

Modellreport

Modellreport für Bodesuri

Bodesuri
Revision: 1
April 19, 2007

Approval

The original of this document is approved and signed by:

Name:

Surname:

Title:

Date:

Signature:

Revision History

Revision	Date	Description	Author
1	April 19, 2007	Use-Cases und Architektur Diagramm hinzugefügt	Pascal Hobus

Table of Contents

Model Analyse	7
Actor Spieler	7
UseCase Figur ziehen	7
UseCase Karte spielen	8
UseCase Karte tauschen	8
UseCase Spiel beitreten	8
UseCase Spiel erstellen	8
UseCase Spiel spielen	9
UseCase Starten	9
Model Design	9
Package App	9
Package Dienste	10
Package PD	10
Package UI	10

Class Diagram Architektur

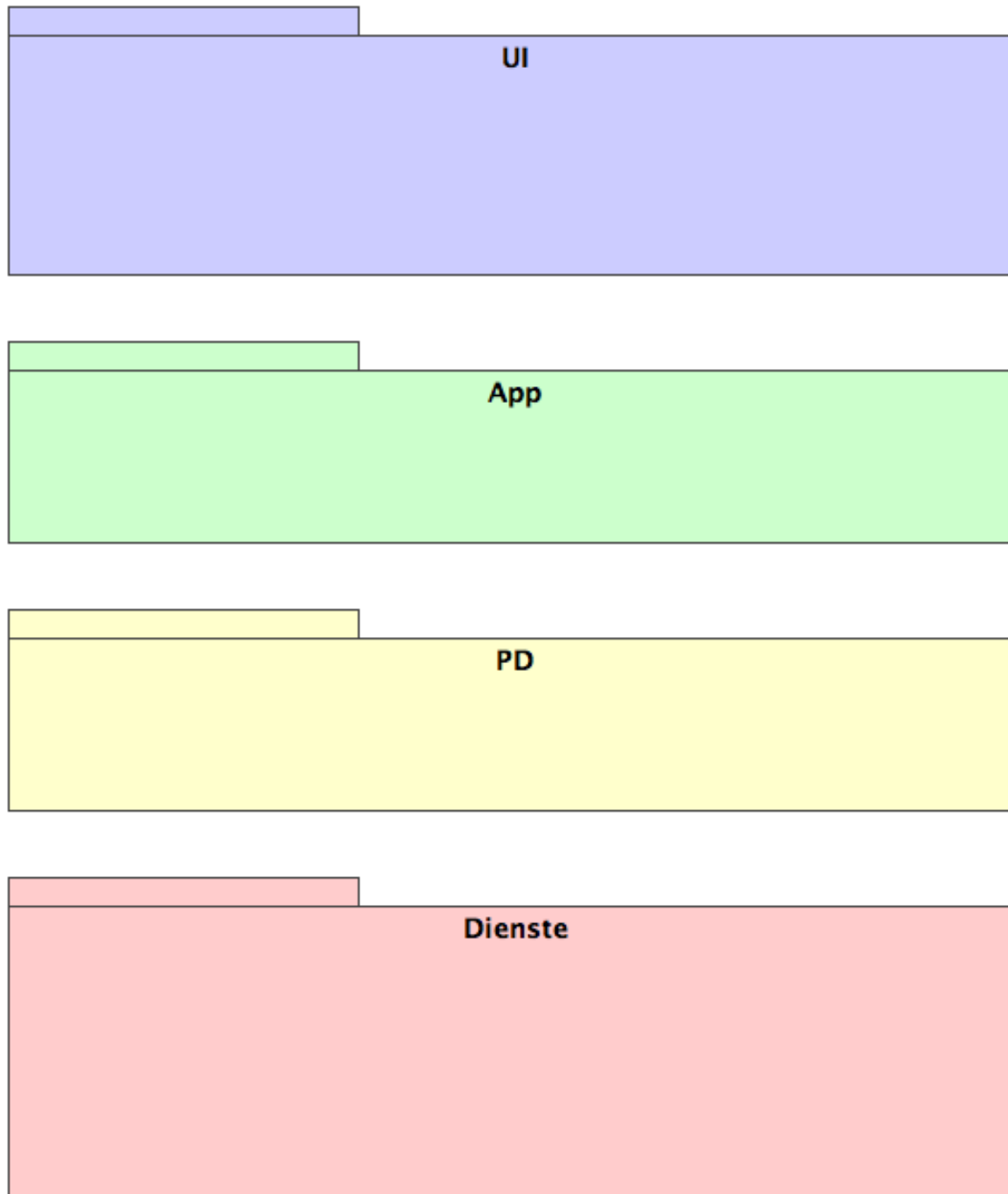



Figure 1 - Architektur Diagram

General Info

Name	Architektur
Owner	 Design
Applied Stereotype	«» DiagramInfo

Class Diagram Domain Model

