# Startszenario

Spieler->Onlineanwendung: Neues Spiel starten.

Onlineanwendung-->Spieler: öffneDialog(Login)

# Da nicht genauer spezifiziert ist ob die Möglichkeit besteht,

# dass der Spielernahme eventuell nicht akzeptiert wird, muss

# man seine Daten solange dem Dialogfenster, bzw. dem Spielsys-

# tem übergeben, bis der Nutzername akzeptiert wurde. Nachdem

# die Anmeldung korrekt erfolgt ist, tritt er dem Onlinespiel

# bei.

loop Nutzername akzeptiert

Spieler->Onlineanwendung: enterName(Name)

Onlineanwendung-->Spieler: akzeptiert / nicht akzeptiert

end

Onlineanwendung-->Spieler: Onlinespiel beigetreten

# Nachdem der Spieler sich erfolgreich eingeloggt hat muss er

# auf andere Spieler (bzw. Gegner) warten. Dies erfolgt so lange,

# bis die korrekte Spieleranzahl erreicht ist. Da davon auszu-

# gehen ist, dass es bei einem Onlinespiel viele Gegner gibt,

# wird die Spielerzusammensetzung gemäß first-come-first-serve

# erstellt. Das heißt als Gegner kann man sich nur so lange an-

# melden wie es freie Plätze gibt. Die Gegner die auch "nur"

# Spieler sind nehmen aus ihrer Perspektive die Selbige sicht

# unseres hier modellierten Spielers an. Somit erfolgt die

# Kommunikation zwischen Onlineanwendung und Gegner für den ak-

# tuellen Spieler in einer "Black-Box".

loop Korrekte Spieleranzahl erreicht

Gegner->Onlineanwendung: enterName(name)

Onlineanwendung-->Gegner: akzeptiert / nicht akzeptiert

end

# Die korrekte Anzahl an Spielern wurde erreicht. Somit wird

# der Spieler zum Spielfeld geführt.

Onlineanwendung-->Spieler: visualisiere(Spielbrett)

# Nun wählt das System nach dem Zufallsverfahren aus, welcher

# Spieler beginnt.

Onlineanwendung-->Onlineanwendung: wähle 1. Spieler [random]

# Hier müssten wir eine Fallunterscheidung machen. Einmal,

# der Spieler ist am Zug. Und der Spieler ist nicht am Zug.

# Wir modellieren hier jedoch, dass der Spieler am Zug sei.

# Denn alles was die Onlineanwendung mit dem Gegner kommuniziert

# findet unter einen Black-Box statt. Da zum einem aus der Per-

# spektive des Gegners, dieses Modell zutrifft, da der Gegner

# ein Spieler ist. Und der Rest der kommunikation zwischen On-

# lineanwendung und Gegner aus Sicht des Spielers findet in einer

# Black-Box statt, da es nur die "Message Order" betrifft, aber

# nicht den Spieler direkt.

Onlineanwendung-->Spieler: Führe einen Zug durch

# Spieler würfelt.

Spieler->Spieler: würfeln

# Die Onlineanwendung verteilt nun die Rohstoffe gemäß der ge-

# würfelten Augenanzahl, die vorerst berechnet werden muss.

# Auch der Gegner erhält Rohstoffe. Hier abstrahieren wir

# wieder, da der Gegner auch ein Spieler ist, jedoch nur aus

# einer anderen Perspektive. Somit ist der Ablauf in einer

# Black-Box.

loop Alle Spieler haben Rohstoffe erhalten

Onlineanwendung-->Onlineanwendung: auszugebeneRohstoffe(Augenzahl, Bebauung)

Onlineanwendung-->Spieler: Rohstoffe

Onlineanwendung-->Gegner: Rohstoffe

end

# Nun hat der Spieler eine Auswahl zu treffen. Er kann zwischen

# Handeln, Bauen oder Aussetzen entscheiden. Da der Spieler,

# nur die 3 Alternativen hat und beliebig fortfahren kann,

# außer die Zeit ist Abgelaufen, haben wir uns dazu entschieden

# dass im else-Fall der mögliche Zugabbruch des Spielers steht.

opt Handeln, Bauen, Aussetzen

Onlineanwendung-->Spieler: Handeln

Onlineanwendung-->Spieler: Bauen

Onlineanwendung-->Spieler: Aussetzen

Onlineanwendung-->Onlineanwendung: dekrementiereInSekBis0s(300s)

alt Handeln

# Spieler kann so lange handeln wie er Rohstoffe verfügt

loop verfügbare Rohstoffe

Onlineanwendung-->Spieler: öffneDialog(Wechselkurse, Bank)

