2008

Project Software Engineering

Nick Michiels, : 0623764
Jimmy Cleuren : 0623773
Wouter Nivelle : 0624019
Kenneth Devloo : 0623746
Kevin Boutsen : 0623964

[SOFTWARE ENGINEERING]

1 Inhoudstabel

2 Overzicht probleemstelling

Maak een toepassing waarbij meerdere spelers via een netwerk, op dezelfde computer of allebei een versie van het spel Carcassone kunnen spelen. Elke gebruiker heeft de keuze uit een naam en een spelerskleur waarmee hij zich kan onderscheiden van de andere spelers. Indien er niet genoeg spelers aanwezig zijn in het spel kan men gebruik maken van een computergestuurde speler. Deze speler werkt op net dezelfde manier als een menselijke speler buiten het feit dat deze gebruikt maakt van een zelfgemaakte AI. Spelers krijgen vervolgens een aantal beurten waarin ze zogenaamde tegels op het spelbord kunnen leggen, vervolgens op die tegel manschappen plaatsen om hiermee punten te verkrijgen. De tegels kunnen bestaan uit 3 delen, namelijk weide, stad of weg. De tegels kunnen maar op een bepaalde manier tegen elkaar gelegd worden zodat het geheel er “logisch” uitziet. Hiermee wordt bijvoorbeeld bedoeld dat een weg tegen een weg moet liggen op de tegel erlangs. Steden kunnen zo bijvoorbeeld ook alleen maar met de muren tegen elkaar liggen. Hiermee moet dus rekening gehouden worden. Het plaatsen van de pionnen op de tegels moet ook nog zo bepaald worden dat een speler de pion daadwerkelijk op het stuk weide, het stuk stad of het stuk weg van de tegel kan plaatsen aangezien een tegel uit deze drie elementen kan bestaan. Per beurt kan er maar 1 pion gezet worden. Deze pionnen kunnen niet meer verplaatst worden totdat het gebied vervolledigd is. Dit heeft dan wel invloed op de punten.

Per geplaatste pion krijgt een speler dan punten gebaseerd op de grootte van de tegels die een gebied afwerken. Deze punten worden berekend zodra een gebied compleet begrensd is door andere delen zoals de weg of een stad.

De spelers moeten met elkaar kunnen praten via een interface, de handelingen van de tegenstanders kunnen zien en hierop reageren tijdens hun eigen beurt.

Als het spel uiteindelijk gedaan is, als de laatste tegel gelegd is, moet er een scorebord getoond worden met daarop de statistieken en punten van alle spelers die meegedaan hebben.

3 Gebruikers

De gebruikers van deze applicatie zijn mensen die eens een nieuw gezelschapsspel willen spelen of mensen die het echte bordspel al gespeeld hebben en het ook eens op een andere manier willen spelen. Met behulp van de netwerk functie kan er met meerdere spelers over het internet gespeeld worden. Dit draagt bij tot een fijnere en meer interactieve belevenis van het spel. De gebruikers die tegen elkaar spelen communiceren met de ingebouwde chatfuncties. Hiermee kan men berichten versturen en ontvangen. Om de competitie in het spel te vergroten is er ook een mogelijkheid voorzien om highscores te bekijken en te vergelijken. Zodoende kunnen de gebruikers proberen de anderen slimmer af te zijn en dus ook meer punten te behalen. Zo behalen ze een betere highscore.

Om de spelbelevenis optimaal te maken wordt een gebruiksvriendelijke interface voorzien zodat jong en oud overweg kunnen met het spel zonder al te veel moeite.

4 Systeemfuncties

4.1 Algemene Spelfuncties

Ref.	*Functionele Requirement*	*Info*
1.1	Het syteem moet de applicatie kunnen opstarten	
1.2	Het systeem moet een spel joinen	
1.3	Het systeem moet spel afsluiten	
1.4	Het systeem moet een spel opslaan	
1.5	Het systeem moet een spel inladen	
1.6	Het systeem moet een nieuw spel aanmaken	Deze functie moet de speler de mogelijk heden bieden om het nieuwe spel in te stellen
1.7	Het systeem moet de highscores kunnen registreren en bijhouden	
1.8	Het systeem moet met filters voor de highscores kunnen werken	
1.10	Het systeem moet het menu met opties kunnen tonen op het scherm	
1.11	Het systeem moet de applicatie kunnen afsluiten	
1.12	Het syteem moet om bevestigingen kunnen vragen	
1.13	Het systeem moet wijzigingen kunnen doorvoeren	

4.2 Spel starten

Ref.	*Functionele Requirement*	*Info*
2.1	Het systeem moet spelergegevens instellen	o.a.: naam, kleur
2.2	Het systeem moet X aantal spelers instellen	
2.3	Het systeem moet spelertype instellen	o.a.: lokaal, AI, netwerkspeler
2.4	Het systeem moet X aantal landtegels instellen	X is een begrensd getal
2.5	Het systeem moet het spel opstarten	
2.6	Het systeem moet het bord kunnen uittekenen	

4.3 Spelverloop

Ref.	*Functionele Requirement*	*Info*
3.1	Het systeem moet begin-landtegel genereren	Deze tegel bevat de 3 landeigenschappen (weg, weide en stad)
3.3	Het systeem moet random een landtegel kunnen genereren	Deze moet rekening houden met totaal aantal landtegels en de verdeling ervan
3.4	Het systeem moet een landtegel Y op locatie X leggen met een bepaalde rotatie	
3.6	Het systeem moet een pion plaatsen	
3.7	Het systeem moet testen of een spel is afgelopen	
3.8	Het systeem moet van speler wisselen	
3.9	Het systeem moet een locatie kiezen voor het leggen van een landtegel.	Rekening houdend met het feit dat de landtegel misschien niet op de aangeduide locatie mag geplaatst worden.
3.10	Het systeem moet punten berekenen en toekennen aan spelers	
3.11	Het systeem moet overige punten bij afloop van spel berekenen	
3.12	Het systeem moet bereken of een gebied is afgewerkt	Rekening houdend met het type gebied
3.13	Het systeem moet berekenen wie gewonnen is	Spelers verwittigen
3.14	Het systeem moet bepalen of een tegel op een bepaalde positie mag gelegd worden	
3.15	Het systeem moet kunnen bepalen of een pion geldig is gezet	
3.16	Het systeem moet toelaten dat tegels gedraaid kunnen worden.	
3.18	Het systeem moet de mogelijkheid bieden om een pionpositie aan te duiden	

3.20	Het systeem moet kunnen bevestigen of een actie met een tegel of pion goed is.	
3.21	Het systeem moet een speler de mogelijkheid geven om een positie opnieuw te kiezen	
3.22	Het systeem moet een speler de mogelijkheid geven om een zet over te slaan	
3.23	De speler moet de mogelijkheid hebben om een pion te kiezen uit zijn lijst van pionnen	

4.4 Netwerk

Ref.	***Functionele Requirement***	***Info***
4.1	Het systeem moet netwerkspelers ondersteunen	Over een netwerk met andere personen kunnen spelen
4.2	Het systeem moet een connectie maken met een host	Host controleert het spel
4.3	Het systeem moet een connectie opstellen voor een host	
4.4	Het systeem moet slots kunnen openen voor netwerspelers	
4.5	Het systeem moet spelgegevens kunnen doorsturen	
4.6	Het systeem moet een connectieverbreking opvangen	Door plaatsvervanging van netwerkspeler door AI
4.7	Het systeem moet een chatbericht kunnen sturen naar een server	
4.8	Het systeem moet een chatbericht kunnen sturen naar alle spelers	
4.9	Het systeem moet een chatbericht kunnen laten zien	

4.5 AI

Ref.	***Functionele Requirement***	***Info***
5.1	Het systeem moet een AI bevatten	

5 Systeemattributen volgens de FURPS

5.1 Functionality

Onze applicatie moet op zijn minst al over een basisfunctionaliteit beschikken waarmee het spel kan worden gespeeld. In dit spectrum van functionaliteit zit o.a. het aangeven van landtegels, het plaatsen van landtegels op het spelbord, het plaatsen van pionnen op het spelbord, het berekenen van punten, het berekenen of een gebied volledig is en de ondersteuning van meerdere gebruikers. Naast deze basisfunctionaliteit zal onze applicatie een netwerk gaan ondersteunen waardoor externe gebruikers zich kunnen aanmelden in een spel. Deze externe spelers zullen dan ook via een chatfunctie met elkaar kunnen communiceren. Verder is er ook de mogelijkheid om een speler als AI te beschouwen met een bepaalde moeilijkheidsgraad.

5.2 Usability

Het programma dat we maken is bedoeld als entertainment. De usability is dus van groot belang. Niet alleen moet de gebruiker vlot door het hoofdmenu kunnen navigeren om bijvoorbeeld een nieuw spel op te starten. Hij moet ook duidelijk weten hoe en wanneer hij met het spel moet interageren. Het is van groot belang dat het programma duidelijk aangeeft hoe de speler een landtegel trekt en hoe hij deze kan plaatsen op het spelbord. Daarnaast moet hij in één oogopslag kunnen zien welke zijn pionnen zijn, om deze vervolgens gemakkelijk op het spelbord te plaatsen. Dit alles zal verwezenlijkt moeten worden met een degelijke UI.

5.3 Reliability

Het is vanzelfsprekend dat een speler moet kunnen betrouwen dat, bij het leggen van een landtegel of pion, het systeem ook effectief op de juiste plaats de actie uitvoert. Of, beter, dat een speler niet een verkeerd aantal punten krijgt toegewezen. Indien de algoritmes hiervoor goed zijn uitgewerkt zal dit 'normaal' geen ongewenste effecten geven. Bij netwerkspellen moeten de acties van alle spelers na een zet onmiddellijk kunnen geregistreerd worden door andere spelers.

Ten tweede moet op de AI ook worden vertrouwd dat deze correcte spelacties uitvoert. Het is niet zo dat de AI boven het spel staan en dus acties onderneemt die tegen de spelregels zijn.

5.4 Performance

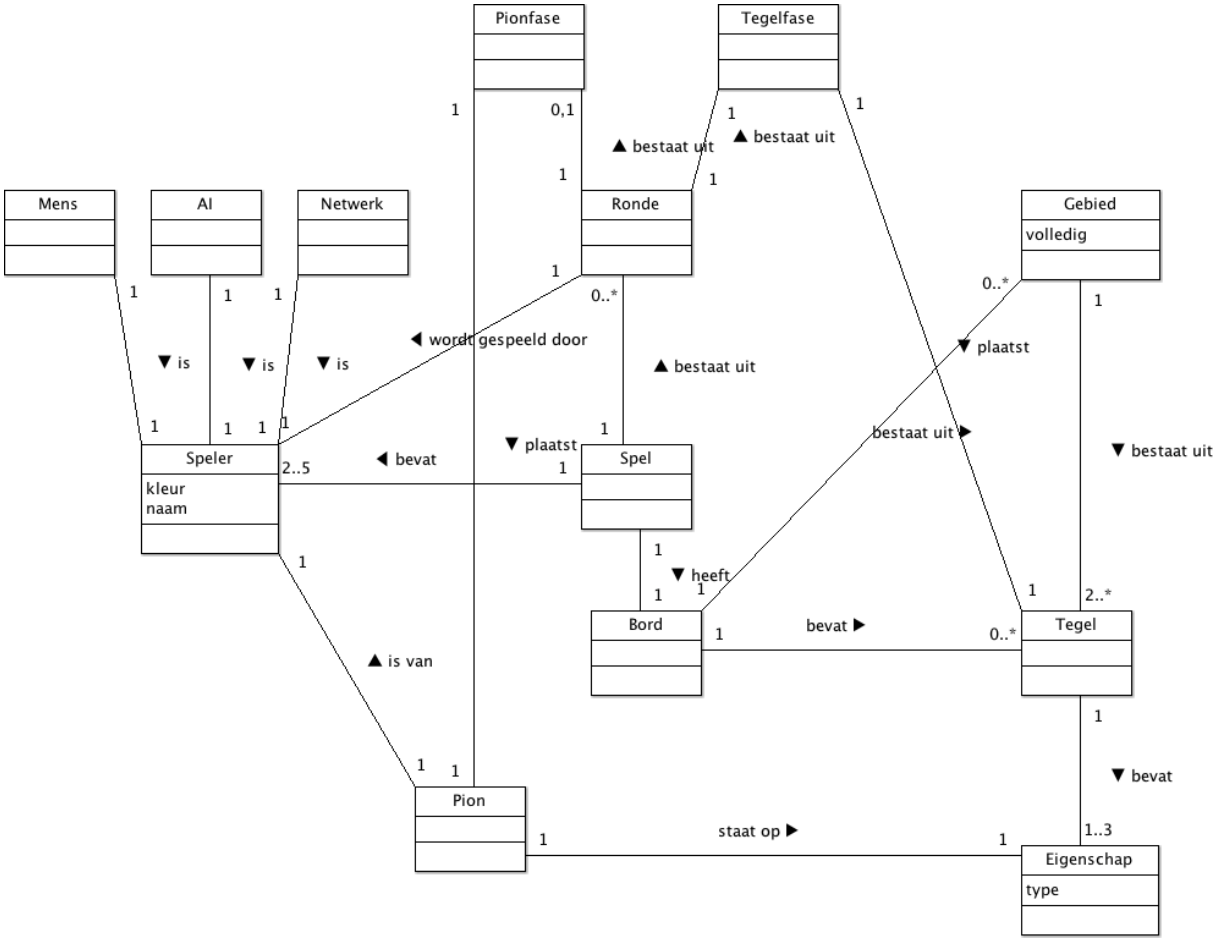
Ons spel zal over het algemeen 2 grote 'bottlenecks' hebben. Op deze twee plaatsen zullen we dus vooral moeten gaan optimaliseren wat betreft performance. De eerste bottleneck is het berekenen of een gebied volledig is. De tweede het gebruik van een AI. De AI moet de mogelijkheid bieden om snel te berekenen waar de landtegel moet geplaatst worden en waar de beste plaats is voor een pion. Andere functionaliteit waar op performance moet gelet worden is: snel doorgeven van speldata met netwerkspelers, punten berekenen, bepalen of een gebied volzet is met een pion en bepalen of een landtegel op een bepaalde positie mag gelegd worden.

5.5 Supportability

De applicatie is bedoeld om op meerdere platformen te kunnen werken. Tussen de verschillende platformen moet ook een consequente netwerkverbinding kunnen ontstaan. Op grafisch vlak moet de applicatie toonbaar zijn op verschillende resoluties. Met behulp van een updater

kunnen we de spelers updates laten installeren om eventueel fouten uit het systeem te halen. Op vlak van foutcorrectie zullen we fouten loggen zodat we een overzicht hebben van waar de applicatie faalt en zo snel onze applicatie kunnen debuggen.

6 Domeinmodel



7 Use Cases

7.1 UC1 Spel opstarten

7.1.1 UC1.1 Brief

Actoren: Speler
Beschrijving:

De applicatie wordt gestart en het eerst beeld wordt op het scherm getoond. De speler krijgt een menu te zien waar hij kan kiezen een spel te starten met computerspelers en/of andere spelers (eventueel over netwerk) of een spel dat al gestart is door een andere netwerkspeler te vergezellen. Bij de eerste opties kan de speler de spelregels opstellen en het aantal spelers. Hij kiest dan ook het type spelers (AI, speler, netwerkspeler) en het aantal tegels waarmee gespeeld kan worden. Indien alle opties zijn ingesteld en de netwerkspelers gereed zijn kan de speler het spel starten.