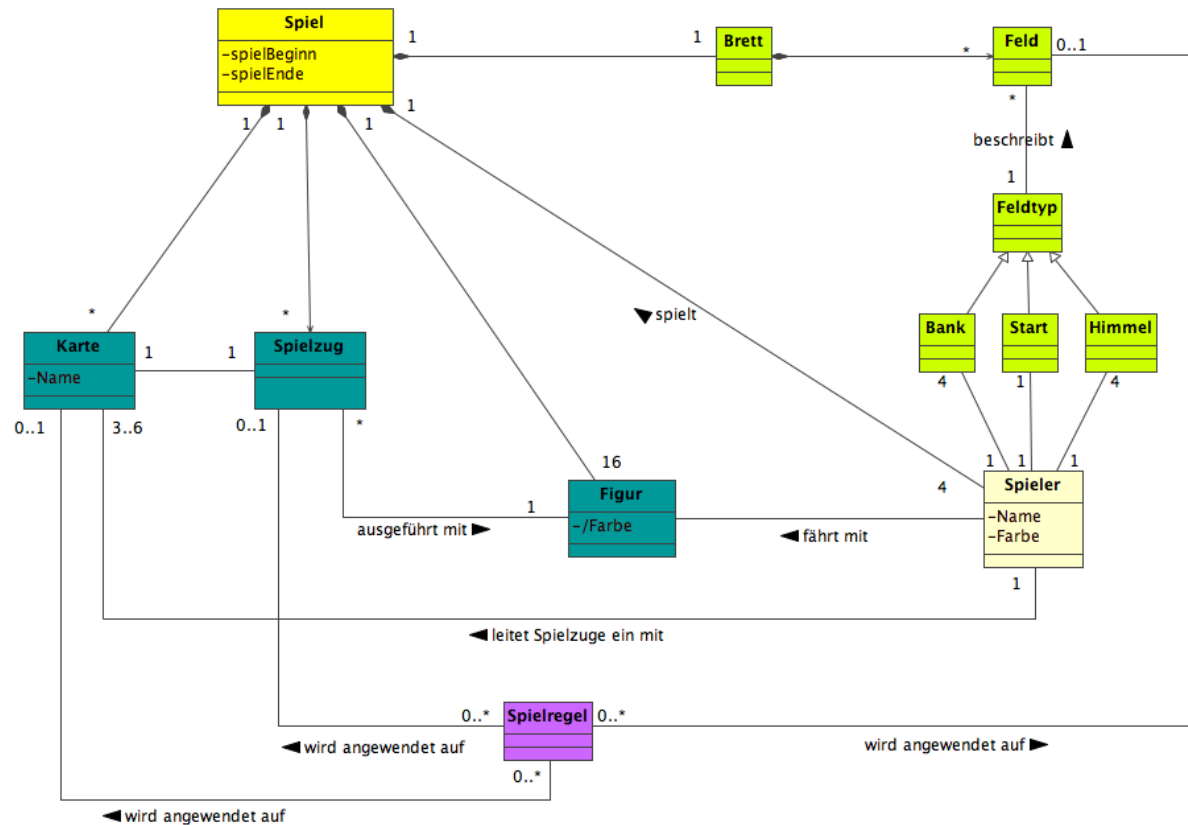


Figure 2 - Domain Model Diagram

General Info

Name	Domain Model
Owner	Analyse
Applied Stereotype	«» DiagramInfo

Sequence Diagram System-Sequenz Diagramm

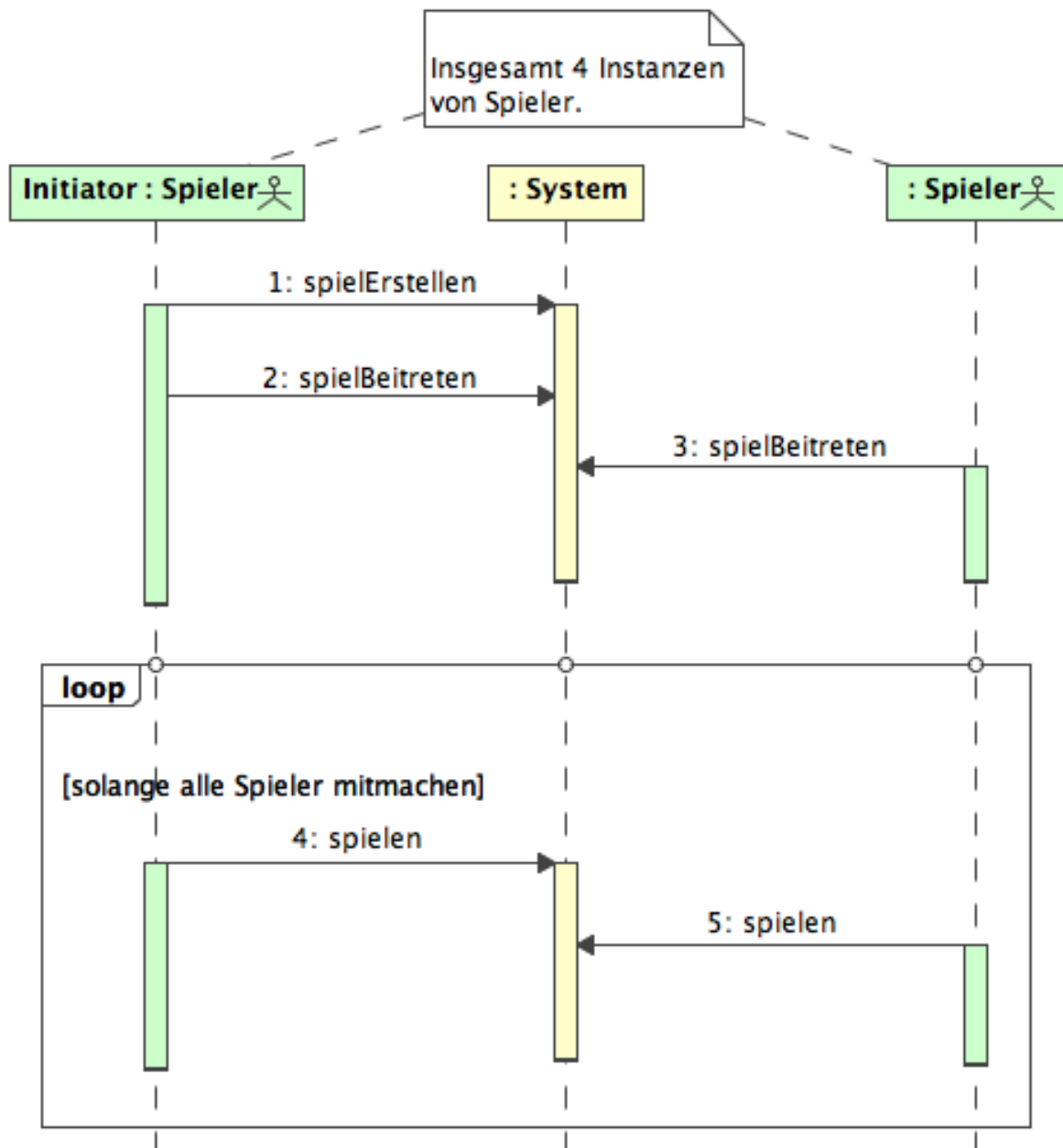



Figure 3 - System-Sequenz Diagramm Diagram

General Info

Name	System-Sequenz Diagramm
Owner	 System-Sequenz Diagramm
Applied Stereotype	«» DiagramInfo

