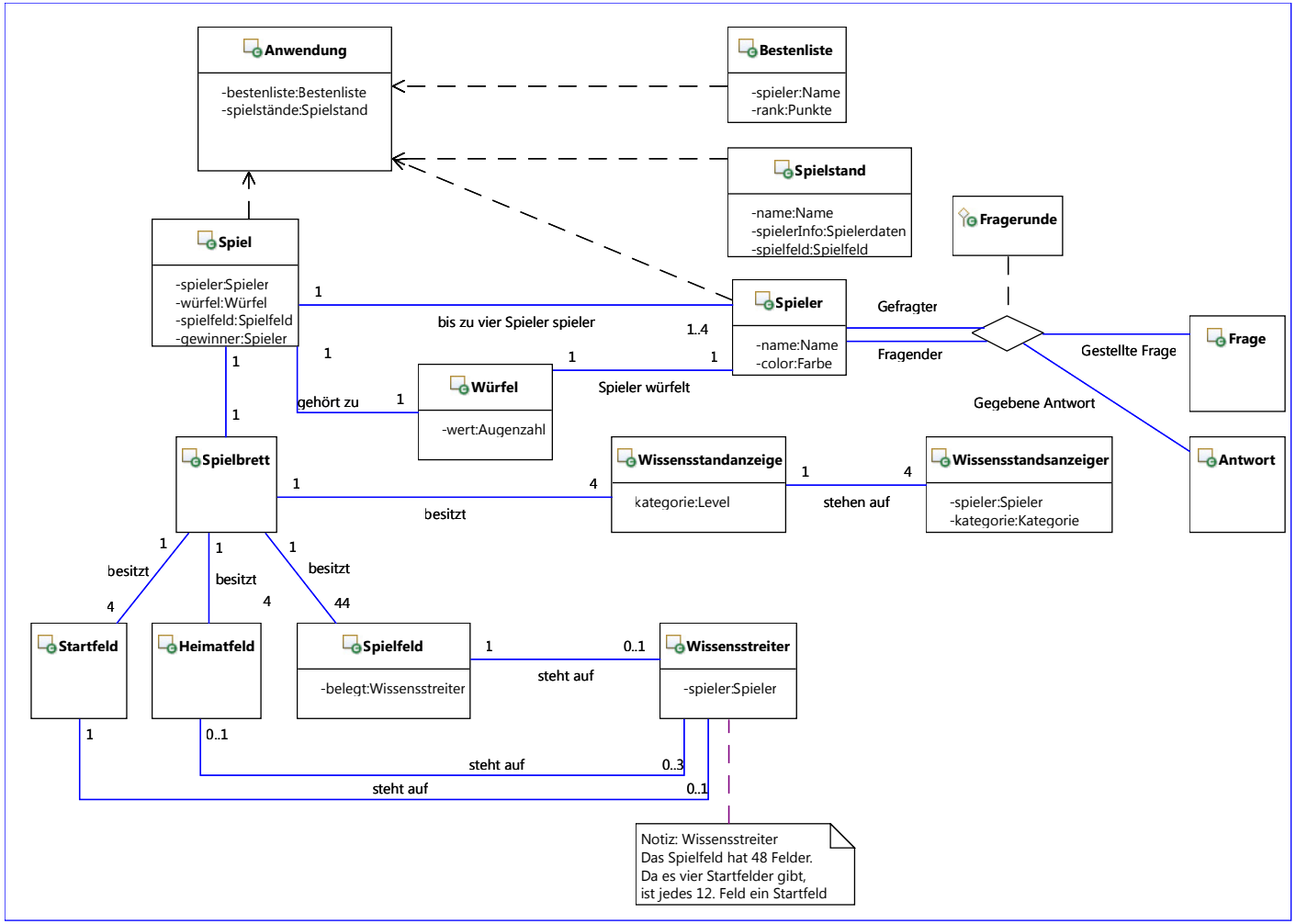
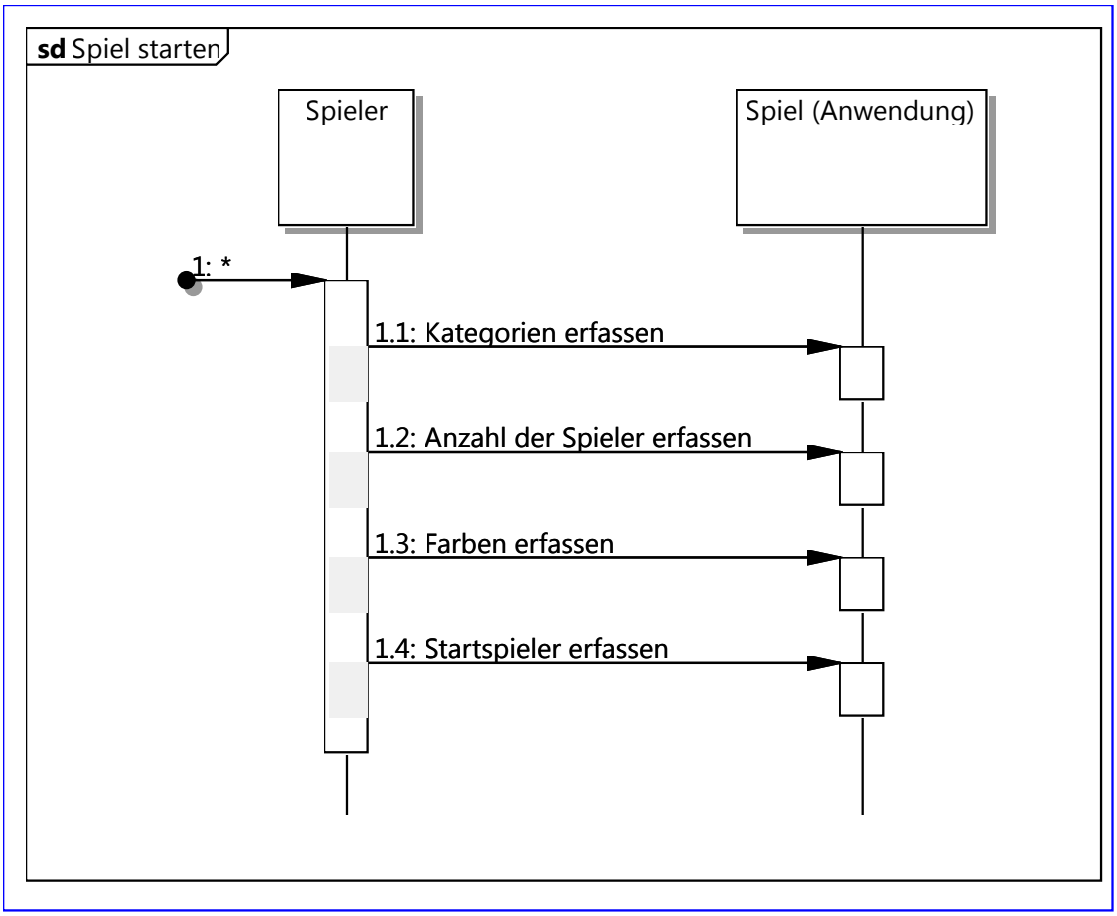