7.1.2 UC1.2 Fully Dressed

7.1.2.1 Basic Flow

Primary Actor: Speler
Stakeholders and Interests:
Spelers: Willen een snel en betrouwbare opstart van het spel.
Preconditions:
Het spel is juist geïnstalleerd (alle libraries zijn aanwezig).
Het spel is compatible met het operating system.
Postconditions:
Het spelbord wordt op het scherm getoond.
Elke speler heeft een naam en pion met zijn gekozen kleur.
De eerste speler is gekozen.
Gebruikte sub use cases: UC 2
Cross references: FR 1.1, FR 1.6, FR 2.1, FR 2.2, FR 2.3, FR 2.4, FR 2.5, FR 2.6, FR 4.4

Speler	*System*
1.a Speler opent applicatie (FR 1.1)	1.b Systeem start applicatie (FR 1.1) 1.c Systeem laadt nodige files in 1.d Systeem toont applicatie op het scherm
2.a.a Speler selecteert nieuw spel (FR 1.6) 2.a.c Speler selecteert spelopties	2.a.b Systeem toont spelopties (FR 1.6)
2.a.d Speler wacht eventueel op andere spelers	2.a.e Systeem wacht op netwerkspelers (FR 4.4) (UC 2)
2.a.f Speler start nieuw spel (FR 2.5)	2.a.h Systeem controleert spelopties 2.a.i Systeem start spel met geselecteerde spelopties (FR 2.1 – FR 2.5)

	3.Systeem toont bord op scherm (FR 2.6) 4. Systeem kiest speler die beginnen mag

7.2 UC2 Spel Joinen

7.2.1 UC2.1 Brief

Actoren: Speler
Beschrijving:

De applicatie wordt gestart en het eerst beeld wordt op het scherm getoond. De speler krijgt een menu te zien waar hij kan kiezen een spel te starten met computerspelers en/of andere spelers (eventueel over netwerk) of een spel dat al gestart is door een andere netwerkspeler te vergezellen. De speler kiest hier dan voor de opties om een ander spel op het netwerk te vergezellen. Deze client maakt een verbinding met de server die de gegevens van het spel zal doorsturen van zodra de speler als klaar geseint is en het spel gestart wordt.

7.2.2 UC2.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:
Spelers: Willen een snel en betrouwbare opstart van het spel.
Preconditions:
Het spel is juist geinstalleerd (alle libraries zijn aanwezig).
Het spel is compatible met de operating system.
Postconditions:
Het spelbord wordt op het scherm getoond.
Netwerkverbinding gemaakt met host.
Speler heeft een naam en een pion in de gekozen kleur.
Gebruikte sub use cases: geen
Cross references: FR 1.1, FR 1.2, FR 2.6, FR 4.1, FR 4.2, FR 4.3, FR 4.5

7.2.2.1 Basic Flow

Speler	*System*
1.a Speler opent applicatie	1.b Systeem start applicatie (FR 1.1) 1.c Systeem laadt nodige files in 1.d Systeem toont applicatie op het scherm
2.c.a Speler selecteert “join” spel (FR 1.2)	2.c.b Systeem start netwerkconnectie (FR 4.1) 2.c.c Systeem zoekt naar spellen op netwerk
2.c.d Speler kiest server 2.c.f Speler stelt opties in 2.c.g Speler is gereed	2.c.e Systeem maakt connectie met server (FR 4.2 , FR 4.3) 2.c.i Systeem stuurt bericht naar server dat client gereed is. (FR 4.5)

	2.c.j Systeem wacht op gegevens server (FR 4.5) 3. Systeem toont het spel op het scherm (FR 2.6) 4. Systeem kiest de speler die mag beginnen

7.3 UC3 Spelverloop

7.3.1 UC3.1 Brief

Actoren: Spelers
Beschrijving:

Er is een wereld opgestart door het systeem. Het systeem gaat een speler aanduiden die mag beginnen. Verder gaat het systeem ook een begintegel aanmaken en deze op het spelbord leggen. Beginnende met de aangeduide speler worden elke spelers afgegaan totdat alle tegels op het spelbord liggen en dus het spel is afgelopen. In een beurt kan een speler een tegel trekken totdat hij deze kan afleggen. Als de speler een landtegel plaatst, kan het systeem nagaan of er punten moeten worden uitgedeeld. Vervolgens kan hij een pion van zijn kleur plaatsen op een tegel. Na het plaatsen van deze pion is zijn beurt afgelopen. Bij het aflopen van het spel worden de highscores aangepast en eventueel eindpunten uitgedeeld.

7.3.2 UC3.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:
Spelers: Willen een vlot en snel spelverloop dat heel overzichtelijk in elkaar zit.
Preconditions:
X aantal spelers zijn aanwezig.
Elke speler heeft zijn eigen kleur en naam.
Verbindingen met netwerkspelers zijn tot stand gebracht **.**
Een leeg spelbord is ingeladen.
Postconditions:
Er is een winnaar bekend.
Highscores moeten upgedated zijn.
Gebruikte sub use cases: UC 4, UC 5, UC 6
Cross references: FR 2.5, FR 3.1, FR 3.3, FR 3.4, FR 3.6, FR 3.7, FR 3.8, FR 3.10, FR 3.11, FR 3.12, FR 3.13, FR 4.6, FR 5.1

7.3.2.1Basic Flow

Speler	*System*
0. Host start het spel (FR 2.5)	
	1. Begintegel genereren (FR 3.1) 2. De beurt wordt doorgegeven aan de volgende speler. (FR 3.8)
3a.a. Speler trekt een landtegel (FR3.3)	3a.b. Landtegel toewijzen aan speler X (FR 3.3)

3b. Speler kiest plaats waar hij de landtegel wil leggen. (UC 4) (FR 3.4)	3c. Landtegel op het spelbord leggen (FR 3.4) 3d. Punten uitdelen indien nodig (FR 3.10 + FR 3.12)
3e. Speler kiest een plaats voor een pion (UC 5)	3f. Pion wordt geregistreerd door het systeem (FR 3.6) 3g. Systeem checkt of het spel afgelopen is (FR 3.7) 3h. De beurt wordt doorgegeven aan de volgende speler. (3.8)

* Herhaal stap 3 totdat het spel is afgelopen

4. Overige punten worden verdeeld (3.11)
5 Berekenen wie gewonnen is en dit laten weten aan de spelers (FR 3.13)
6. Highscores worden berekend (UC 6)

7.3.3 UC3.2a Alternative flow 1

Alternative flow: Een speler kiest ervoor om het spel te verlaten

Speler	*System*
* Ergens in het spelverloop verlaat speler het spel	1. Connectie van host met speler wordt verbroken (FR 4.6) 2. Speler wordt vervangen door een AI (FR 4.6)

* Het spel kan hervat worden op de plaats waar het is gestopt

7.3.4 UC3.2b Alternative flow 2

Alternative flow: De speler plaatst geen landtegel

Speler	*System*
* Speler heeft geen landtegel geplaatst	1. Random de tegel ergens leggen (FR 3.4) 2. Beurt doorgeven aan volgende speler (FR 3.8)

* Het spel kan hervat worden met de nieuwe speler

7.3.5 UC3.2c Alternative flow 3

Alternative flow: De speler kiest er voor geen pion te plaatsen

Speler	*System*
* Speler heeft geen pion geplaatst	1. Beurt doorgeven aan volgende speler (FR 3.8)

* Het spel kan hervat worden met de nieuwe speler

7.3.6 UC3.2d Alternative flow 4

Alternative flow: Een netwerkspeler kiest ervoor het spel te verlaten

Speler	*System*
* Netwerkspeler verlaat het spel	1. Speler wordt vervangen door AI (FR 5.1, 4.6)

* Het spel kan hervat worden met de AI speler

7.4 UC4 Landtegel positie kiezen

7.4.1 UC4.1 Brief

Use Case: Landtegel positie kiezen
Actoren: Spelers
Beschrijving: De speler kiest een correcte positie om zijn landtegel neer te leggen.

7.4.2 UC4.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:

- Speler: Wil een tegel plaatsen op een correcte plaats.
- Medespelers: Willen niet dat de speler vals kan spelen.

Preconditions:

- De speler is aan beurt en heeft zijn tegel nog niet geplaatst.

Postconditions:

- De tegel kan op het bord gelegd worden.

Gebruikte sub use cases: geen
Cross references: FR 3.4, FR 3.9, FR 3.14, FR 3.16, FR 3.20,FR 3.21, FR 3.22

7.4.2.1Basic Flow

Speler	*System*
1a Speler kiest een rotatie (FR 3.16) 2a Speler kiest een positie (FR 3.9)	1b systeem bevestigt (FR 3.20) 2b Systeem controleert de zet (FR 3.14) 2c Systeem aanvaardt de zet (FR3.4)

7.4.2.2 UC4.2a Alternative Flow

Speler	*System*
1.c Speler kiest nieuwe rotatie (FR 3.16)	1.d basic flow hervat bij 1b
2.a Speler kiest een positie (FR 3.9)	2.b Systeem controleert de zet (FR 3.14) 2.c Systeem merkt dat een stuk weg doodloopt op een andere tegel 2.d Systeem stelt speler op de hoogte(FR 3.20) 2.e Systeem laat de speler opnieuw een positie kiezen (basic flow hervat bij 1.b) (FR 3.21)
2.a Speler kiest een positie (FR 3.9)	2.b Systeem controleert de zet(FR 3.14) 2.c Systeem merkt dat een stuk 'open' stad niet doorloopt op een andere tegel 2.d Systeem stelt speler op de hoogte(FR 3.20) 2.e Systeem laat de speler opnieuw een positie kiezen (basic flow hervat bij 1.b) (FR 3.21)
2.a Speler wil geen tegel plaatsen(FR 3.22)	2.b Systeem aanvaard(alternatieve postconditie 2) (FR 3.22)

7.5 UC5 Pion plaatsen

7.5.1 UC5.1 Brief

Use Case: pion positie kiezen
Actoren: Spelers
Beschrijving: De speler kiest een correcte positie om zijn pion te plaatsen.

7.5.2 UC5.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:

- Speler: Wil een pion plaatsen op een correcte plaats.
- Medespelers: Willen niet dat de speler vals kan spelen.

Preconditions:

- De speler is aan beurt en heeft zijn pion nog niet geplaatst.

Postconditions:

- De pion kan op het bord gelegd worden.

Gebruikte sub use cases: geen

Cross references: FR 3.6, FR 3.15, FR 3.18, FR 3.20, FR 3.21, FR 3.22, FR 3.23

7.5.2.1 Basic Flow

Speler	*System*
1a Speler bedenkt een zo goed mogelijke manier om een pion te plaatsen	
2.a Speler kiest een positie (FR 3.18)	2.b Systeem controleert de positie (FR 3.15) 2.c Systeem aanvaardt de positie (FR 3.6)

7.5.2.2 UC5.2a Alternative Flow

Speler	*System*
2.a Speler kiest een pion (FR 3.23) 2.c Speler kiest een positie (FR 3.18)	2.b Systeem bevestigd en toont de keuze op het scherm(FR3.20) 2.d Systeem controleert de positie (FR 3.15) 2.e Systeem merkt dat er geen juist soort stuk land is gekozen of dat dat stuk al bezet is. 2.f Systeem stelt speler op de hoogte(FR3.20) 2.g Systeem laat de speler opnieuw een positie kiezen en basic flow wordt hervat bij 2.c (FR3.21)
2.a Speler kiest geen pion (FR 3.22)	2.b Systeem bevestigd en toont de keuze op het scherm(FR3.20) 2.c Systeem aanvaardt (FR 3.22)

7.6 UC6 Highscore registreren

7.6.1 UC6.1 Brief

Use Case: Highscore registreren

Actoren: speler

Beschrijving: Op het einde van een spel krijgen gebruikers de kans hun behaalde score te registreren als highscore. Hun score wordt ook geregistreerd bij andere pc's.

7.6.2 UC6.2 Fully Dressed

Use Case: Highscore registreren

Primary Actors: speler

Stakeholders and Interests:

- Spelers: Ze willen dat hun aantal gewonnen, verloren en gelijkgespeelde spellen geregistreerd worden .

Preconditions:

- Er is net een spel uitgespeeld.

Postconditions:

- De high scores zijn geregistreerd bij alle aan het spel deelnemende pc's.

Gebruikte sub use cases: UC 7

Cross references: FR 1.7

7.6.2.1 Basic Flow

Speler	*System*
1.a.a Speler kiest om highscores te registreren	1.a.b Systeem slaat de nieuwe scores op (FR 1.7)
	1.a.c Systeem meldt dat de high scores zijn geregistreerd
2.a.a Speler kiest om de huidige nieuwe scores te bekijken. (UC 7)	2.a.b Systeem toont de totale high scores van de huidige spelers op het scherm
2.a.c Speler kiest om het highscorescherm af te sluiten	2.a.c Systeem sluit het venster met highscores
2.b.a Speler kiest om het highscorescherm niet weer te geven	

7.7 UC7 Highscores bekijken

7.7.1 UC7.1 Brief

Use Case: Highscore bekijken
Actoren: speler
Beschrijving: De speler kan bekijken hoeveel keer hij, en andere vroegere tegenspelers, een spel gewonnen, verloren of gelijkgespeeld heeft.

7.7.2 UC7.2 Fully Dressed

Primary Actor: speler

Stakeholders and Interests:

- Spelers: Ze willen een overzicht op de highscores, eventueel met scores van andere netwerkspelers weggefilterd.

Preconditions:

- Het spel is juist geïnstalleerd (alle libraries zijn aanwezig).
- Het spel is compatible met het operating system.

Postconditions:

- De juiste high scores worden getoond.
- De scores van netwerkspelers zijn weggefilterd.

Gebruikte sub use cases: geen

Cross references: FR 1.1, FR 1.7, FR 1.8, FR 1.11

7.7.2.1Basic Flow

Speler	*System*
1.a Speler opent applicatie (FR 1.1)	1.b Systeem start applicatie (FR 1.1) 1.c Systeem laadt nodige files in 1.d Systeem toont applicatie op het scherm
2.a Speler selecteert Highscores	2.b Systeem toont het scherm met high scores (FR 1.7)
3.a.a Speler maakt geen gebruik van filters.	3.a.b Systeem blijft alle high scores tonen op het scherm
3.b.a Speler kiest een filter voor te sorteren (FR 1.8)	3.b.b Systeem toont alle high scores die lokaal behaald zijn

7.7.2.2 UC7.2a Alternative Flow

Speler	*System*
1.a Speler opent applicatie	1.b Systeem crasht bij de start. 1.c De applicatie wordt afgesloten (FR 1.11)
2.a.a Speler selecteert High scores	2.a.b Systeem toont een leeg scherm 2.a.c Systeem bericht dat er geen high scores zijn
3.a.a Speler maakt geen gebruik van filters	3.a.b Systeem toont een leeg scherm 3.a.c Systeem bericht dat er geen high
3.b.a Speler kiest de filter	3.b.b Systeem toont een leeg scherm 3.b.c Systeem bericht dat er geen high scores zijn

7.8 UC8 Spel opslaan

7.8.1 UC8.1 Brief

Actoren: Spelers
Beschrijving:

Tijdens het spelen van het spel wordt er gekozen door een speler om het spel op te slaan. De speler geeft de bestandsnaam naar waar het moet worden weggeschreven. Vervolgens gaat het systeem al de nodige informatie in een file wegschrijven

7.8.2 UC8.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:
Spelers: Willen snel een spel kunnen opslaan en er ook op vertrouwen dat het spel correct is opgeslagen.
Preconditions:
De speler is aan het deelnemen in een spel.
Postconditions:
Het spel is weggeschreven in een file.

Het spel kan gewoon verder worden gespeeld

Gebruikte sub use cases: geen

Cross references: FR 1.4

7.8.2.1 Basic Flow

Speler	*System*
	*spel onderbroken door een speler
1. Spel opslaan in bestand X (FR 1.4)	
	2. Spel opslaan in bestand X (FR 1.4)
	*spel wordt terug hervat

7.9 UC9 Spel inladen

7.9.1 UC9.1 Brief

Actoren: Spelers

Beschrijving:

Een speler kan ervoor kiezen om een spel in te laden. Bij het inladen van een spel wordt er een bestandsnaam gevraagd aan de speler. Het systeem gaat dit bestand inladen. Elke speler die netwerkspeler was in het spel wordt ingesteld als AI

7.9.2 UC9.2 Fully Dressed

Primary Actor: Speler

Stakeholders and Interests:

Spelers: Willen snel een spel kunnen inladen om het te hervatten. Ze willen er op kunnen vertrouwen dat het spel exact hetzelfde verder gaat dan dat het afgesloten is.

