

Modellreport

Modellreport für Bodesuri

Bodesuri
Revision: 1
May 2, 2007

Approval

The original of this document is approved and signed by:

Name:

Surname:

Title:

Date:

Signature:

Revision History

Revision	Date	Description	Author
1	May 2, 2007	Use-Cases und Architektur Diagramm hinzugefügt	Pascal Hobus

Table of Contents

Model Analyse	6
Actor Spieler	6
UseCase Figur ziehen	7
UseCase Karte spielen	7
UseCase Karte tauschen	7
UseCase Spiel beitreten	7
UseCase Spiel erstellen	8
UseCase Spiel spielen	8
UseCase Starten	8
Package Applikation	8
Package ZugEntgegennahme	9
Package Zustandsynchronisation	9
Package Dienste	9
Package Client	10
Package Netzwerk	10
Package Server	10
Package Zustandsynchronisation	10
Package PD	10
Package Deck	11
Package Regelsystem	11
Package Spielerverwaltung	11
Package Validierungssystem	11
Package Zugsystem	11
Package UI	11
Package Brett	12
Package Deck	12
Package Eigenschaften	12
Package Figuren	12
Package Ressourcen	12

Class Diagram Domain Model

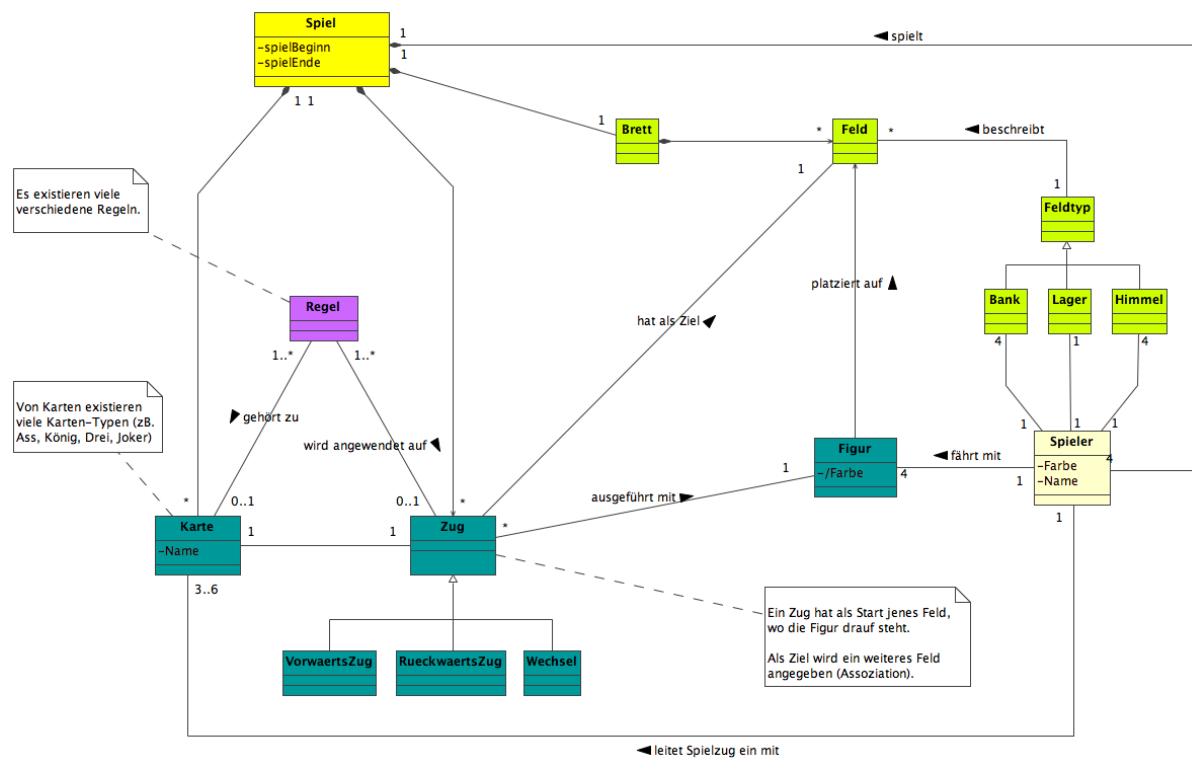


Figure 1 - Domain Model Diagram

General Info

Name	Domain Model
Owner	Analyse

Sequence Diagram System-Sequenz Diagramm