Abbildung 1.6: Klassendiagramm

1.2.3 Systemoperationen

Bestimmen Sie aus den Activity-Diagrammen die möglichen Systemoperationen. Er stellen Sie zur besseren Übersicht System-Sequenz-Diagramme und beschreiben Sie jede Operation mit eigenen Worten.

1.2.3.1 Systemoperationen im Use Case „Spiel starten“



1.2.3.1.1 Kategorie erfassen

Name Kategorien erfassen

Verantwortlichkeit Ein Spieler wählt die Kategorien für die Fragerunden aus.

Referenzen Use Case „Spiel starten“

Bemerkungen Der Einfachheit halber, wählt Spieler 1 die Kategorien aus.

Ausnahmen Keine

Output Die vier Kategorien für die Fragerunden stehen fest.

Vorbedingungen

- Ein neues Spiel muss gestartet worden sein.
- Das Spielbrett muss vorbereitet worden sein.

Nachbedingungen

- Die Kategorien wurden im System dem aktuellen Spiel zugeordnet.

1.2.3.1.2 Anzahl der Spieler erfassen

Name Anzahl der Spieler erfassen

Verantwortlichkeit Ein Spieler wählt die Anzahl der am Spiel teilnehmenden Spieler aus.

Referenzen Use Case „Spiel starten“

Bemerkungen Es müssen mindestens zwei und maximal vier Spieler an einem Spiel teilnehmen.

Ausnahmen Keine

Output Die Anzahl der aktiven Spieler wurde ausgewählt.

Vorbedingungen

- Alle Vor- und Nachbedingungen der Systemoperation „Kategorien auswählen“.
- Die Kategorien müssen ausgewählt worden sein.
- Die Karten müssen gemischt und auf dem Spielbrett platziert worden sein.

Nachbedingungen

- Die Anzahl der aktiven Spieler wurde vom System übernommen.

1.2.3.1.3 Farben erfassen

Name Farben erfassen

Verantwortlichkeit Ein Spieler wählt seine Farbe aus.

Referenzen Use Case „Spiel starten“

Bemerkungen Dieser Vorgang wird für jeden teilnehmenden Spieler wiederholt.

Ausnahmen Keine

Output Alle Spieler haben eine Farbe ausgewählt.

Vorbedingungen

- Alle Vor- und Nachbedingungen der Systemoperation „Anzahl der Spieler erfassen“.

Nachbedingungen

- Die ausgewählten Farben müssen vom System erfasst und den jeweiligen Spielern zugeordnet werden.

1.2.3.1.4 Startspieler erfassen

Name Startspieler erfassen

Verantwortlichkeit Die Spieler würfeln um zu entscheiden, welcher Spieler mit dem ersten Spielzug beginnt.

Referenzen Use Case „Spiel starten“

Bemerkungen Der Startspieler wird durch Würfeln bestimmt.

Ausnahmen Keine

Output Der Startspieler steht fest.