Preconditions:

De applicatie is opgestart.

Postconditions:

Er is een bestand ingeladen.
Het ingeladen spel kan hervat worden.
Alle netwerkspelers zijn ingesteld als AI.

Gebruikte sub use cases: geen

Cross references: FR 1.5

7.9.2.1 Basic Flow

Speler	*System*
1. Spel in bestand X inladen (FR 1.5)	
	2. Bestand X inladen (FR 1.5)
	*spel wordt terug hervat

7.10 UC10 Spel opties

7.10.1 UC10.1 Brief

Actoren: Speler

Beschrijving:
De speler kan in het menu nog opties instellen voor betere performance en usability. Zo kan deze bijvoorbeeld de resolutie instellen en aanduiden of hij in een window of fullscreen wil spelen. Hij kan geluidseffecten en achtergrondgeluid in- of uitschakelen. De speler kan de nieuwe instellingen accepteren waarop het systeem zal vragen of deze zeker is. Indien ja zal het programma indien nodig het spel herstarten met de gewenste instellingen.

7.10.2 UC10.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:
Spelers: Willen het spel naar hun wenst kunnen maken.
Preconditions:
Systeem laat enkel opties zien die het systeem aankan.
Postconditions:
Speler krijgt melding ter confirmatie.
Spel wordt onmiddellijk aangepast met de nieuwe instellingen.
Gebruikte sub use cases: UC 1
Cross references: FR 1.10, FR 1.12, FR 1.13, FR1.1

7.10.2.1 Basic Flow

Speler	*System*
1. Speler kiest optie menu	2 Systeem laadt menu in (FR 1.10)
3.a Speler stelt resolutie in	
3.b Speler duidt fullscreen aan	
3.c Speler verandert geluidseffecten	
3.d Speler veranderd achtergrondgeluid	5. Systeem stuurt melding naar gebruiker (FR 1.12)
4. Speler klikt op Aanpassen	
6. Speler accepteert melding	7. Systeem veranderd de instellingen (FR 1.13)
	8. Systeem opent hoofdmenu (FR 1.1)

7.10.2.2 UC10.2a Alternative Flow

Speler	*System*
1. Speler opent optiemenu	2. Systeem toont menu (FR 1.10)
3. Speler beëindigt menu voortijdig	4. Systeem opent hoofdmenu (FR 1.1)

7.11 UC11 Spel afsluiten

7.11.1 UC11.1 Brief

Actoren: Speler

Beschrijving:
De speler wilt het spel afsluiten. Er kan op dit moment nog een spel bezig zijn, dus moet er gevraagd worden of het spel opgeslagen moet worden, of het spel is afgelopen en er is geen nieuw spel meer gestart. Na deze controle sluit de applicatie zich af.

7.11.2 UC11.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:
Spelers: Willen de applicatie eenvoudig en snel kunnen afsluiten.
Preconditions:
De applicatie is opgestart.
De speler bevindt zich in het hoofdmenu en er is dus geen spel meer bezig.
Postconditions:
De applicatie wordt zonder fouten afgesloten.
Gebruikte sub use cases: geen
Cross references: FR 1.12, FR 1.3

7.11.2.1 Basic Flow

Speler	*System*
1. Speler wil het spel afsluiten	
	2. Systeem vraagt om bevestiging van afsluiten (FR 1.12)
3.a.a Speler bevestigt afsluiten	
	3.a.b Spel sluit af (FR 1.3)
3.b.a Speler wil niet afsluiten	
	3.b.b Systeem gaat verder met uitvoering

7.12 UC12 Chatbericht versturen

7.12.1 UC12.1 Brief

Actoren: Speler

Beschrijving:
De speler wil een berichtje naar de andere spelers sturen. Deze functie is alleen beschikbaar als er minstens 1 netwerkspeler aanwezig is. De speler typt zijn bericht in het tekstvak, en drukt op de knop verzenden of drukt op enter. Als de speler niet de server is, wordt het bericht naar de server gestuurd, die het dan doorstuurt naar alle andere spelers. Als de speler de server is, wordt het bericht onmiddellijk naar alle spelers gestuurd.

7.12.2 UC12.2 Fully Dressed

Primary Actor: Speler
Stakeholders and Interests:

Spelers: Willen een eenvoudige afhandeling van chatberichten.

Preconditions:

De applicatie is opgestart

Er is minstens 1 netwerkspeler aanwezig in het spel.

Postconditions:

Het chatbericht wordt afgeleverd bij alle netwerkspelers

Gebruikte sub use cases: geen

Cross references: FR 4.7, FR 4.8, FR 4.9

7.12.2.1 Basic Flow

Speler	*System*
1. Speler typt een chatbericht 2. Speler drukt op verzenden of drukt op enter	
	3. Systeem verzendt het bericht naar de server, ook al is deze zelf de server (FR 4.7) 4. Server verzendt bericht naar alle spelers (FR 4.8) 5. Systeem geeft het bericht weer (FR 4.9)

8 Systeem Sequentie Diagrammen

8.1 SSD1: UC 1 Spel Starten

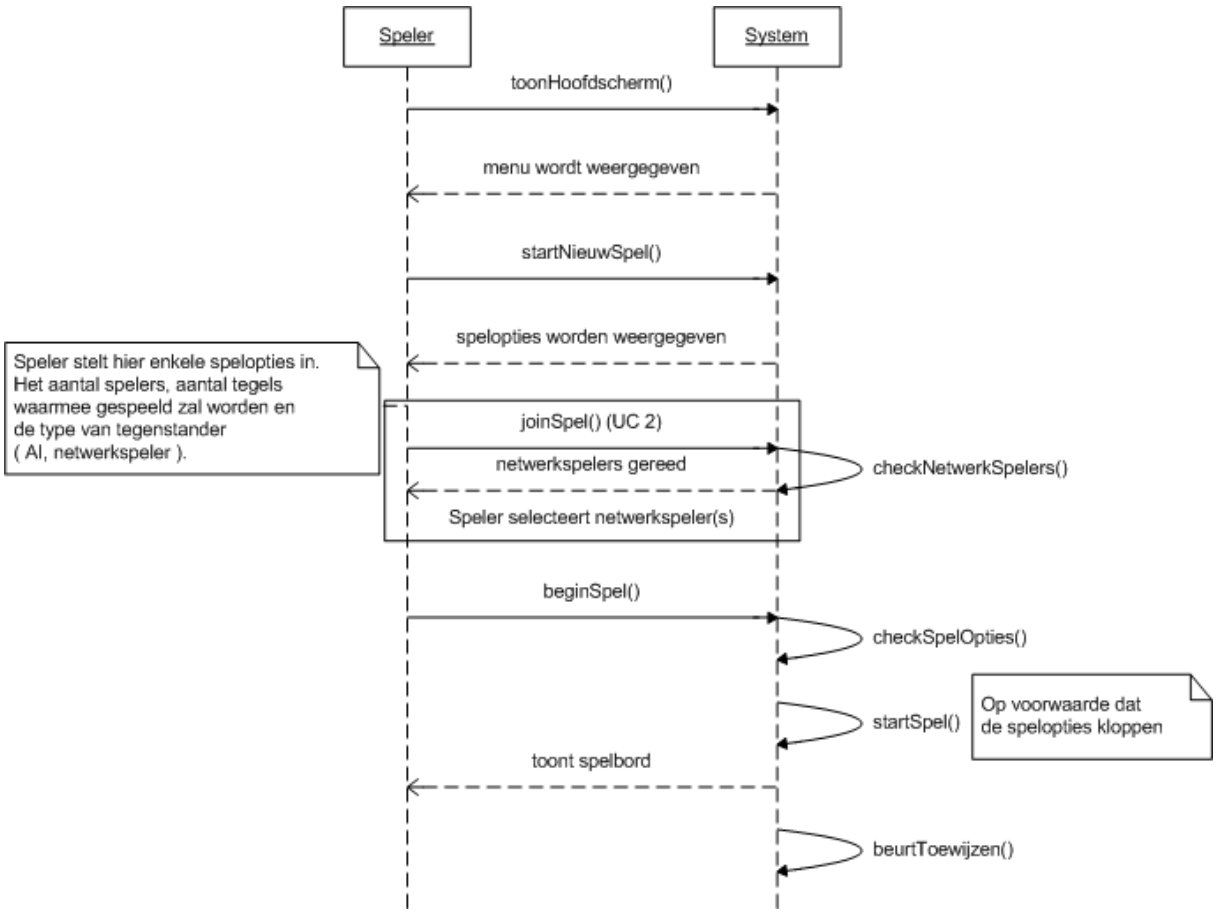


Figure 1 SSD 1 Spel Opstarten (UC 1)

8.2 SSD2: UC 2 Spel joinen

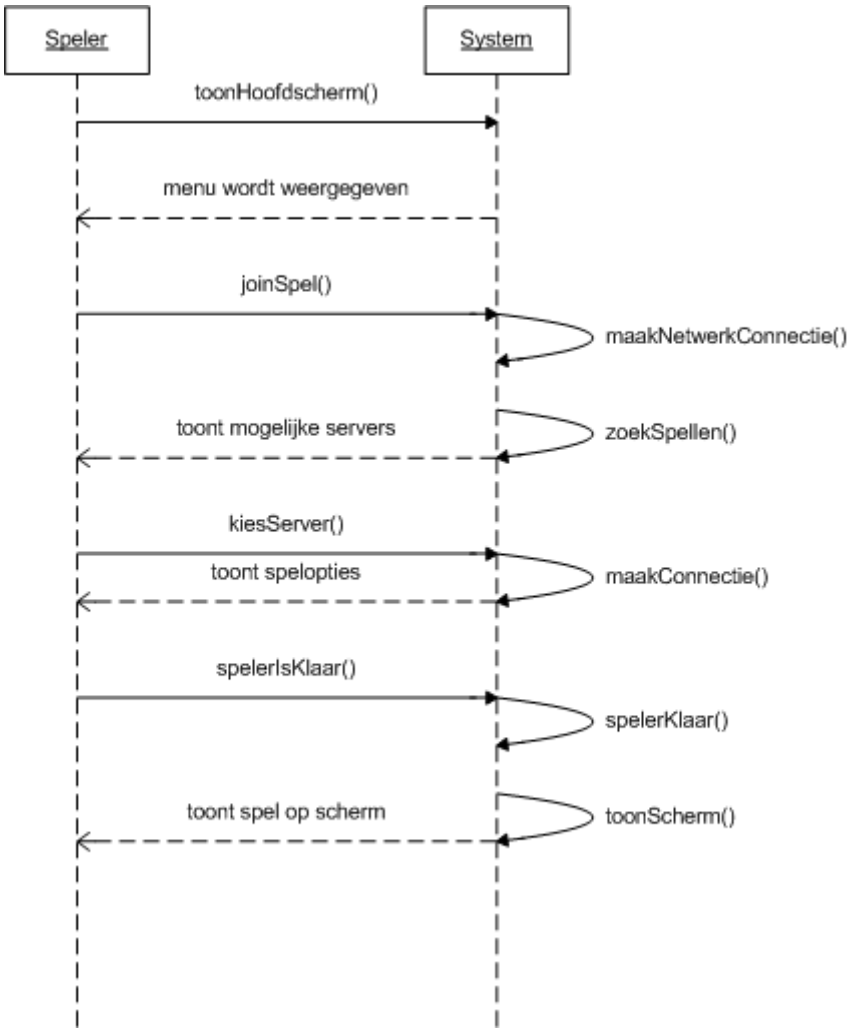


Figure 2 SSD2 Spel Joinen (UC 2)

8.3 SSD3: UC 3 Spelverloop

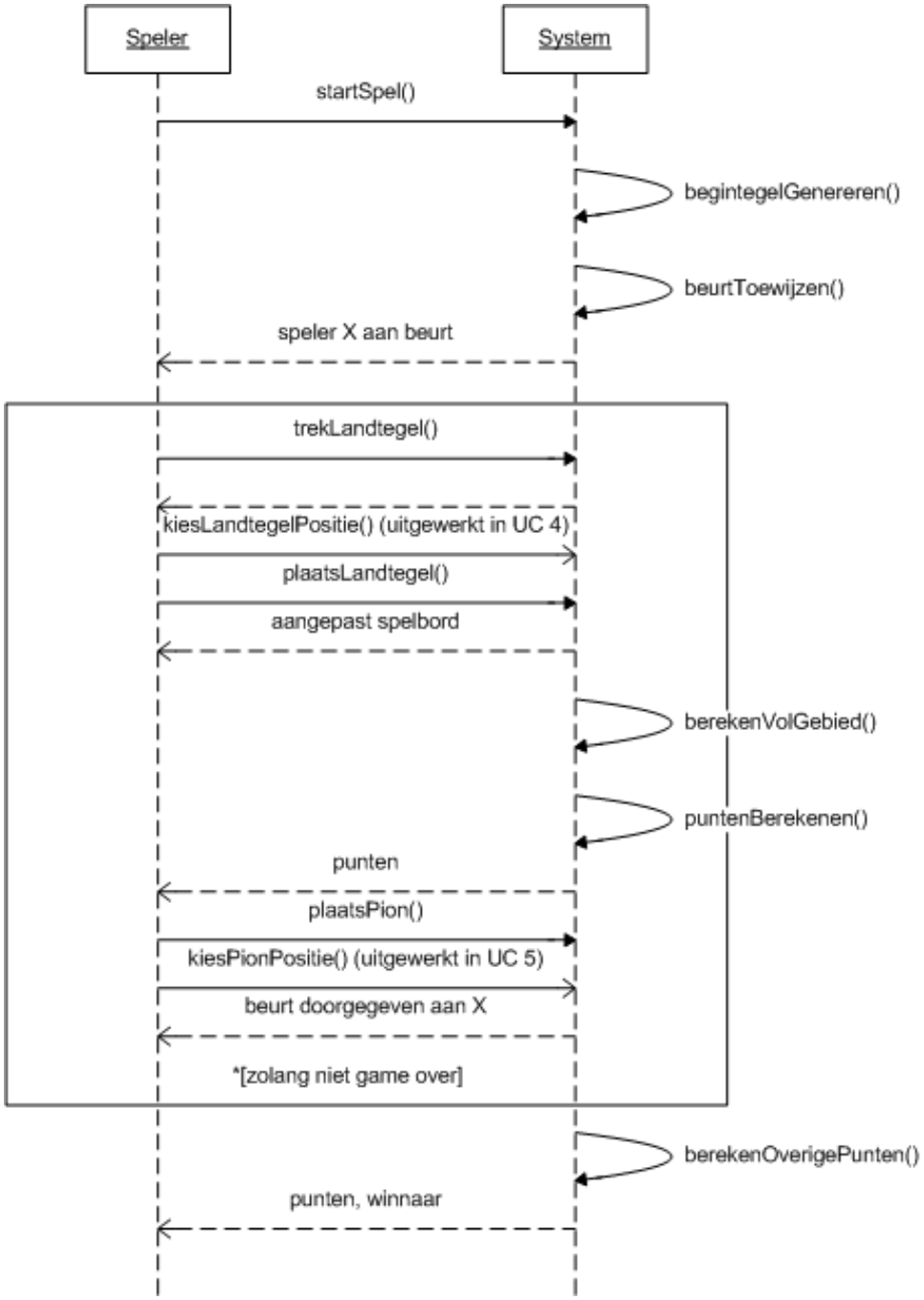


Figure 3 SSD3 Spelverloop (UC 3)