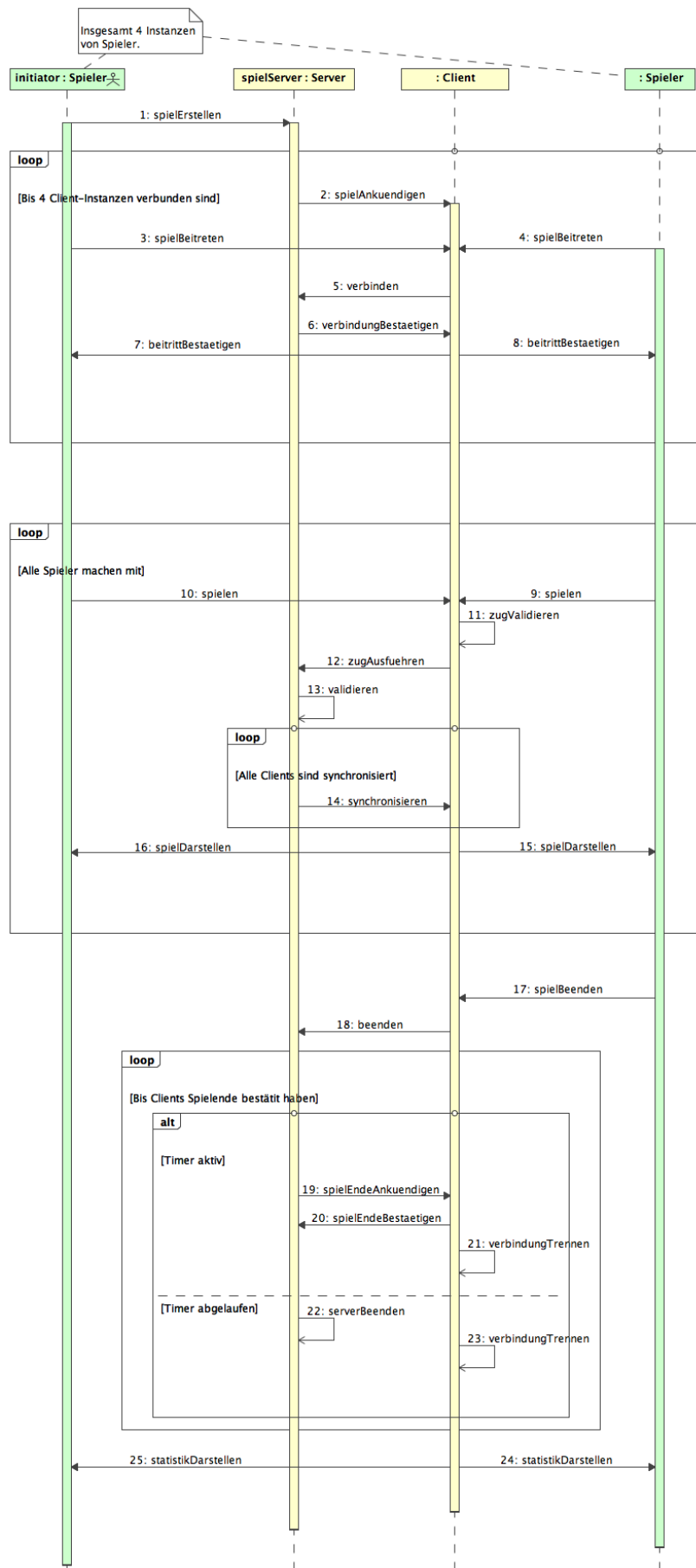



Figure 2 - System-Sequenz Diagramm Diagram

General Info

Name	System-Sequenz Diagramm
Owner	 System-Sequenz Diagramm

Use Case Diagram Use-Case Diagramm

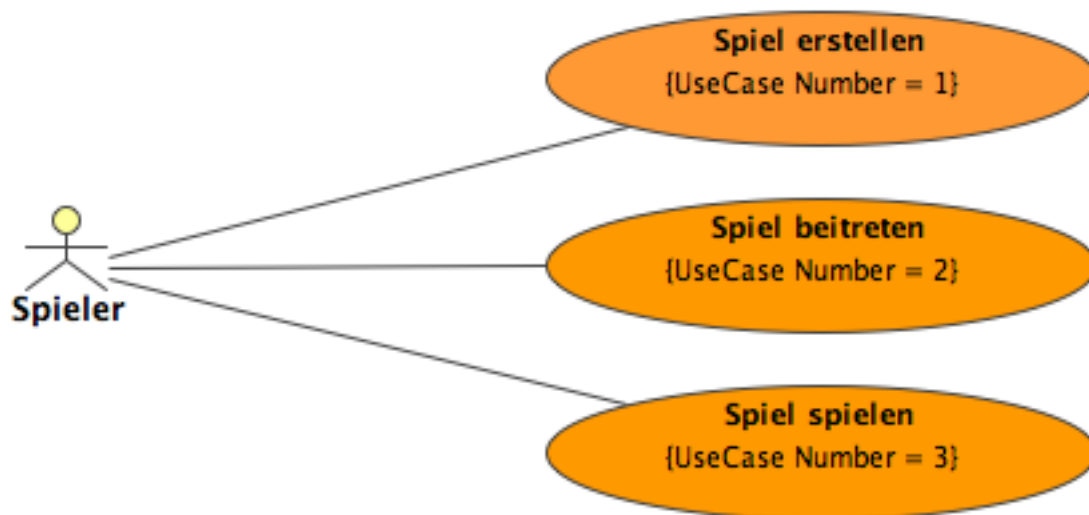



Figure 3 - Use-Case Diagramm Diagram

General Info


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

Model Analyse

Documentation

Analyse Diagramme und Modelle für das Projekt Bodesuri.

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 stellt 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

Package **Design::Applikation**

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 / abstrah-

iert sein soll.


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

General Info

Name	Applikation
Owner	 Design


Package **Design::Applikation::ZugEntgegennahme**

General Info

Name	ZugEntgegennahme
Owner	 Applikation

Package **Design::Applikation::Zustandsynchronisation**

General Info

Name	Zustandsynchronisation
Owner	 Applikation

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::Dienste::Client**

General Info

Name	Client
Owner	 Dienste


Package **Design::Dienste::Netzwerk**

General Info

Name	Netzwerk