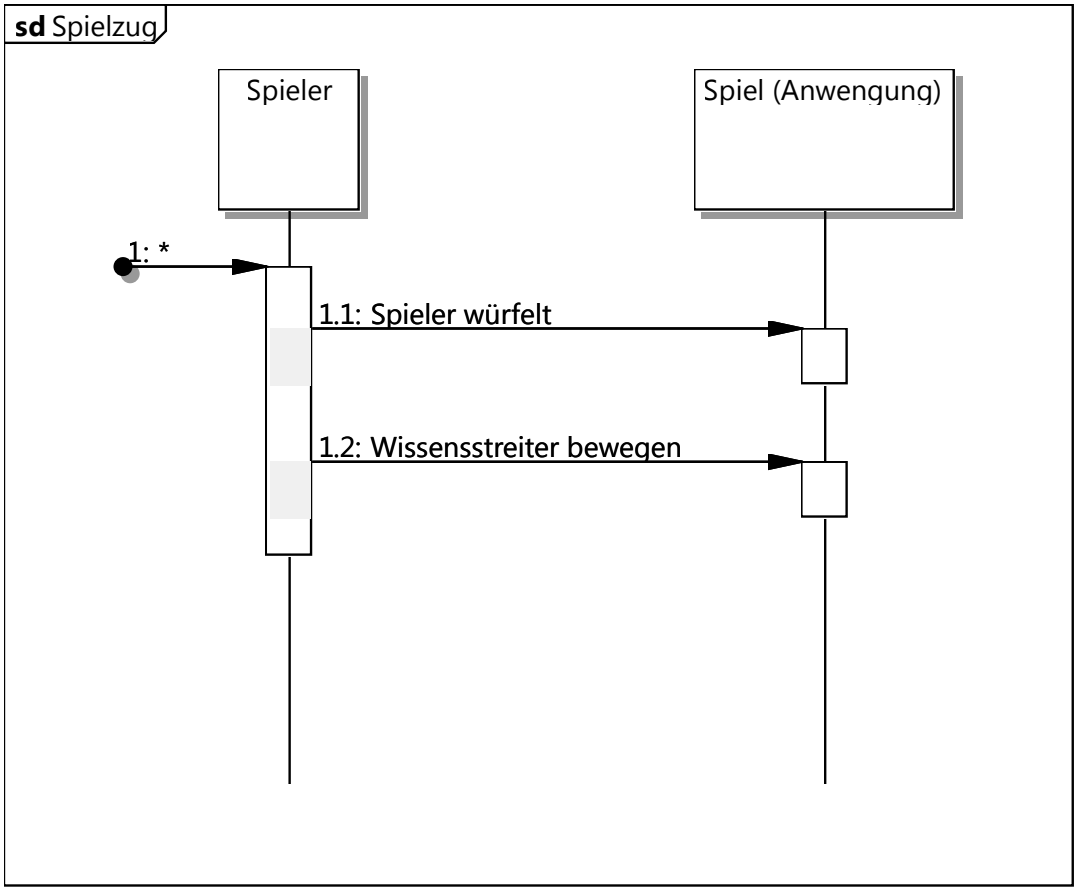
Vorbedingungen

- Alle Vor- und Nachbedingungen der Systemoperation „Farben erfassen“.
- Alle Spieler wurden als „verbleibende“ mögliche Startspieler erfasst.

Nachbedingungen

- Der letzte als möglicher Startspieler markierte Spieler wird vom System als Startspieler erfasst.

1.2.3.2 Systemoperationen im Use Case „Spielzug“



1.2.3.2.1 Spieler würfelt

Name Spieler würfelt

Verantwortlichkeit Der Spieler der zurzeit an der Reihe ist, beginnt seinen Spielzug, indem er würfelt.

Referenzen Use Case „Spielzug“

Bemerkungen Keine

Ausnahmen Keine

Output Es steht fest, ob der Spieler einen neuen Wissensstreiter auf Spielbrett bringen kann, oder ob er einen bereits auf dem Spielbrett stehenden Wissensstreiter vorwärts bewegen kann.

Vorbedingungen

- Das Spiel wurde gestartet.

Nachbedingungen

- Die Augenzahl wurde im System erfasst.

1.2.3.2.2 Wissensstreiter bewegen

Name Wissensstreiter bewegen

Verantwortlichkeit Der zur Zeit aktive Spieler bewegt seinen Wissensstreiter auf dem Spielbrett um n Felder. (n ≈ Augenzahl des Würfels)

Referenzen Use Case „Spielzug“

Bemerkungen Der Spieler kann bei einer Augenzahl kleiner 6 jeden seiner auf dem Spielbrett befindlichen Wissensstreiter auswählen und um n Felder bewegen. (n ≈ Augenzahl des Würfels)

Ausnahmen Falls der Wissensstreiter auf einem Spielfeld landet, welches bereits von einem anderen eigenen Wissensstreiter belegt ist, ist der Spielzug nicht möglich. Wenn kein Wissensstreiter auf dem Feld ist, darf der Spieler erneut würfeln.

Output Der Wissensstreiter wurde um n Felder bewegt. (n ≈ Augenzahl des Würfels)

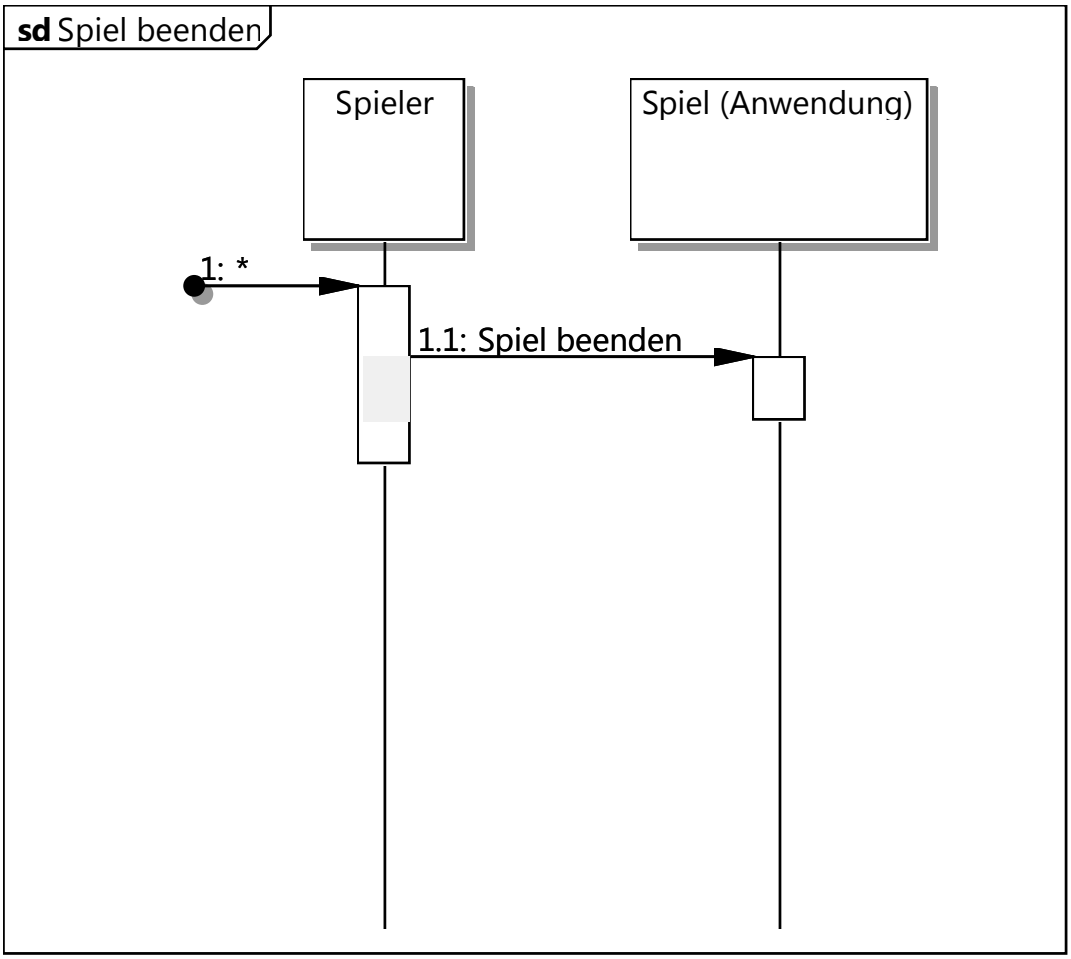
Vorbedingungen

- Alle Vor- und Nachbedingungen der Systemoperationen „Spieler würfelt“
- Die Augenzahl ist kleiner 6.