8.4 SSD4: UC4 een landtegelpositie kiezen (uitwerking van stap in UC3)

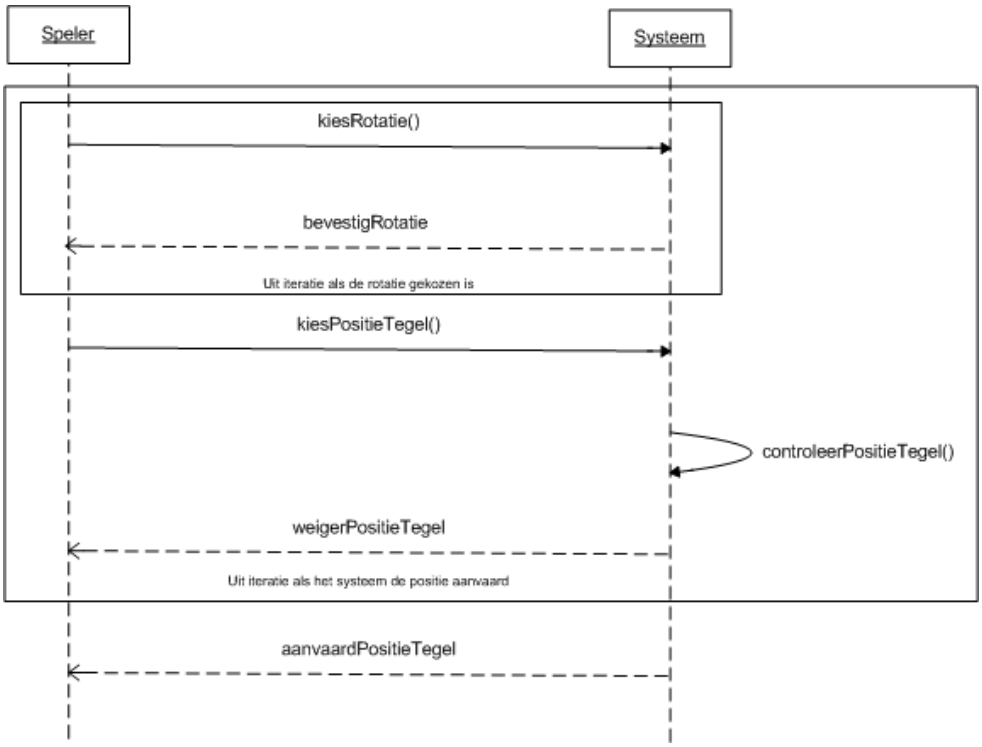


Figure 4 SSD4 een landtegel positie kiezen (UC 4)

8.5 SSD5: UC5 een pion positie kiezen(uitwerking van stap in UC3)

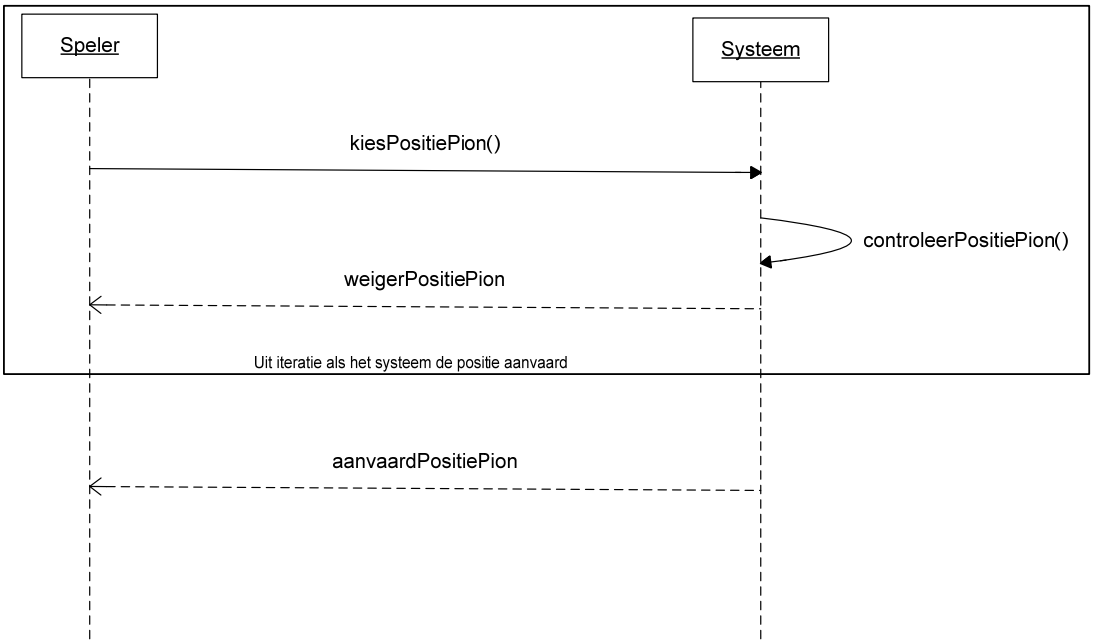


Figure 5 SSD5 een pion plaatsen (UC 5)

8.6 SSD6: UC8 Spel opslaan

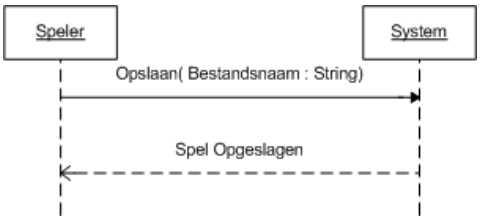


Figure 6 SSD6 Spel opslaan (UC8)

8.7 SSD7: UC9 Spel inladen

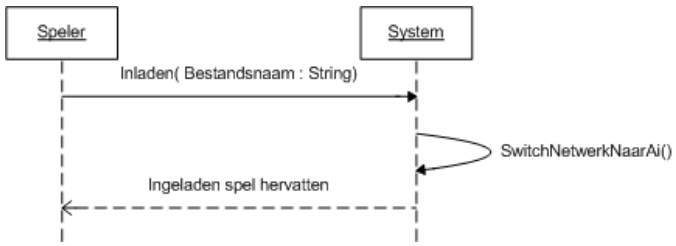


Figure 7 SSD7 Spel Inladen (UC 9)

8.8 SSD8: UC11 Spel afsluiten

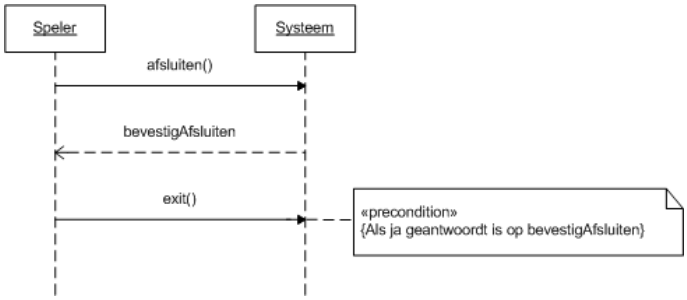


Figure 8 SSD8 Spel afsluiten (UC 11)

8.9 SSD9: UC12 Chatbericht versturen

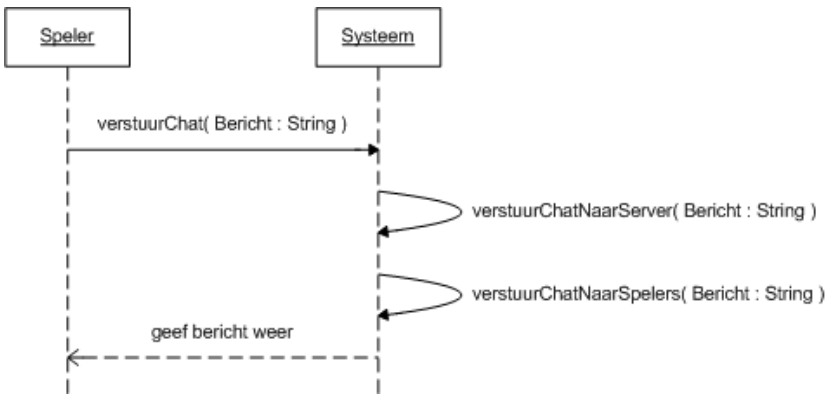


Figure 9 SSD9 Chatbericht versturen (UC 12)

8.10 SSD10: UC10 Spel opties

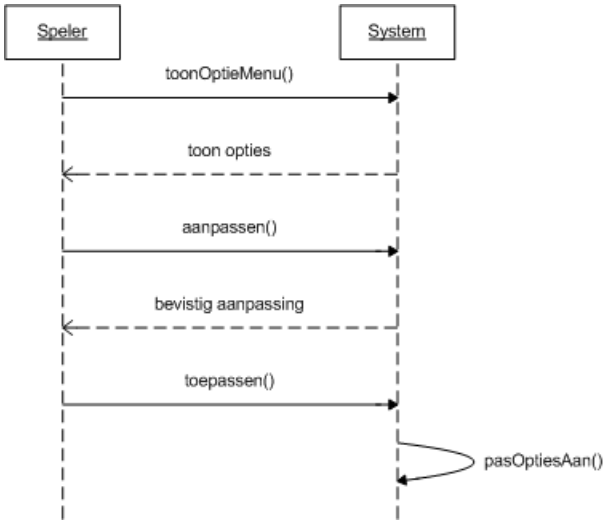


Figure 10 SSD10a Spel Opties (UC 10.2)

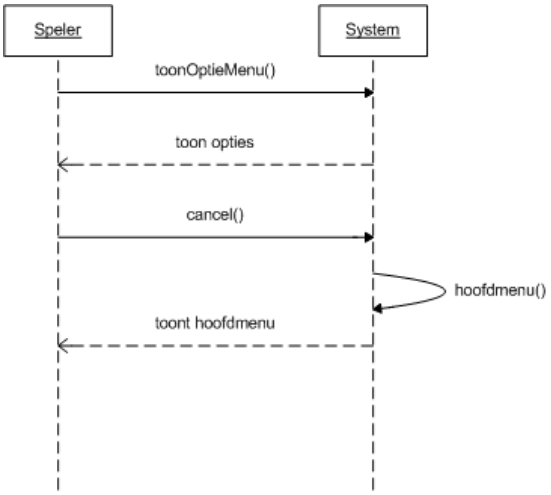


Figure 11 SSD10b Spel Opties Alternative Flow (UC 10.2a)

8.11 SSD11: UC6 Highscores registreren

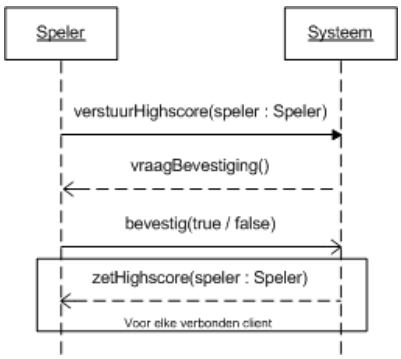


Figure 12 SSD11 Highscores registreren (UC 6)

8.12 SSD12: UC7 Highscores bekijken

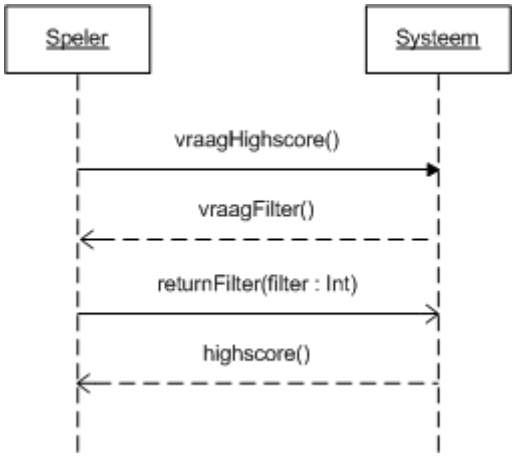


Figure 13 SSD12 Highscores bekijken (UC 7)

9 Contracten

9.1. Begintegel genereren

Naam	LayStartTile()
Cross References	SSD3
Pre-condities	- Al de nodige spellers zijn aanwezig - Leeg spelbord ingeladen
Post-condities	- Er ligt een field op het bord **Instance creation and deletion** - 1 Field-object aangemaakt (de begintegel) **Attribute modification** - In bord zijn er de coördinaten van de begintegel bijgekomen - Er zijn 3 eigenschappen voor de begintegel aangemaakt. **Associations formed and broken** - Bord heeft een verwijzing naar begintegel gekregen

9.2. Beurt aan de volgende speler toewijzen

Naam	NextPlayer()
Cross References	SSD3
Pre-condities	- Beurt van de huidige speler is afgelopen of er is nog geen beurt toegewezen - Er ligt minstens 1 field op het spelbord
Post-condities	**Attribute modification** - In Game is de huidige speler veranderd naar de nieuwe speler - Iterator van de Player is verplaatst naar de volgende speler

9.3. Trek landtegel

Naam	GiveField() : Field
Cross References	SSD3
Pre-condities	- De beurt is juist gegeven aan een speler
Post-condities	**Instance creation and deletion** - 1 Field-object aangemaakt **Attribute modification** - Er zijn 1, 2 of 3 eigenschappen aangemaakt in de field - Op de field is een rotatie ingesteld **Associations formed and broken** - Een verwijzing naar deze nieuwe Field wordt teruggegeven

9.4. Plaatsen van landtegel

Naam	LayField(t : Field) bool
Cross References	SSD3

Pre-condities	- De speler heeft een tegel ter beschikking en ingesteld met welke rotatie en op welke plaats hij deze wil plaatsen - De tegel kan minstens op 1 manier ergens geplaatst worden
Post-condities	**Attribute modification** - Board heeft een field-object bij gekregen - Een player zijn points zijn misschien veranderd **Associations formed and broken** - Verwijzing in board naar het Field-object

9.5. Plaatsen van pion

Naam	SetPion (l : Location)
Cross References	SSD3
Pre-condities	- De speler heeft een pion ter beschikking die hij kan leggen
Post-condities	**Instance creation and deletion** - 1 Pion-object aangemaakt **Attribute modification** - Pion locatie is toegevoegd - Pion area is toegevoegd - De nieuwe pion bevat nu de speler die de pion heeft toegevoegd **Associations formed and broken** - Bord heeft nieuwe verwijzing gekregen naar de nieuwe pion

9.6. Bereken overige punten

Naam	CalculateLastPoints()
Cross References	SSD3
Pre-condities	- Het bord ligt vol met alle landtegels - Het spel is dus afgelopen - Alle pionnen zijn geplaatst
Post-condities	- Winnaar bekend **Instance creation and deletion** - Arraylist van pionnen aangemaakt - Clone arraylist als backup van de pionnen arraylist - Arraylist aangemaakt voor pionnen die al berekend zijn **Attribute modification** - De punten bij de spelersobjecten zijn aangepast naar de nieuwe punten - Highscores van het object highscores zijn aangepast

9.7. Pionpositie kiezen

Naam	*choosePionPosition(p : Pion)*
Cross References	SSD5, SSD3
Pre-condities	- De speler is aan zet en heeft een tegel gelegd. - De speler is aan zet en heeft net een pion gekozen.
Post-condities	**Attribute modification** - het pion-bject werd bijgewerkt met de positie

9.8. Controleer positie pion

Naam	checkPionPosition(p: Pion):bool
Cross References	SSD5, SSD3
Pre-condities	- De speler is aan zet en heeft een tegel gelegd. - De speler is aan zet en heeft net een pionpositie gekozen.
Post-condities	**Instance creation and deletion** - 1 boolean-object aangemaakt **Attribute modification** - De uitkomst wordt opgeslaan in het bool-object.

9.9. Kies de rotatie voor een landtegel

Naam	chooseRotation(rotation:int, t:Field)
Cross References	SSD4, SSD3
Pre-condities	- De speler is aan beurt en moet nu net een tegel plaatsen. - Er is een tegel toegewezen aan de speler.
Post-condities	**Attribute modification** - het Field-object werd bijgewerkt met de rotatie.