Use Case Diagram Use-Case Diagramm

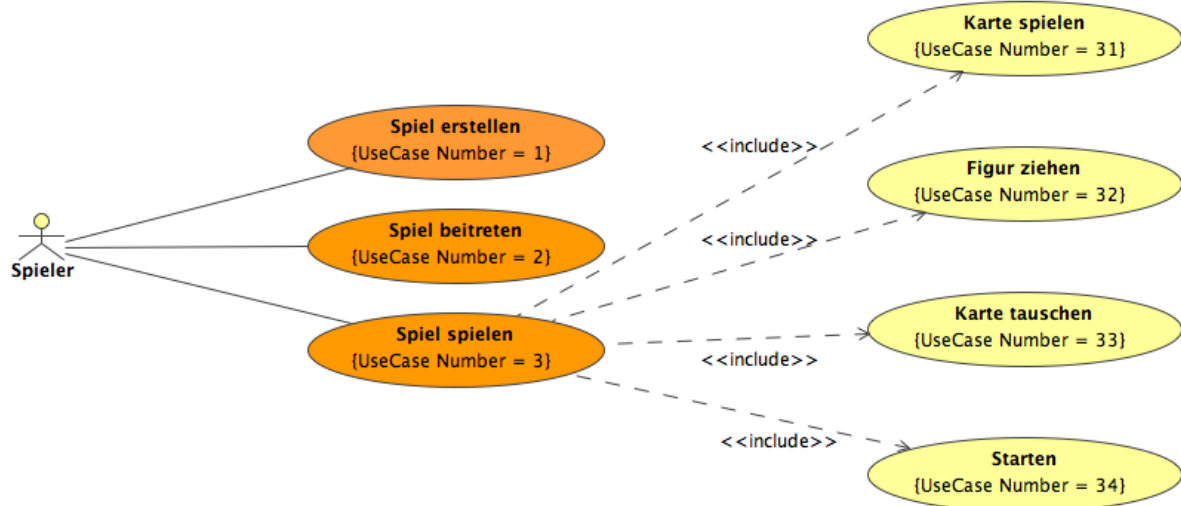


Figure 4 - Use-Case Diagramm Diagram

General Info

Name	Use-Case Diagramm
Owner	Analyse
Applied Stereotype	«» DiagramInfo

Model Analyse

General Info

Name	Analyse
Owner	Data

Actor Spieler

Documentation


Der Spieler ist der Hauptakteur des Dog-Spiels. Es gibt insgesamt vier Spieler, welche an einem Spiel beteiligt sind.

General Info

Name	Spieler
Owner	Analyse
Is Abstract	false


UseCase Figur ziehen

General Info

Name	Figur ziehen
Owner	 Analyse
Applied Stereotype	«> requirementUseCase


UseCase **Karte spielen**

General Info

Name	Karte spielen
Owner	 Analyse
Applied Stereotype	«> requirementUseCase

UseCase **Karte tauschen**

General Info


Name	Karte tauschen
Owner	 Analyse
Applied Stereotype	«> requirementUseCase

UseCase **Spiel beitreten**

Documentation

Spieler gibt einen Host an. System stellte eine Verbindung her und tritt dem Spiel bei. System zeigt bereits beigetretene Spieler an und wartet bis vier Spieler dem Spiel beigetreten sind. Sind vier Spieler beigetreten wird das Spiel begonnen.

General Info


Name	Spiel beitreten
Owner	 Analyse
Applied Stereotype	«> requirementUseCase

UseCase **Spiel erstellen**

Documentation

Spieler, welcher als Host agieren möchte, startet die Serverapplikation. System bestätigt erfolgreiche Erstellung.

General Info

Name	Spiel erstellen
Owner	 Analyse


Applied Stereotype	«» requirementUseCase
---------------------------	-----------------------

UseCase **Spiel spielen**

Documentation


Das System verteilt allen Spieler Spielkarten. Jeder Spieler tauscht mit seinem Partner eine Karte. Danach werden reihum je eine Karte gezogen und die Spielfiguren entsprechend bewegt. Wenn die Spieler alle ihre Karten gespielt haben, verteilt das System wieder neue Karten und eine neue Runde beginnt.

General Info

Name	Spiel spielen
Owner	 Analyse
Applied Stereotype	«» requirementUseCase

UseCase **Starten**

General Info


Name	Starten
Owner	 Analyse
Applied Stereotype	«» requirementUseCase

Model **Design**

Documentation

Design Diagramme für Projekt Bodesuri.

General Info

Name	Design
Owner	 Data

Package **Design::App**


Documentation

Applikationsschicht

Stellt die logische Verbindung zur Präsentationsschicht dar. Abstrahiert die Logik für die Präsentationsschicht, da diese für das Brettspiel komplex ist und vom GUI getrennt / abstrahiert sein soll.

- Handhabt GUI Anfragen
- Bedienabläufe
- Session-Zustände
- Spielzüge

General Info

Name	App
Owner	 Design

Package **Design::Dienste**


Documentation

Technische Dienste

Stellt die logische Verbindung zur Problem-Domain dar. Implementiert folgende Dienste:

- Netzwerkkommunikation (Client / Server) mittels RMI
- Datenstrukturen (low-level)
- Threads, Synchronisation
- Mathematische Berechnungen (low-level)
- Persistenz
- Sicherheitsaspekte

General Info

Name	Dienste
Owner	 Design

Package **Design::PD**


Documentation

Problem Domain

Stellt die logische Verbindung zur Applikationsschicht dar. Enthält die Grundlogik, implementiert folgende Systeme:

- Regelsystem --> Validierung
- Zugsystem --> Validierung
- Grundzustände
- Spielzustandssynchronisation

General Info


Name	PD
Owner	 Design

Package **Design::UI**

Documentation **Präsentationsschicht**

- Fenster, Views, Frames, Panels
- Java 2D Ausgaben

General Info

Name	UI
Owner	 Design