Nachbedingungen

- Der bewegte Wissensstreiter wurde vom System erfasst.
- Weitere Schritte wurden vom System eingeleitet.

1.2.3.3 Systemoperationen im Use Case „Spiel beenden“



1.2.3.3.1 Spiel beenden

Name Spiel beenden

Verantwortlichkeit Ein Spieler beendet das Spiel (die Anwendung)

Referenzen Use Case „Spiel beenden“

Bemerkungen Der aktive Spieler will das Spiel über das Menü beenden.

Ausnahmen Keine

Output Die Terminierung des Spiels wird eingeleitet.

Vorbedingungen

- Ein Spiel wurde gestartet

Nachbedingungen

- Das System erfasst den aktuellen Spielstand aus.
- Das System leitet Schritte zur Terminierung des Spiels ein.

2 Design

2.1 Aufbereitung der Analyse und Systemoperationen

In diesem Teil wird die Realisierung des Use Cases `Spielzug durchführen` dokumentiert.

- Entwerfen Sie für diese Use Cases ein Bedienkonzept in Form von Mockups.
- Beschreiben Sie die zu realisierenden Use Cases in Form von System-Use-Cases.
  - Use-Case-Beschreibung
  - Use-Case-Diagram
  - detaillierte Beschreibung in Form von Activity-Diagrammen

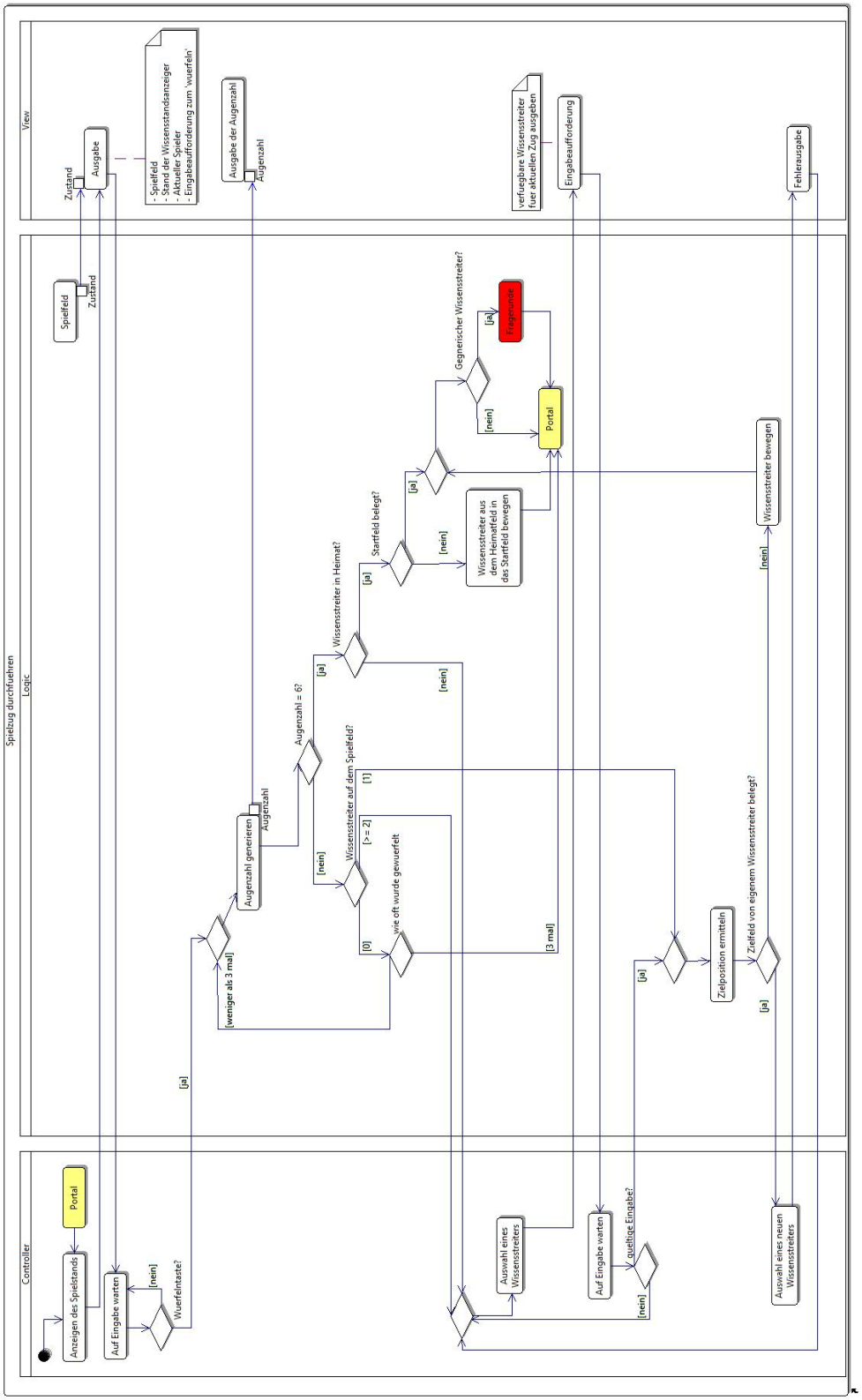
  Orientieren Sie sich hierzu an Ihren Analyseergebnissen und berücksichtigen Sie die gewählte Architektur und Ihr Bedienkonzept.
- Bestimmen Sie die Schnittstelle der Applikationsschicht, wie sie für die Realisierung gebraucht wird.
  - Skizzieren Sie die System-Operationen in Form von System-Sequenz-Diagrammen.
  - Beschreiben Sie die System-Operationen.
  - Achten Sie auf Parameter, Ausnahmen, Vor- und Nachbedingungen.
  - Wählen Sie geeignete Use-Case-Controller (oder entsprechende Klassen aus dem Analyse-Modell) und ordnen Sie diesen die System-Operationen zu. Erstellen Sie entsprechende Sub-Packages, Klassen und Interfaces im “application“-Package.