9.10. Kies de positie voor een landtegel

Naam	chooseTlePosition(coords:Coords,, t:Field))
Cross References	SSD4, SSD3
Pre-condities	- De speler is aan beurt en moet nu net een tegel plaatsen. - Er is een tegel toegewezen aan de speler.
Post-condities	**Attribute modification** - Het Field-object werd bijgewerkt met de coördinaten.

9.11. Controleren van de positie van een tegel

Naam	checkTilePosition(t : Tile):bool
Cross References	SSD4, SSD3
Pre-condities	- De speler is aan de beurt, is een tegel toegewezen en heeft een positie gekozen
Post-condities	**Instance creation and deletion** - 1 bool-object aangemaakt **Attribute modification** - De uitkomst wordt toegekend aan het bool-object

9.12. Opslaan

Naam	Save(file : String)
Cross References	SSD6
Pre-condities	- De speler is aan de beurt, is een tegel toegewezen en heeft een positie

	gekozen.
Post-condities	**Instance creation and deletion** - File-object werd aangemaakt en achteraf terug verwijderd. **Attribute modification** - Een filenaam werd toegekend aan het file-object - File bevat een string die het gehele spelbord en spelers beschrijft **Associations formed and broken** - Vanuit het File-object werd er verwezen naar het Game, zo kan voor elk deel van het Game een toString functie worden aangeroepen, deze is achteraf terug verwijderd.

9.13. Inladen

Naam	Load(file : String)
Cross References	SSD7
Pre-condities	- Applicatie is opgestart en speler is van plan om een spel in te laden
Post-condities	**Instance creation and deletion** - File-object werd aangemaakt en achteraf terug verwijderd. **Attribute modification** - Spelerobjecten zijn opgevuld met de nieuwe spelers - Bord object is aangepast met het nieuwe spelbord **Associations formed and broken** - Game heeft een verwijzing naar het aangemaakte file-object gehad, deze wordt ook terug verwijderd

9.14. Applicatie afsluiten

Naam	exit()
Cross References	SSD8
Pre-condities	- De applicatie is opgestart
Post-condities	- De applicatie is afgesloten. **Instance creation and deletion** - Alle objecten werden verwijderd (Program, frmMain, Game, elke Player, Board, elk Field) **Associations formed and broken** - Alle associaties naar de objecten worden verwijderd

9.15. Bericht versturen

Naam	Chat(message : String, fromNetwork:Bool)
Cross References	SSD9
Pre-condities	- De applicatie is opgestart - Er is minstens 1 netwerkspeler aanwezig
Post-condities	**Attribute modifications** - Het netwerk object heeft een bericht om te versturen

9.16. Bericht versturen naar alle spelers (basic flow)

Naam	sendMessagetoPlayers(Message: String)
Cross References	SSD9
Pre-condities	- De applicatie is opgestart - Er is minstens 1 netwerkspeler aanwezig - De server heeft een bericht ontvangen
Post-condities	- Het bericht is verstuurd naar alle spelers

9.17. Nieuw spel starten

Naam	startNewGame()
Cross References	SSD1
Pre-condities	- Hoofdmenu is zichtbaar - Gebruiker klikt op nieuw spel
Post-condities	- Spel is ingesteld en kan nu starten **Instance creation and deletion** - Lijst van Playersobjecten - Bord object - Lijst van Field objecten in bord **Attribute modifications** - Spelertypes, spelernamen, pionnen - IP's van netwerkspelers **Associations formed and broken** - Game heeft verwijzing naar de playerslijst gekregen - Game heeft verwijzing naar Board gekregen - Board heeft verwijzingen naar de Fields gekregen

9.19. Start Spel

Naam	startGame()
Cross References	SSD1, SSD3
Pre-condities	- Host heeft het spel ingesteld - Alle spelers zijn gereed
Post-condities	**Instance creation and deletion** - Spelbord wordt aangemaakt **Attribute modifications** - Players toegevoegd aan iteratielijst - Kleur, pion en naam van spelers - IP en poort wordt opgegeven aan de netwerkspelers

9.20. Spel joinen

Naam	joinGame ()
Cross References	SSD2
Pre-condities	- Client is verbonden met een netwerk
Post-condities	**Instance creation and deletion**

	- Netwerkconnectie gemaakt met lokaal netwerk

9.21. Connectie maken met host

Naam	makeConnection()
Cross References	SSD2
Pre-condities	- Client is verbonden met een netwerk - Client heeft een host geselecteerd uit de lijst
Post-condities	- Spelgegevens worden uitgewisseld **Instance creation and deletion** - NetworkPlayer aangemaakt - TCP object aangemaakt - Thread aangemaakt voor de networkplayer **Attribute modifications** - IP en Poort van networkplayer opgevuld

9.22. Optie-menu tonen

Naam	ShowOptions ()
Cross References	SSD10
Pre-condities	- Speler klikt op spelopties in hoofdmenu
Post-condities	**Instance creation and deletion** - Optie menu is weergegeven **Attribute modifications** - Vorige staat is opgeslagen

9.23. Opties aanpassen

Naam	ChangeOptions ()
Cross References	SSD10
Pre-condities	- De huidige staat is opgeslagen
Post-condities	**Attribute Modification** - Resolutie is aangepast - Achtergrondgeluid is aangepast - Geluideffecten zijn aangepast

9.24. Highscores naar systeem sturen

Naam	sendHighscore(player : Player, score: Int)
Cross References	SSD11
Pre-condities	- Het spel moet begonnen zijn - Er moet een highscore beschikbaar zijn
Post-condities	- Nieuwe highscores zijn verzonden naar clients

9.25. Highscore opslaan bij gebruiker

Naam	saveHighscore()

Cross References	SSD11
Pre-condities	- Het systeem moet bevestiging gekregen hebben om te mogen registreren
Post-condities	**Instance creation and deletion** - Er is een Player object aangemaakt voor degene die gewonnen is - Er is een Player object aangemaakt voor degene die laatste is - Bool object aangemaakt - Player objecten later terug verwijderd **Attribute Modification** - De highscore is bijgewerkt voor speler bij elke speler verbonden in het netwerk. - Lijst van highscores is ingelezen uit de highscore file

9.26. Opvragen Highscores

Naam	askHighscore()
Cross References	SSD12
Pre-condities	- Er moeten highscores van vorige spellen opgeslagen zijn - Dit bestand moet toegankelijk zijn
Post-condities	- Highscore object teruggegeven

9.27. Teruggeven filter aan systeem

Naam	changeFilter(sortBy:Enum, textb:TextBox)
Cross References	SSD12
Pre-condities	- Highscores moeten aangevraagd zijn
Post-condities	**Attribute Modification** - In de Highscore-object is manier van sorteren aangepast

10 Interactiediagrammen

10.1 LayStartTile (Contract 9.1)

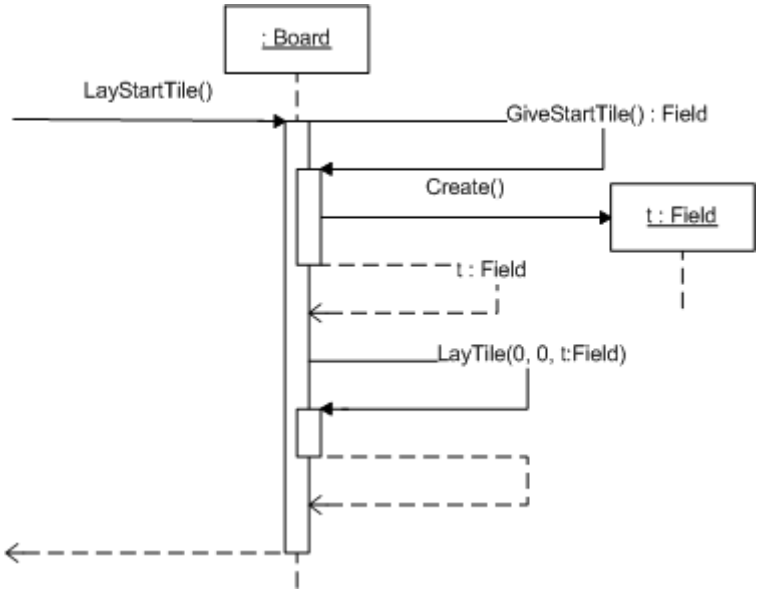


Figure 14 LayStartTile (Contract 9.1)

10.2 NextPlayer (Contract 9.2)

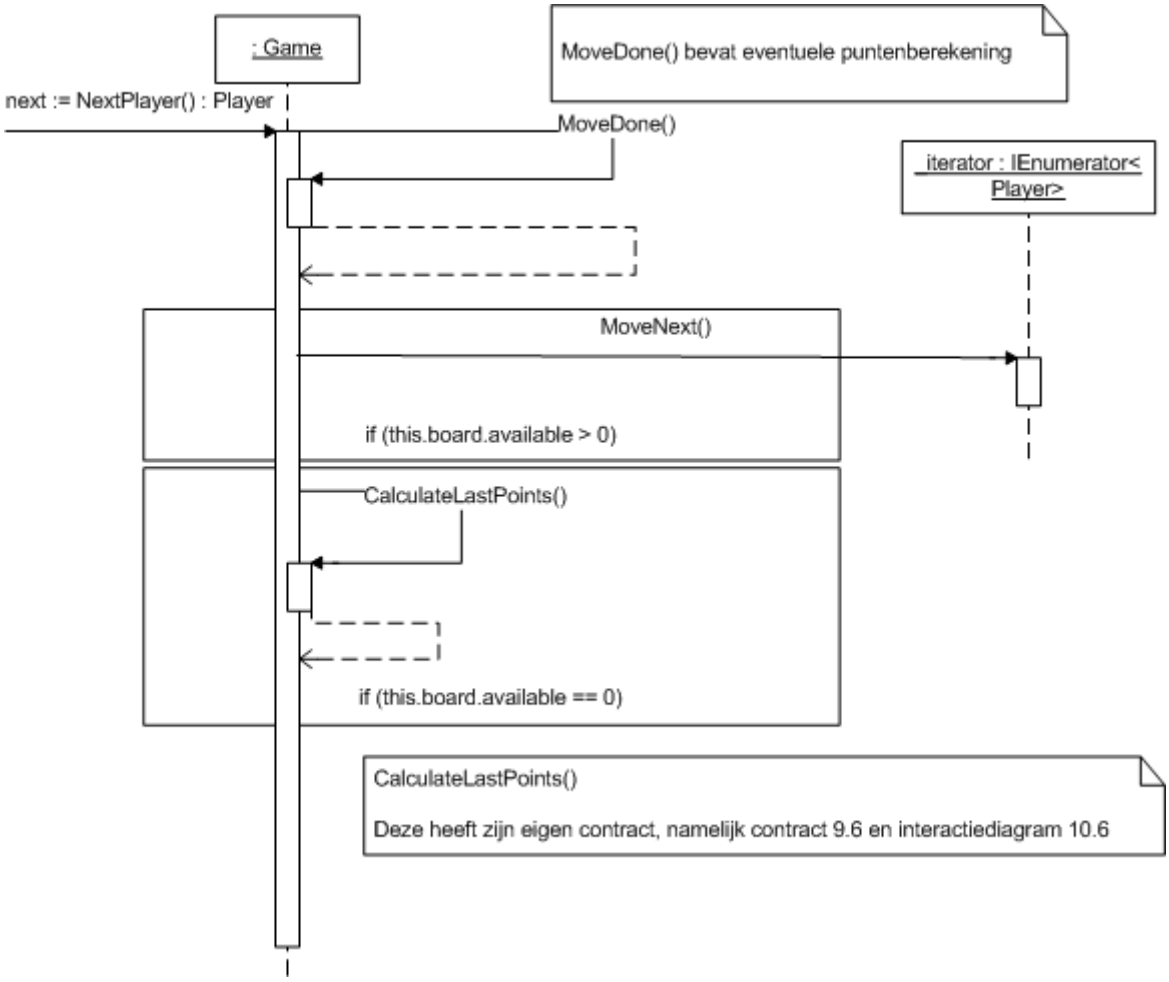


Figure 15 Nextplayer (Contract 9.2)

10.3 GiveTile (Contract 9.3)

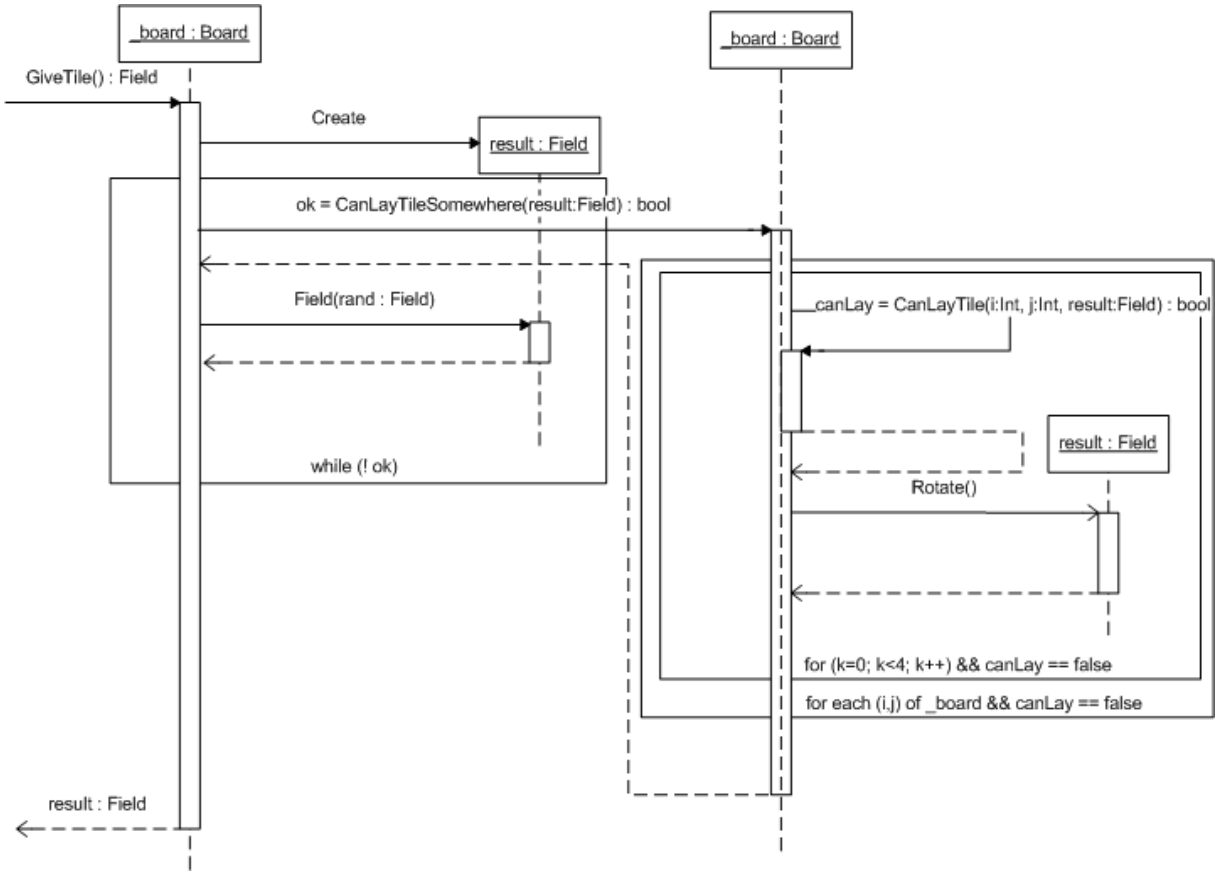


Figure 16 GiveTile (Contract 9.3)