Onlineanwendung-->Spieler: öffneDialog(Wechselkurse, Gegner)

opt wähle Tauschpartner

Onlineanwendung-->Spieler: öffneDialog(wähle Tauschpartner)

alt andere Spieler

# Da der Spieler gerne mit einem Gegner handeln möchte öffnet

# die Onlineanwendung eine Sicht, sodass alle Gegner mit dazu-

# gehörigen Textfeldern aufgelistet sind. Somit kann der Spieler

# jetzt jeden Gegner einmal für einen Tausch fragen

Onlineanwendung-->Spieler: visualiere(Spielernamen, Textfelder)

# Nun kann der Spieler jeden Gegner einmal anschreiben und be-

# kommt von den Gegnern eine Antwort die er akzeptieren oder

# ablehnen kann. Da der Spieler nicht alle Gegner anschreibt

# spezifizieren wir hier die Anzahl der angeschriebenen Gegner

# mit "gewünscht" unter der Voraussetzung das man nur einmal

# den Kontakt aufnehmen darf. Somit bildet "gewünscht" zugleich

# die Abbruchbedingung.

loop gewünschte Anzahl Gegner genau einmal Kontaktiert

Spieler->Gegner: tauschantrag(Rohstoffverhältnis)

Gegner->Spieler: deterministisch(AntwortG)

Spieler->Gegner: deterministisch(AntwortS)

Onlineanwendung-->Onlineanwendung: check(AntwortG, AntwortS)

alt antwortG == antwortS

Gegner<->Spieler: Rohstofftausch

else abweichende Antworten (ja nein / nein ja)

Onlineanwendung-->Gegner: respond(keinHandel)

Onlineanwendung-->Spieler: respond(keinHandel)

end

else Bank

# Da der Spieler mit der Bank tauschen möchte bekommt er

# sofort die Rohstoffe übermittelt

Onlineanwendung-->Spieler: übergebe(Rohstoffe)

end

end

end

else Zeit abgelaufen

Onlineanwendung-->Spieler: Zeit abgelaufen

end

alt Bauen

# Spieler kann so lange handeln wie er Rohstoffe verfügt

loop verfügbare Rohstoffe

Onlineanwendung-->Onlineanwendung: Bebaute Spielfeldflächen ausgrauen

Onlineanwendung-->Spieler: öffneDialog(Möglichkeiten der Bebauung)

# Da im Text der Ablauf nicht weiter spezifiziert wurde erfolgt auch

# nur eine abstrakte Skizze des möglichen Ablaufs

opt Bebaungsstrategie

# wurde nicht weiter beschrieben

# da das Diagramm von oben nach unten ließt, meinen wir,

# dass man in dieser Stufe nicht mehr in die nächst höhere

# springen kann. lediglich in die nächst tiefere. Deshalb

# haben wir unser Design so gewählt mit der Reihenfolge

# Handel -> Bauen -> Aussetzen. Falls man tatsächlich baut

# kann man somit nicht mehr in den Fall Handeln zurückspringen.

end

end

else Zeit abgelaufen

Onlineanwendung-->Spieler: Zeit abgelaufen

end

alt Aussetzen

Onlineanwendung-->Spieler: lable(Fortfahren)

Spieler->Onlineanwendung: wähle(Fortfahren)

else Zeit abgelaufen

Onlineanwendung-->Spieler: Zeit abgelaufen

end

end

# Nun kommt die Berechnung der Siegpunkte, da der Spieler seine möglichen Punkte

# generiert hat.

Onlineanwendung-->Onlineanwendung: kalkulierePunkzahlen(Spieler)

# Es können zwei Szenarien entstehen. Gewinner steht fest oder nicht.

alt Gewinner gefunden

# Spielteilnehmer über ihren Status als Gewinner oder Verlierer informieren

Onlineanwendung-->Spieler: nachricht(Spielstatus)

Onlineanwendung-->Gegner: nachricht(Spielstatus)

# Die generierung der Punkte erfolgt intern in einer Black-Box, da dies nicht

# unmittelbar den Spielverlauf beeinflusst, lediglich den Spielstand des Ge-

# winners

Onlineanwendung-->Onlineanwendung: add(Gewinner, generierte Punkte)

# Ausschließen der Verlierer aus dem weiteren Onlinenwettbewerb [Black-Box] - mit

# selbiger Begründung wie oben, da dies das aktuelle Spielgeschehen nicht beeinflusst.

alt Verifikation[positiv]

Onlineanwendung-->Onlineanwendung: remove(Verlierer)

Onlineanwendung-->Onlineanwendung: download(Verlierer, E-Mail)

else Verifikation[negativ]

Onlineanwendung-->Onlineanwendung: remove(Verlierer)

end

else Kein Gewinner

# Onlinenspielzug beendet

end