2.1.1 Bedienkonzept in Form von Mockups

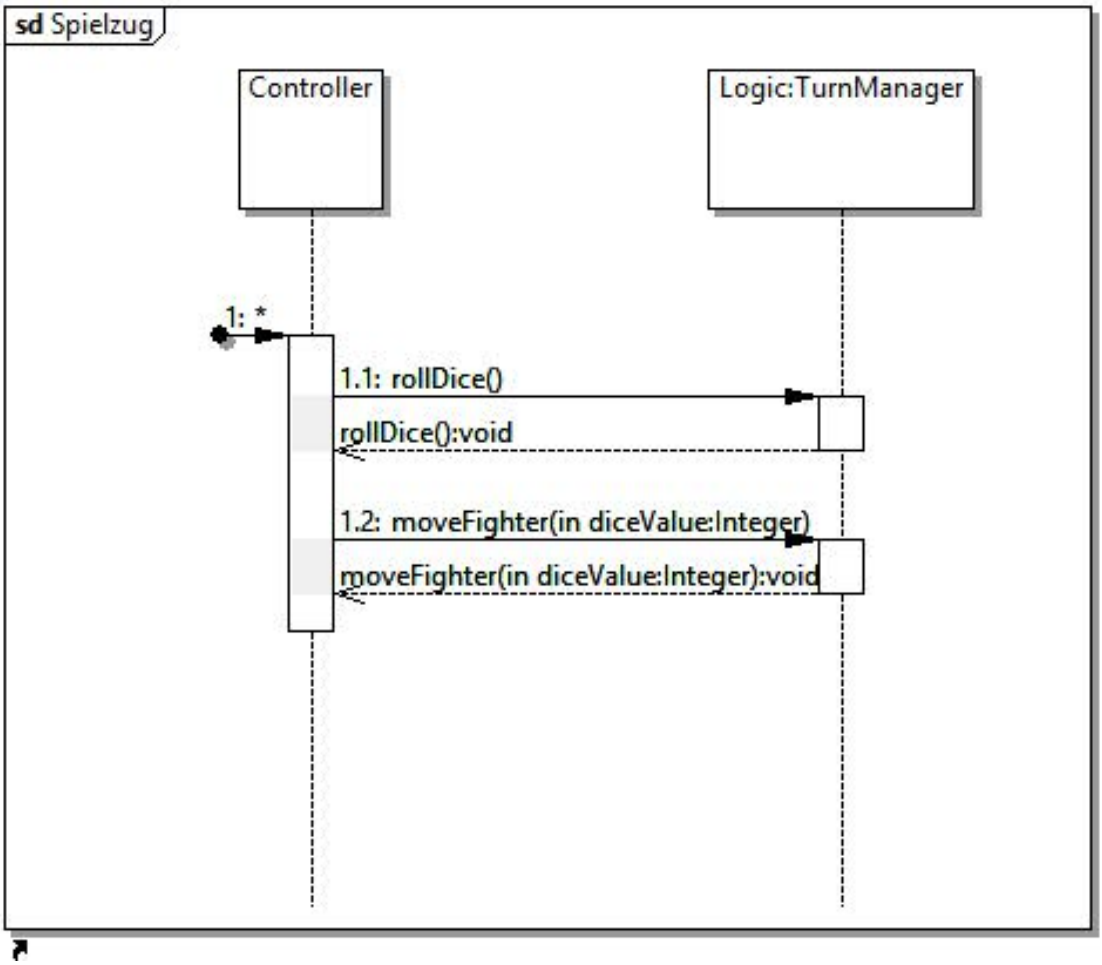
2.1.1.1 Use-Case: Spielzug durchführen

Console Output:

2.1.1.2 Detaillierte Beschreibung in Form eines Activity-Diagram



2.1.2 Systemsequenz Diagramm



2.1.3 System-Operationen im Use-Case: Spielzug durchführen

2.1.3.1 rollDice()

Name `rollDice()`

Verantwortlichkeit Folgende Schritte werden von der Logik durchgeführt:

- Eine Augenzahl (1-6) wird generiert
- Mögliche Züge werden ermittelt
  - Wissensstreiter bewegen
  - Gegenerischen Wissensstreiter schlagen → Fragerunde einleiten

Referenzen Use Case „Spielzug durchführen“

Bemerkungen Der Würfel wird gewürfelt.

Ausnahmen Keine

Vorbedingungen Spiel muss initialisiert worden sein und ein Spieler muss am Zug sein.

Nachbedingungen Je nach Augenzahl wurde entweder

- ein Wissensstreiter auf das Startfeld bewegt,
- ein Wissensstreiter bewegt,
- ein gegnerischer Wissensstreiter geschlagen → Fragerunde wird eingeleitet
- oder ein gültiger Zug ist nicht möglich. (An dieser Stelle wird der Zug beendet und der nächste Spieler ist an der Reihe.)

2.1.3.2 moveFighter(int diceValue)

Name `moveFighter(int diceValue)`

Verantwortlichkeit Der Wissensstreiter wird bewegt und es wird überprüft, ob das Zielfeld leer oder belegt war.

Referenzen Use Case „Spielzug durchführen“

Bemerkungen Der Wissensstreiter wird bewegt

Ausnahmen Keine

Vorbedingungen Eine Augenzahl (1-6) muss generiert worden sein.