10.4 LayTile (Contract 9.4)

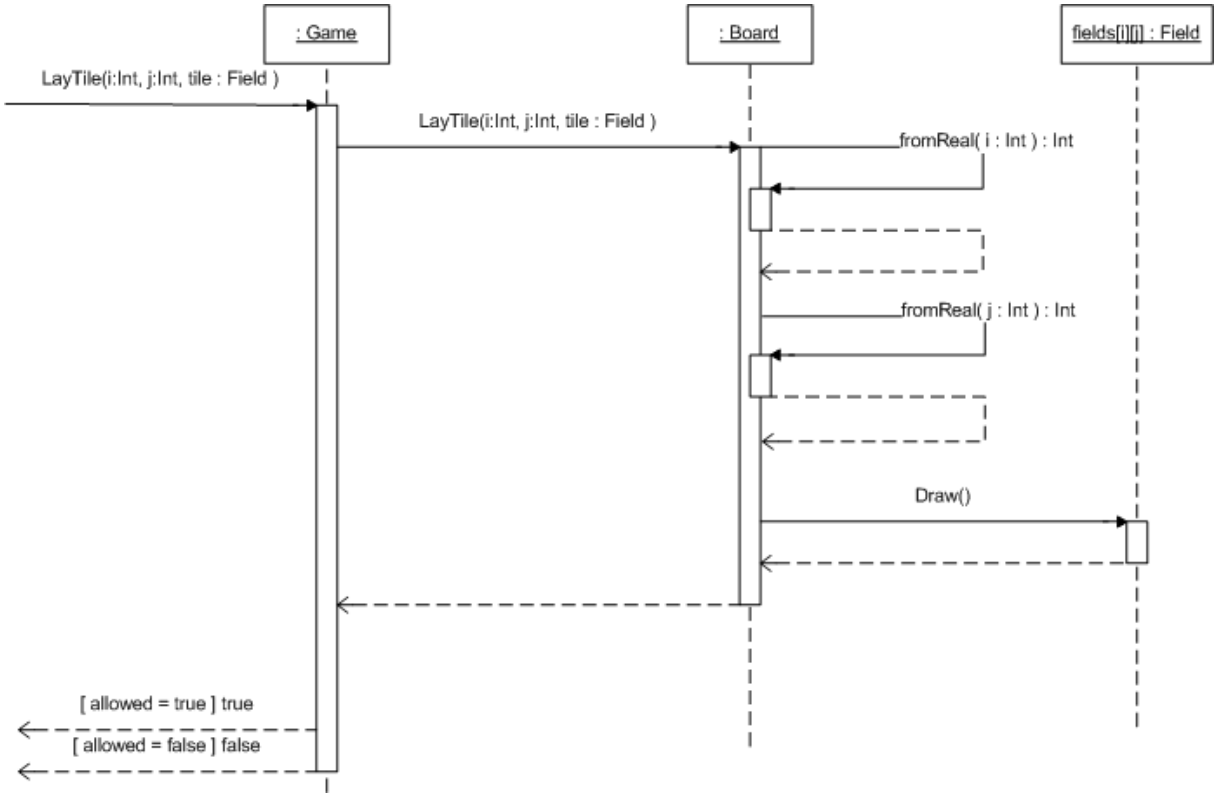


Figure 17 LayTile (Contract 9.4)

10.5 SetPion (Contract 9.5)

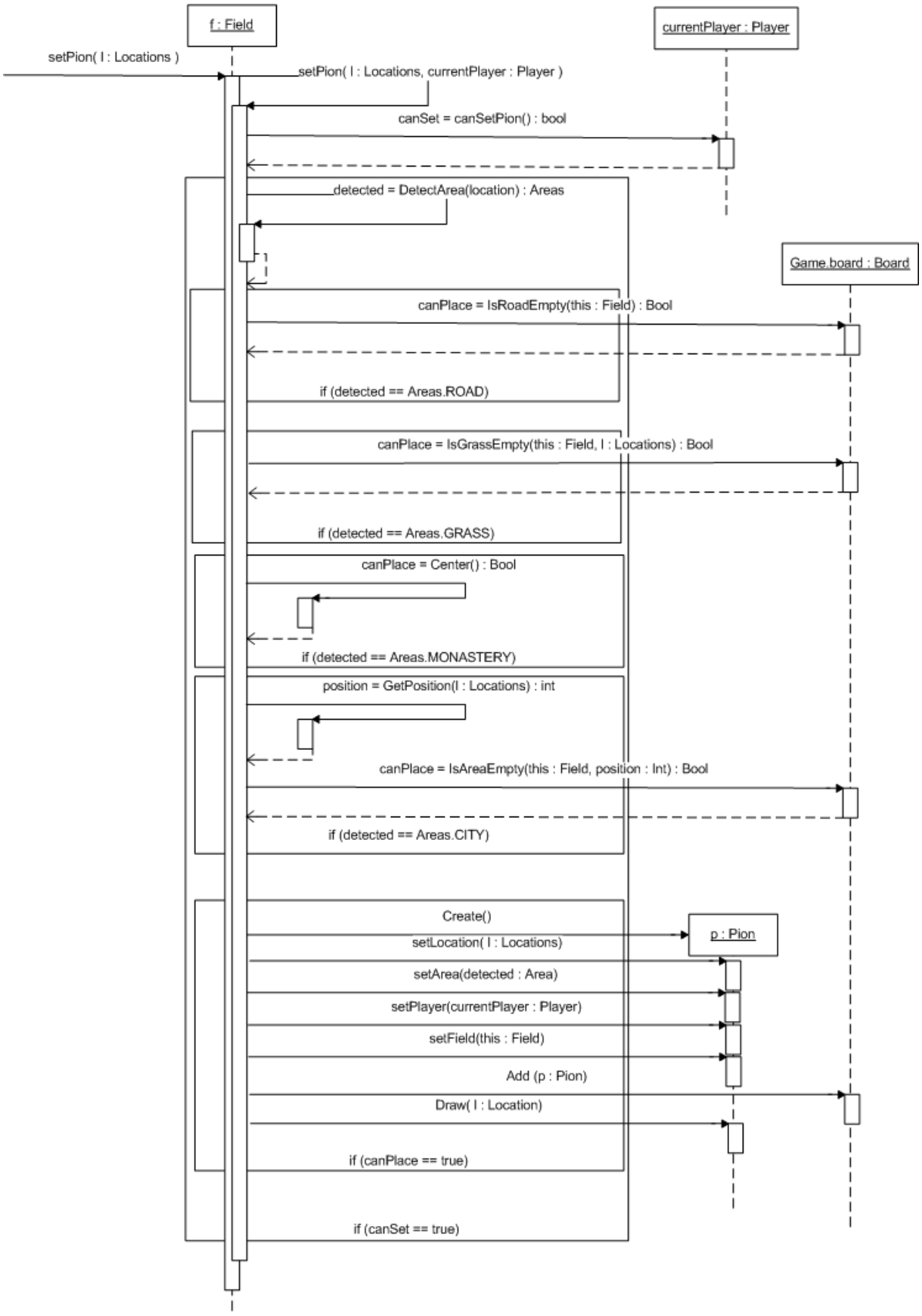


Figure 18 LayPion (Contract 9.5)

10.6 CalculateLastPoints (Contract 9.6)

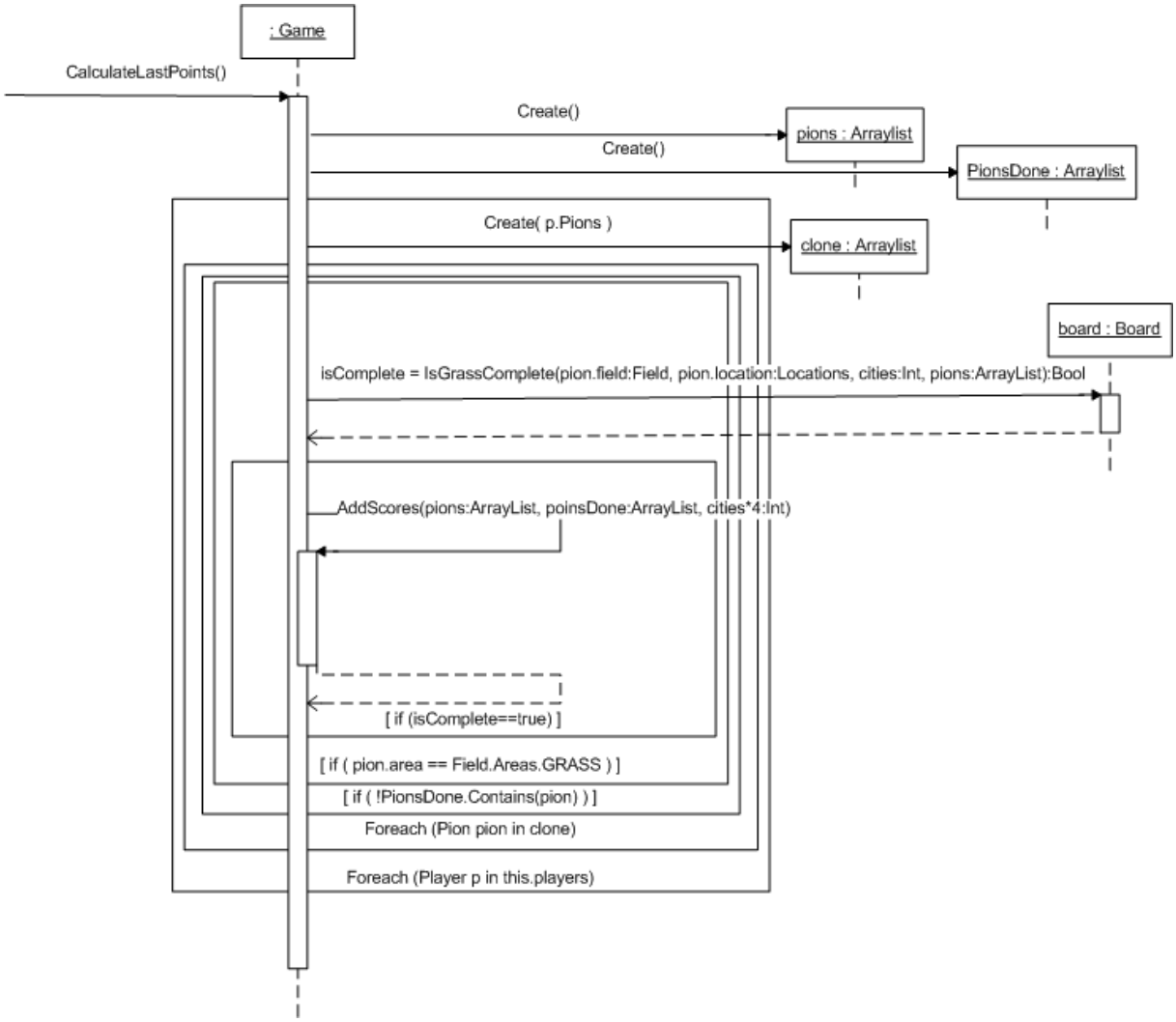


Figure 19 CalculateLastPoints (Contract 9.6)

10.7 ChoosePionPosition(Contract 9.7)

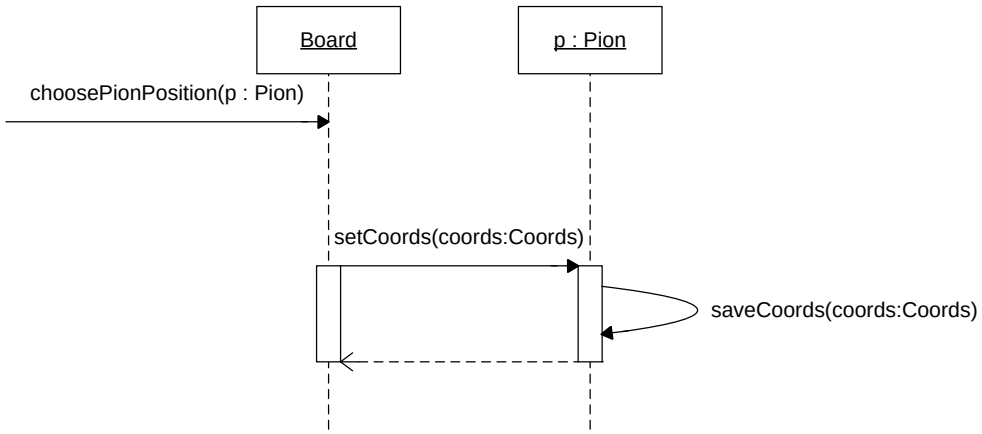


Figure 22 ChoosePionPosition (Contract 9.7)

10.8 CheckPionPosition(Contract 9.8)

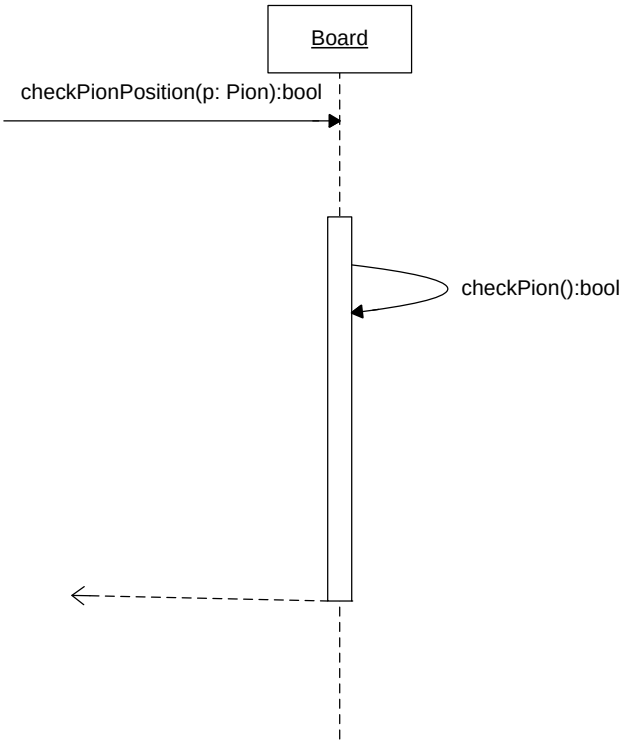


Figure 20 CheckPionPosition (Contract 9.8)

10.9 ChooseRotation(Contract 9.9)

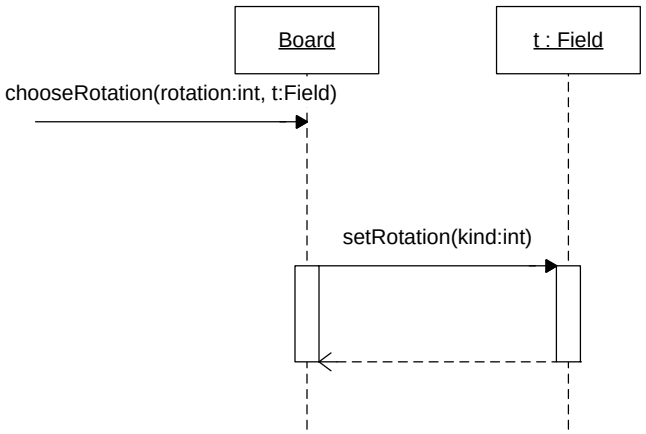


Figure 24 ChooseRotation (Contract 9.9)

10.10ChooseTilePosition(Contract 9.10)

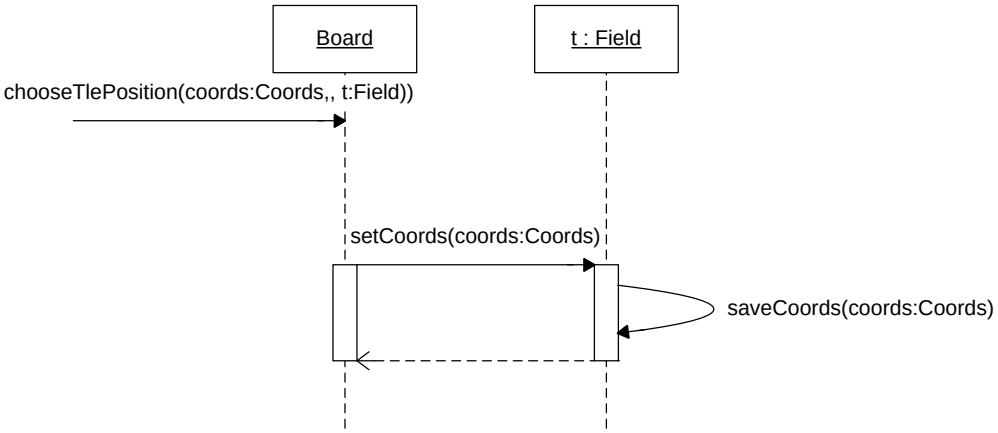


Figure 21 ChooseFieldPosition (Contract 9.10)