Owner	 Dienste


Package **Design::Dienste::Server**

General Info

Name	Server
Owner	 Dienste

Package **Design::Dienste::Zustandsynchronisation**

General Info

Name	Zustandsynchronisation
Owner	 Dienste

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::PD::Deck**

General Info

Name	Deck
Owner	 PD


Package **Design::PD::Regelsystem**

General Info

Name	Regelsystem
Owner	 PD


Package **Design::PD::Spielerverwaltung**

General Info

Name	Spielerverwaltung
Owner	 PD


Package **Design::PD::Validierungssystem**

General Info

Name	Validierungssystem
Owner	 PD

Package **Design::PD::Zugsystem**

General Info


Name	Zugsystem
Owner	 PD

Package **Design::UI**

Documentation Präsentationsschicht


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

General Info

Name	UI
Owner	 Design


Package **Design::UI::Brett**

General Info

Name	Brett
Owner	 UI


Package **Design::UI::Deck**

General Info

Name	Deck
Owner	 UI


Package **Design::UI::Eigenschaften**

General Info

Name	Eigenschaften
Owner	 UI


Package **Design::UI::Figuren**

General Info

Name	Figuren
Owner	 UI

Package **Design::UI::Ressourcen**

General Info

Name	Ressourcen
Owner	 UI