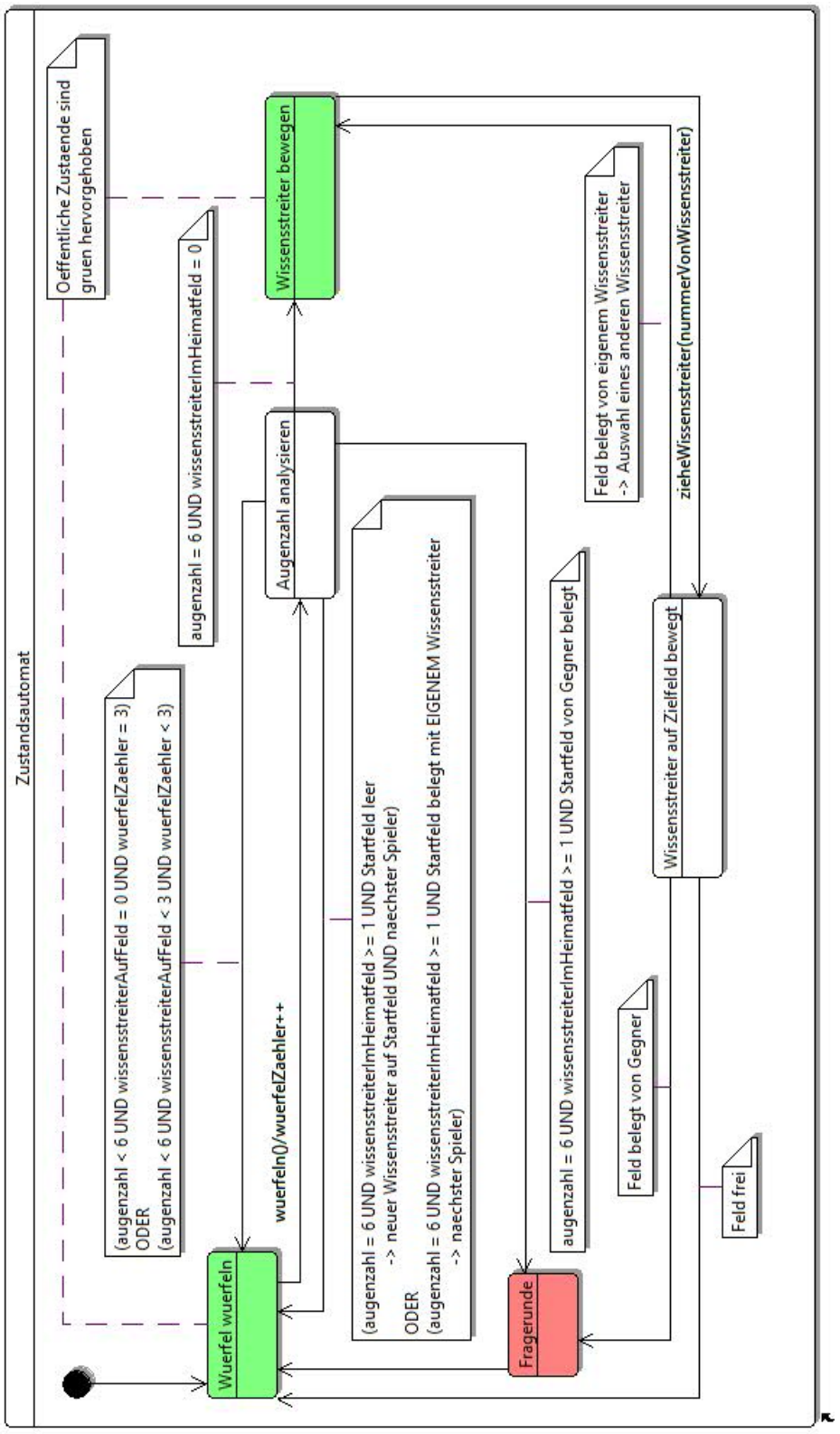
Nachbedingungen Falls das Zielfeld leer war, wird der Spielzug beendet. Falls das Zielfeld belegt war, wird eine Fragerunde eingeleitet.

2.2 Zustandsautomat

Erstellen Sie einen Automaten und ordnen Sie jeder System-Operation mindestens einen Zustand zu, in dem sie ein gültiges äußeres Event darstellt. Beachten Sie insbesondere die Vor- und Nachbedingungen, sowie die gefundenen Ausnahmen aus dem vorherigen Designschritt. Erstellen Sie geeignete Klassen zur Repräsentation Ihrer Zustände.