10.11CheckTilePosition(Contract 9.11)

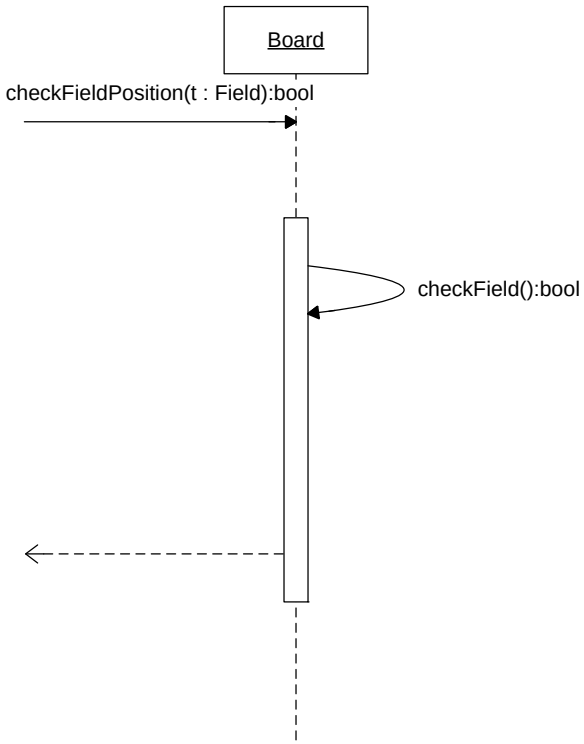
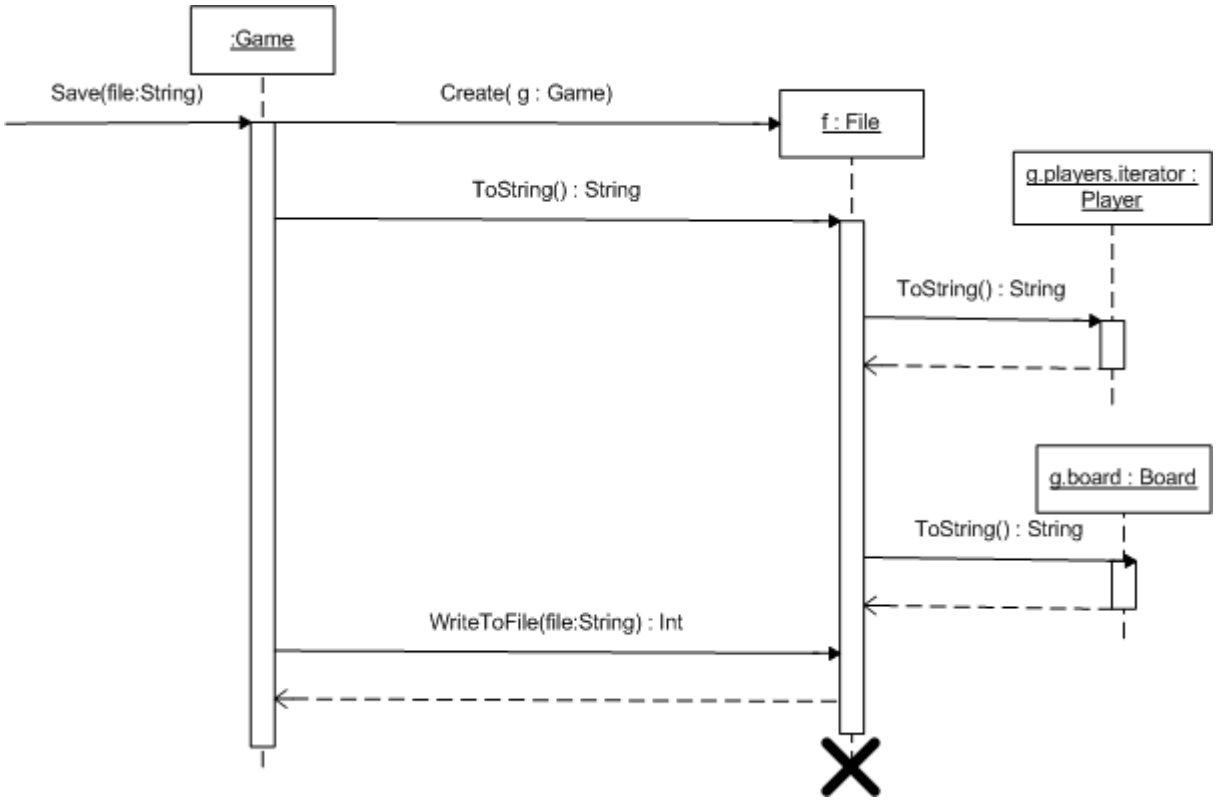
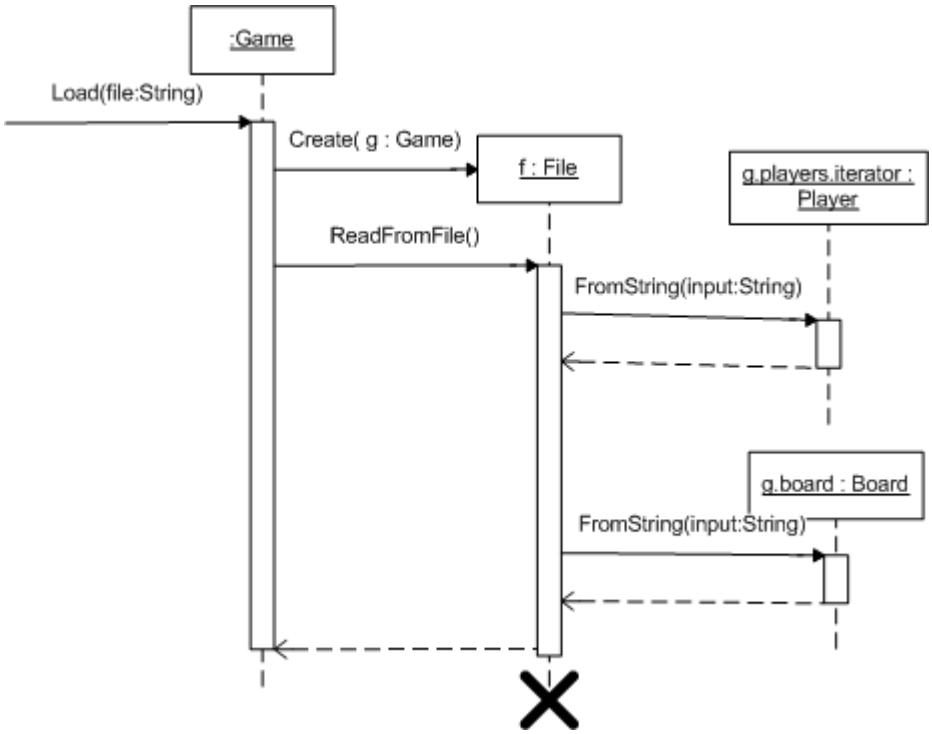


Figure 26 CheckFieldPosition(Contract 9.11)

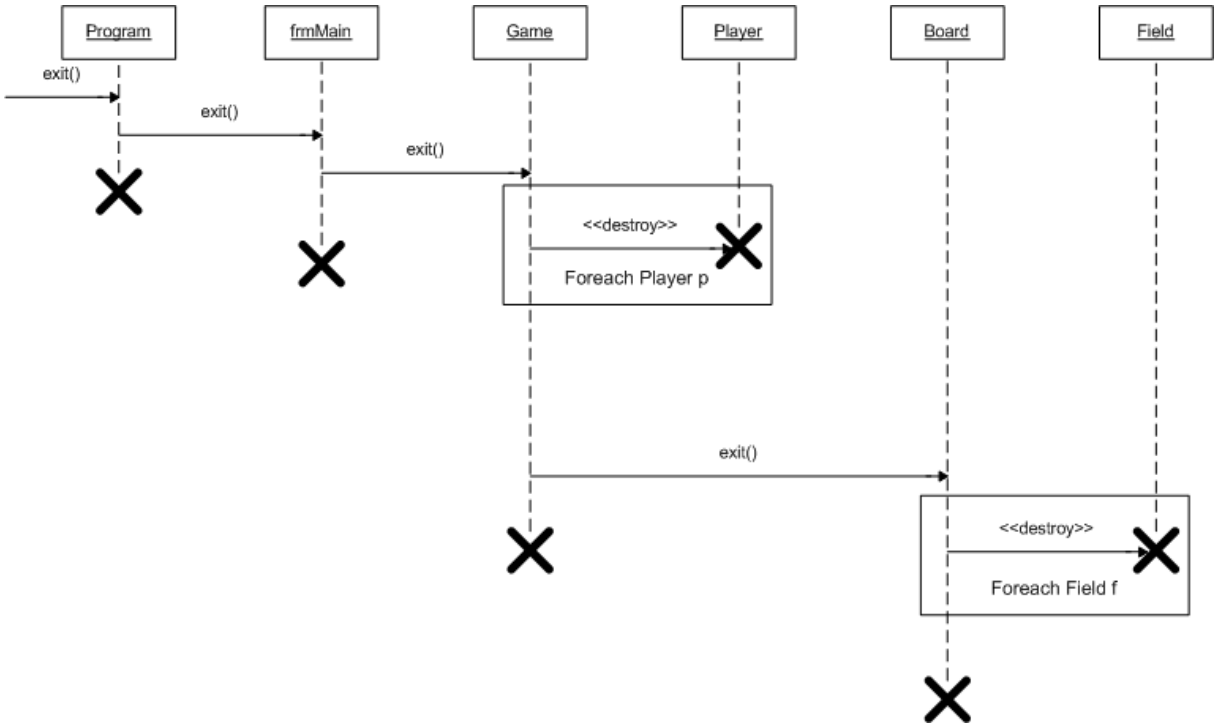
10.12Save (Contract 9.12)



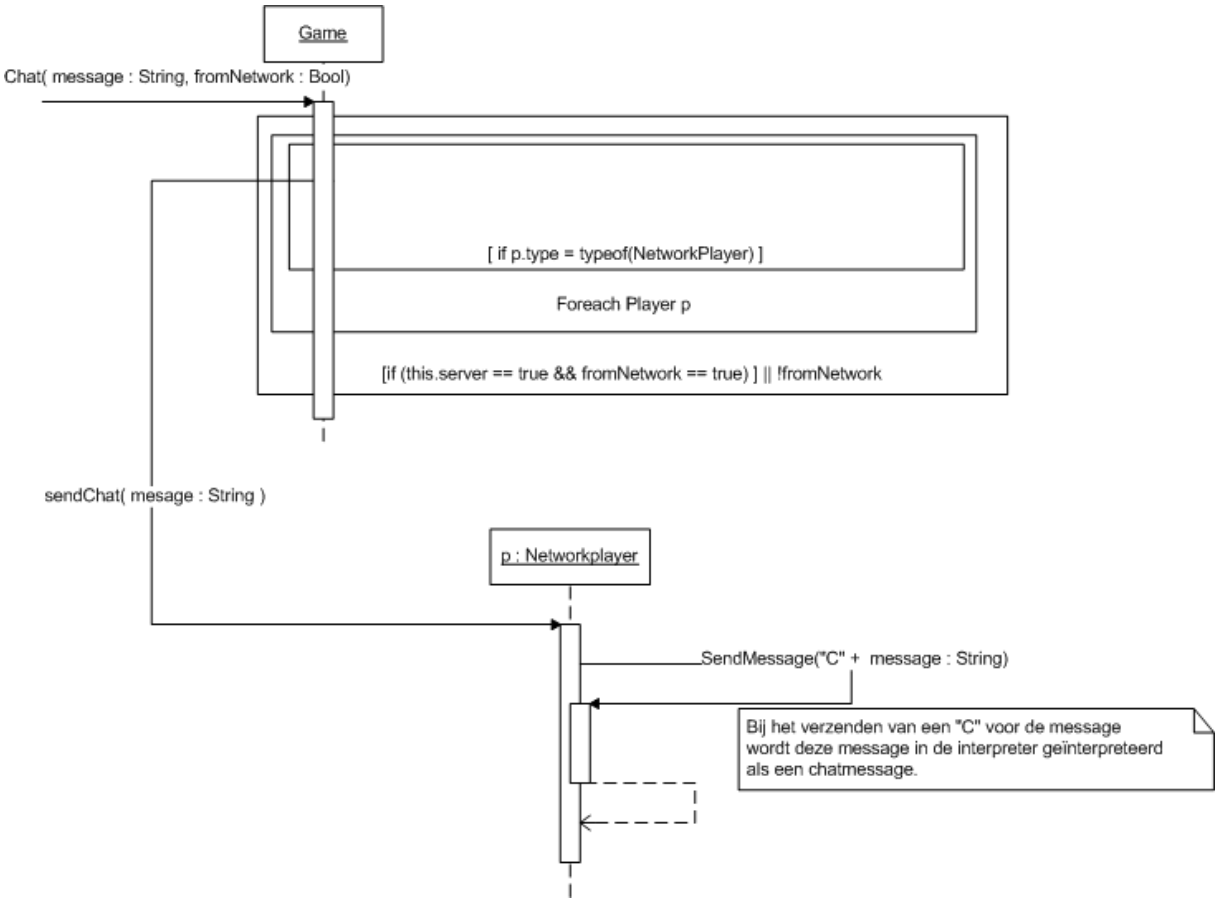
10.13Load (Contract 9.13)



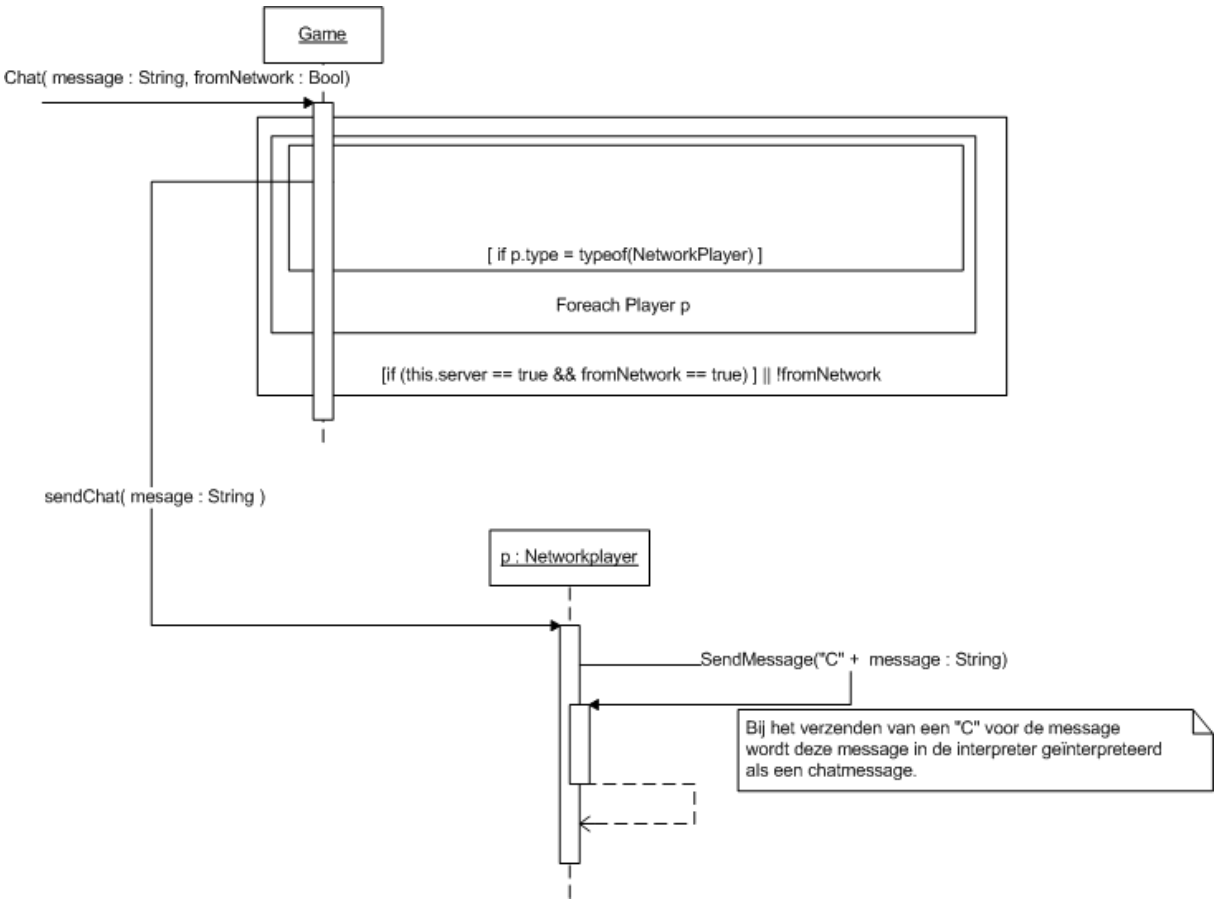
10.14exit(Contract 9.14)