2.2.1 Anmerkung

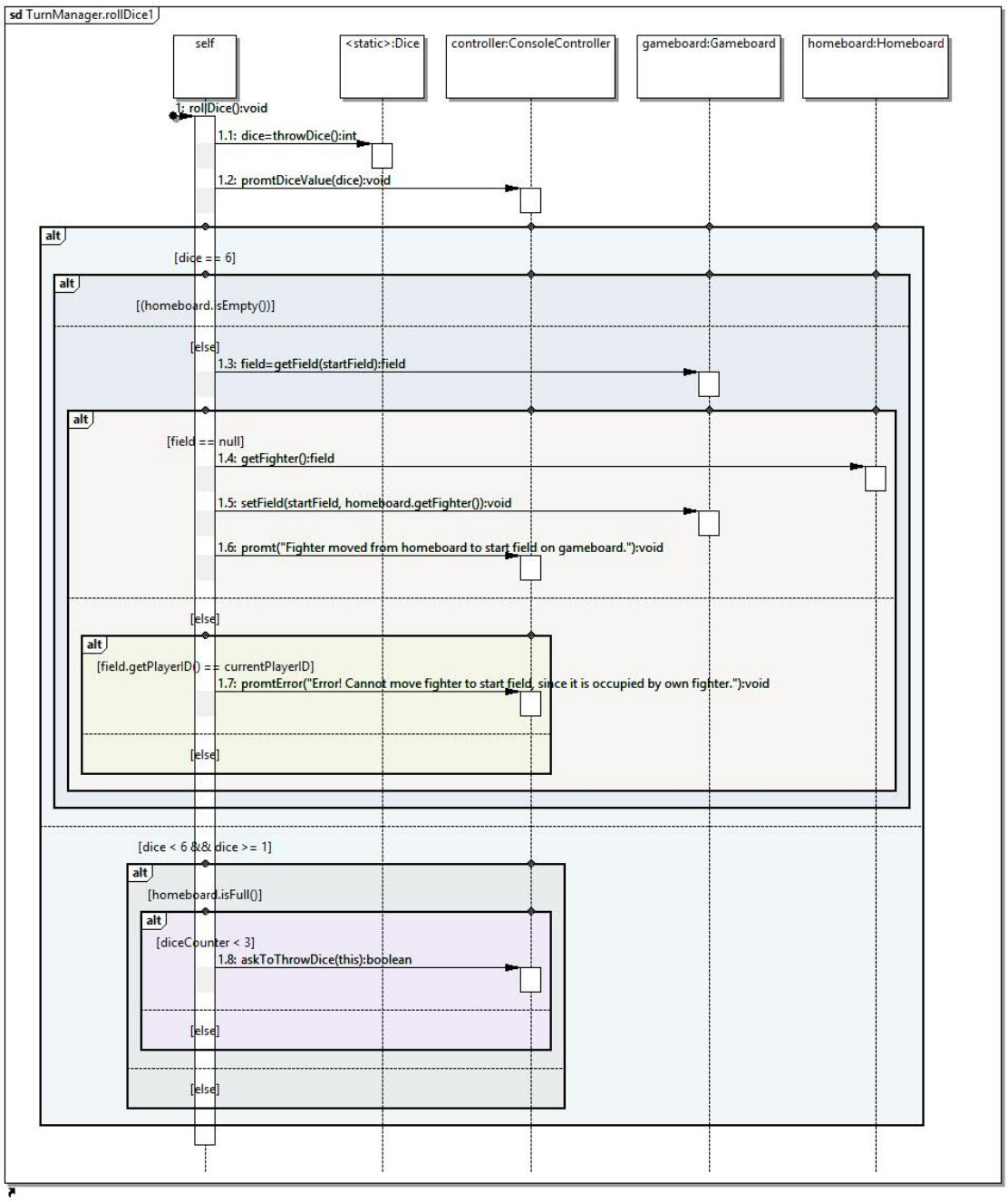
- grün hervorgehobene Zustände symbolisieren die System-Operationen
- weiß hervorgehobene Zustände symbolisieren private Zustände.
- rot hervorgehobene Zustände sind Use-Cases, welche in einem Zustand vereint zur Hilfe dargestellt werden.


Zustandsautomat
Oeffentliche Zustaende sind gruen hervorgehoben
(augenzahl < 6 UND wissensstreiterAufFeld = 0 UND wuerfelZaehler = 3)
ODER
(augenzahl < 6 UND wissensstreiterAufFeld < 3 UND wuerfelZaehler < 3)
augenzahl = 6 UND wissensstreiterImHeimatfeld = 0
Wissensstreiter bewegen
Augenzahl analysieren
Wuerfel wuerfeln
wuerfeln()/wuerfelZaehler++
(augenzahl = 6 UND wissensstreiterImHeimatfeld >= 1 UND Startfeld leer
-> neuer Wissensstreiter auf Startfeld UND naechster Spieler)
ODER
(augenzahl = 6 UND wissensstreiterImHeimatfeld >= 1 UND Startfeld belegt mit EIGENEM Wissensstreiter
-> naechster Spieler)
Feld belegt von eigenem Wissensstreiter
-> Auswahl eines anderen Wissensstreiter
zieheWissensstreiter(nummerVonWissensstreiter)
Fragerunde
augenzahl = 6 UND wissensstreiterImHeimatfeld >= 1 UND Startfeld von Gegner belegt
Wissensstreiter auf Zielfeld bewegt
Feld belegt von Gegner
Feld frei

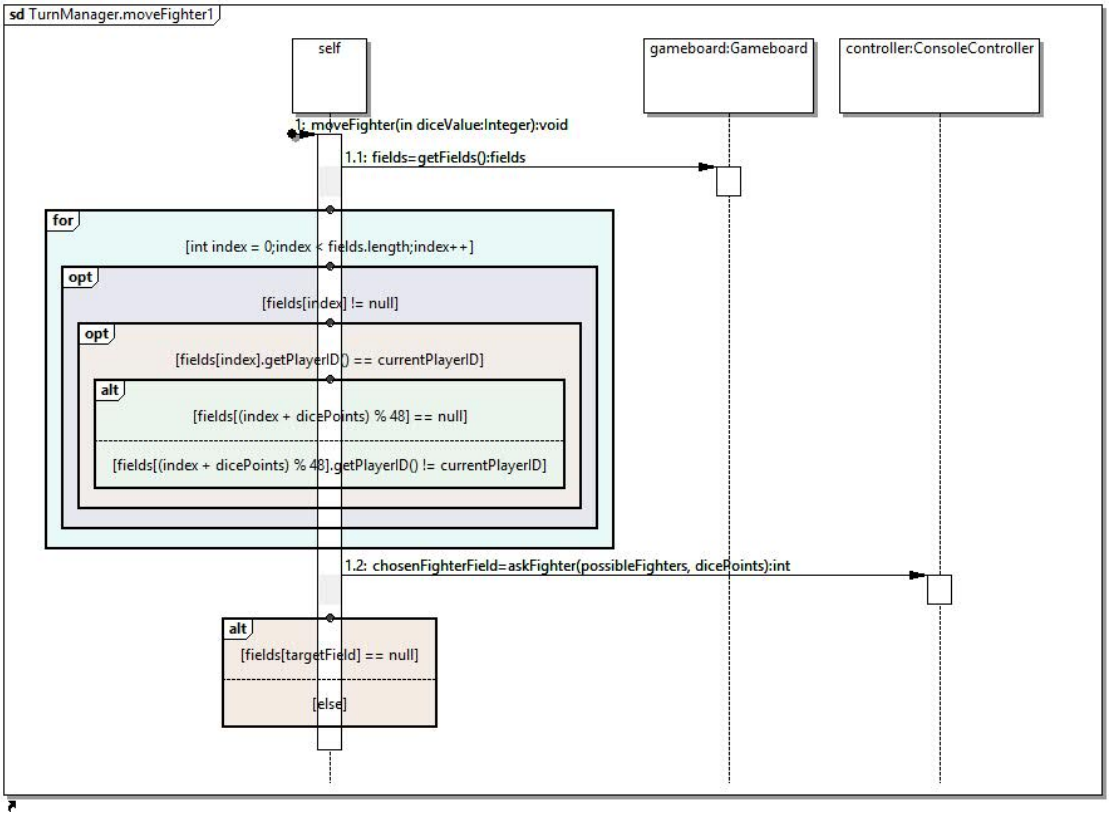
2.3 Detailed Design und das Objektmodell

- Designen Sie die zu realisierenden System-Operationen und die Initialisierung des Systems. Erstellen Sie hierzu für jede System-Operation und Initialisierungsroutine ein detailliertes Sequenz- oder Kommunikationsdiagramm. Beachten Sie hierbei die Vor- und Nachbedingungen der System-Operationen (die Zustände).
- Erstellen Sie das Design-Objektmodell (nur Klassen und ihre Beziehungen, keine Texte). Fügen Sie eine Klasse immer dann dem Objektmodell hinzu, wenn beim Entwickeln der Sequenzdiagramme der Bedarf besteht. Verbinden Sie Klassen über Assoziationen, Aggregationen oder Kompositionen immer dann, wenn dies für die Kommunikation erforderlich ist.
- **Implementieren Sie die entworfenen Operationen sofort.**
- Sorgen Sie dafür, dass alle zu visualisierenden Ergebnisse in entsprechenden Attributen abgelegt werden und stellen Sie sicher, dass über entsprechnende Get-Operationen auf diese Attribute zugegriffen werden kann. Erweitern Sie die bereits bestehenden Interfaces nach Bedarf.

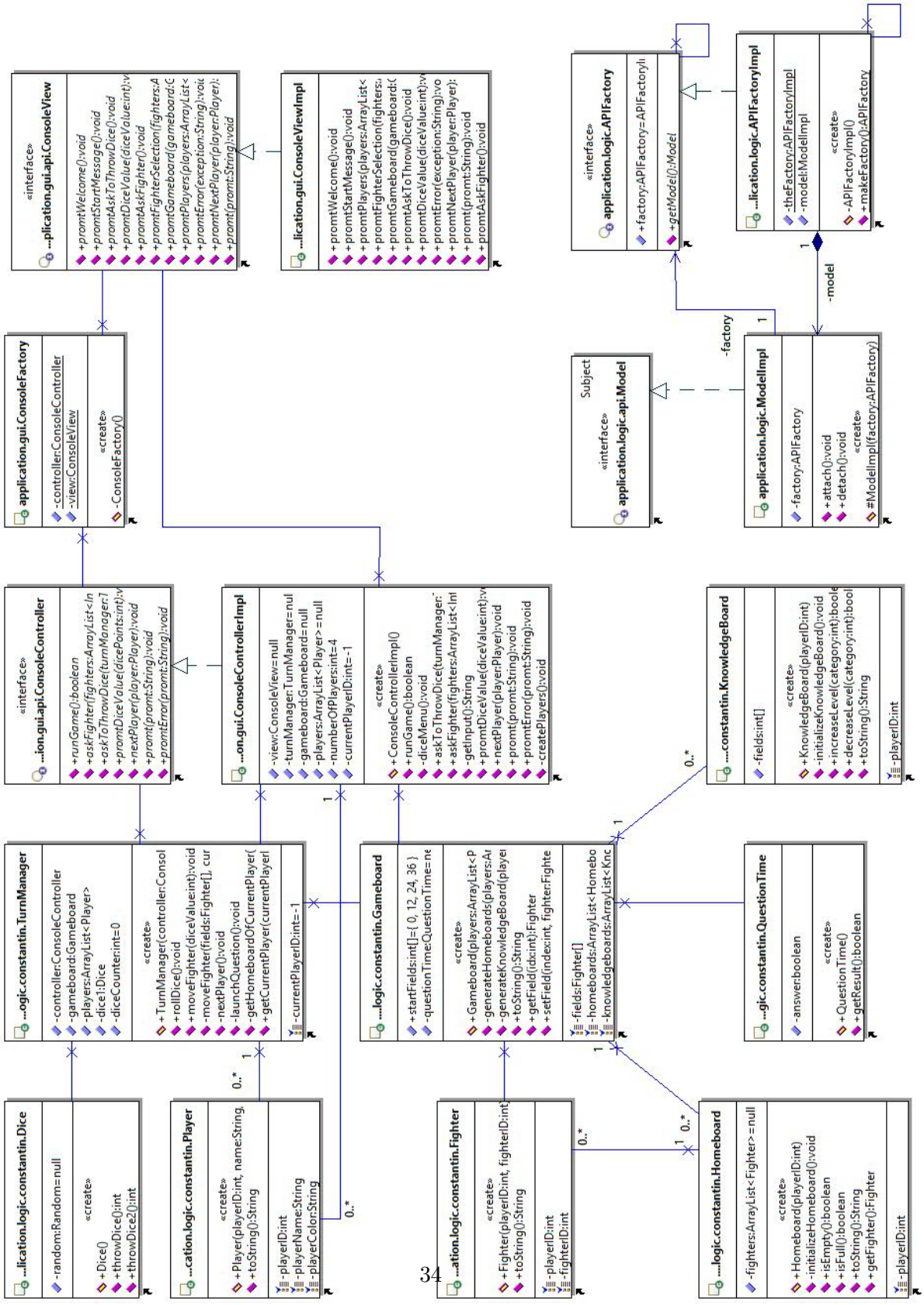
2.3.1 Sequenzdiagramm: rollDice()



2.3.2 Sequenzdiagramm: moveFighter(int diceValue)



2.3.3 Klassendiagramm



2.4 Vervollständigen Sie die Implementierung um eine einfach Oberfläche:

- Views visualisieren Zustände textuell.
- Controller verwandeln Benutzereingaben in den Aufruf entsprechender System-Operationen

Das Mockup wurde umgesetzt.