10.15sendMessage(Contract 9.15)

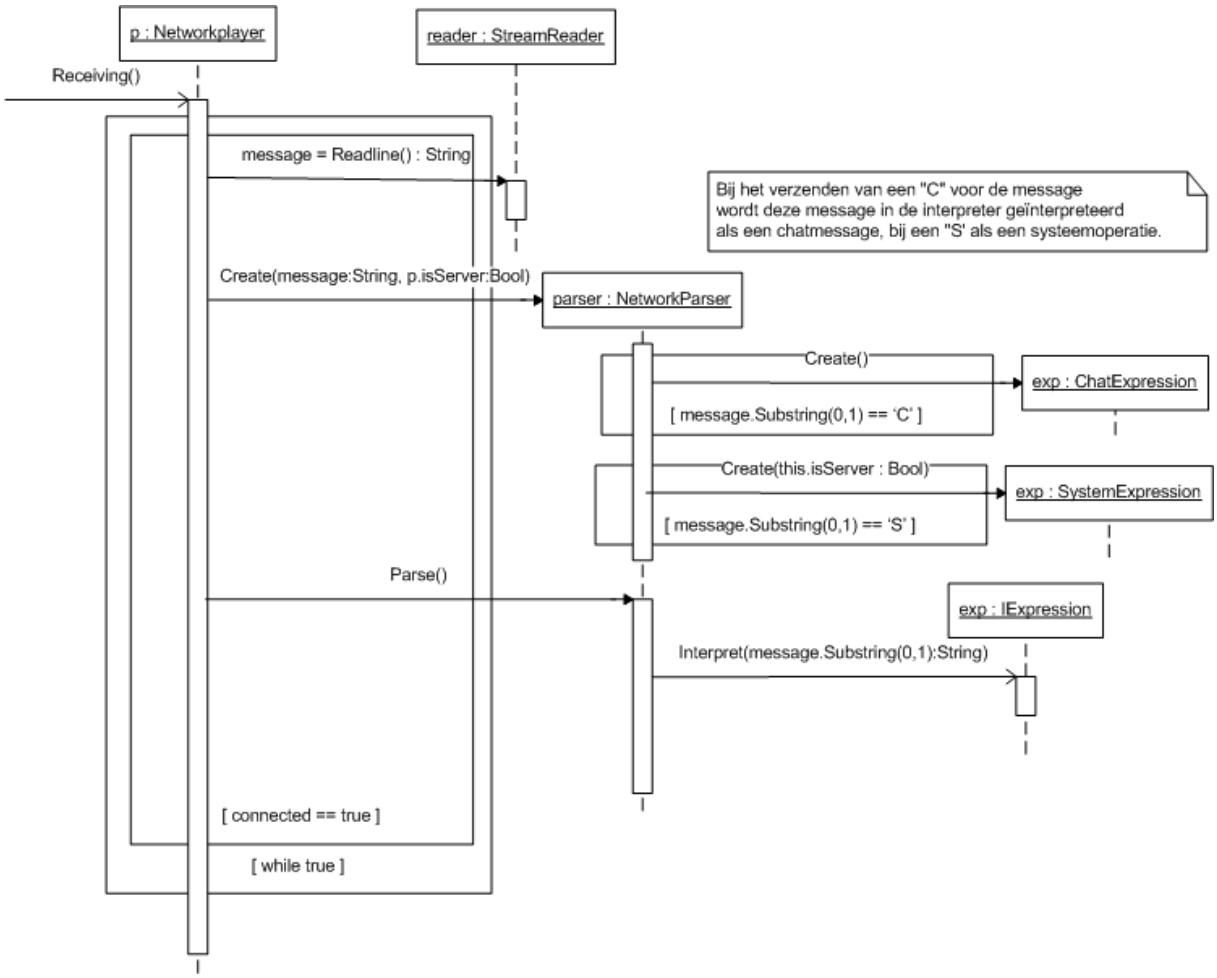


10.16sendMessageToPlayers(Contract 9.16)

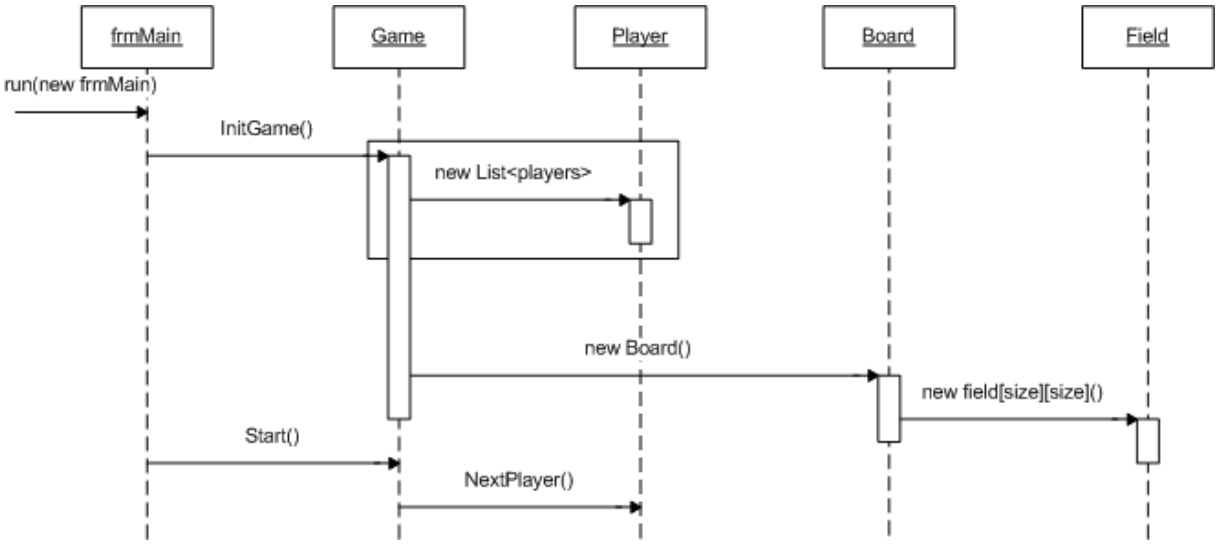


Deze is hetzelfde als de vorige omdat in de implementatie dit op hetzelfde neer komt.

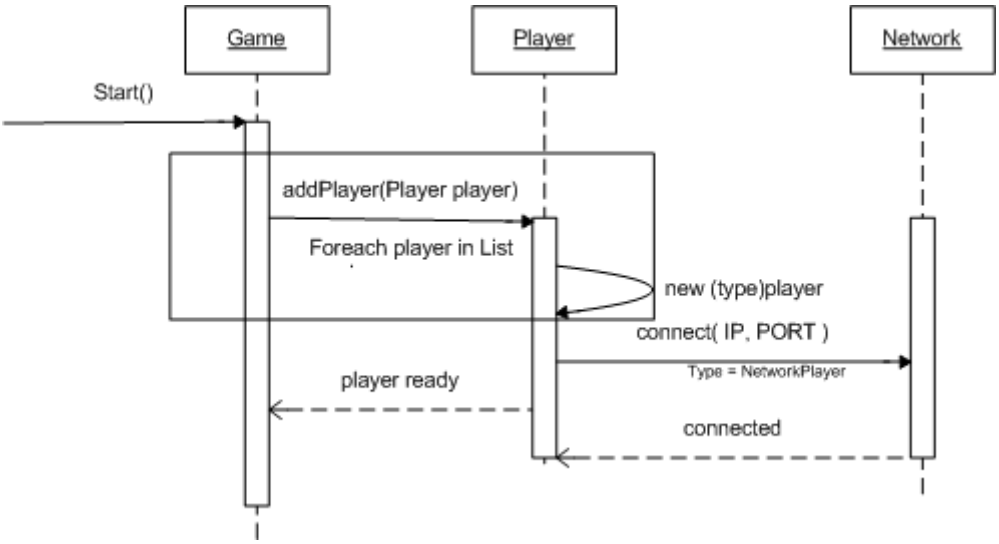
De receive van messages zijn (hetzij chatberichten of systeemberichten) in de analyse niet gemodelleerd omdat deze in aparte threads uitgevoerd worden. Hieronder is toch een interactiediagram ervan wegens er gebruik wordt gemaakt van de interpreter design pattern:



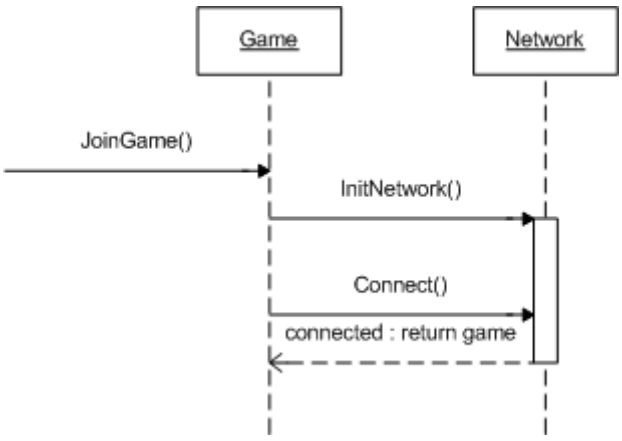
10.17startNewGame(Contract 9.17)



10.19StartGame(Contract 9.19)

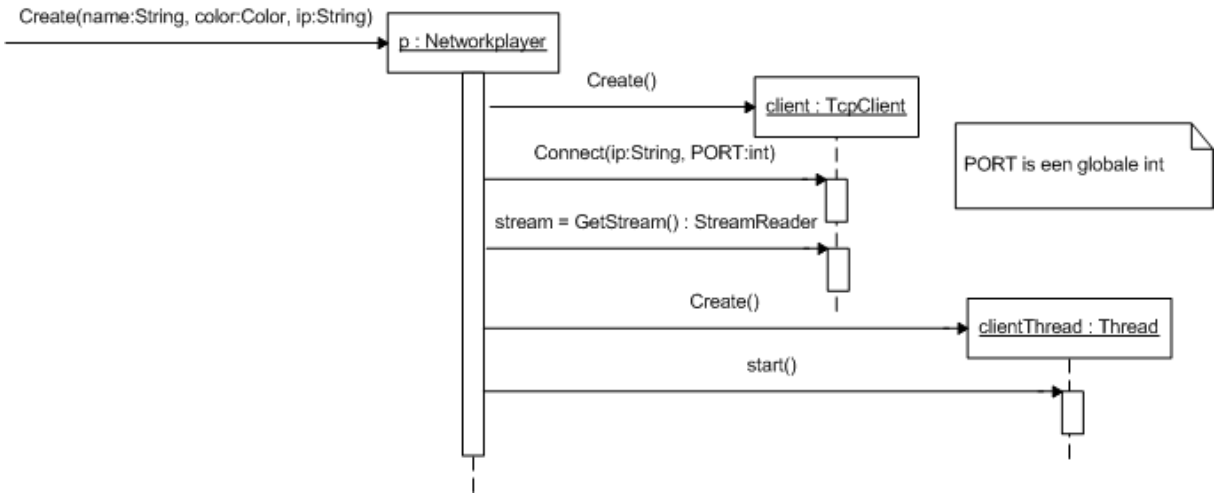


10.20JoinGame(Contract 9.20)

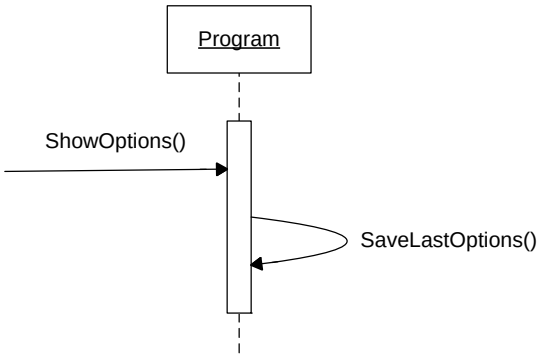


10.21makeConnection(Contract 9.21)

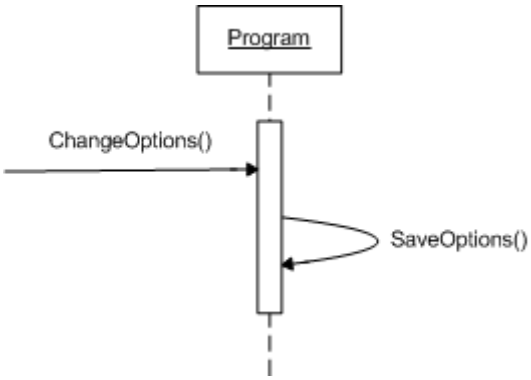
Deze is lichtjes aangepast in de implementatie. De connectie wordt nu direct uitgevoerd bij de creatie van een netwerkspeler. Het ziet er nu als volgt uit:



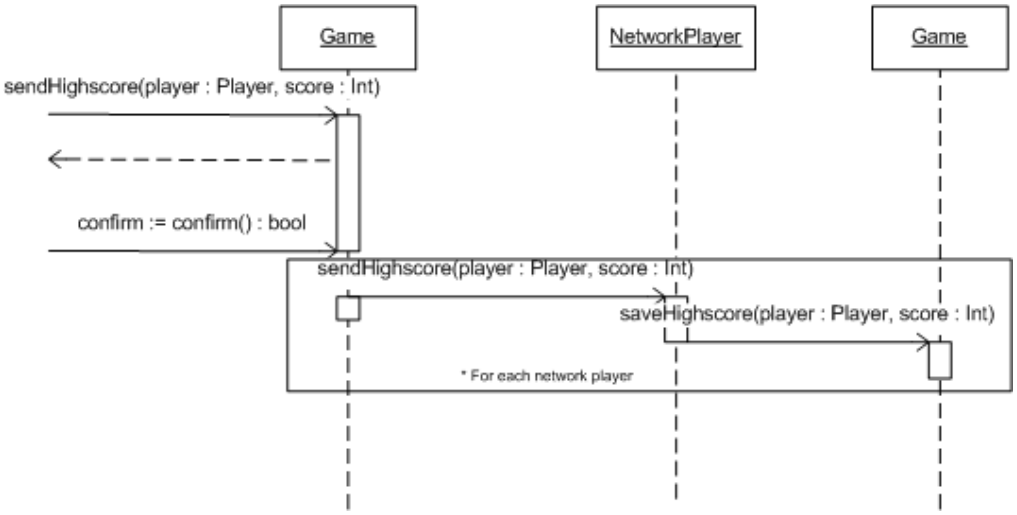
10.22showOptions(Contract 9.22)



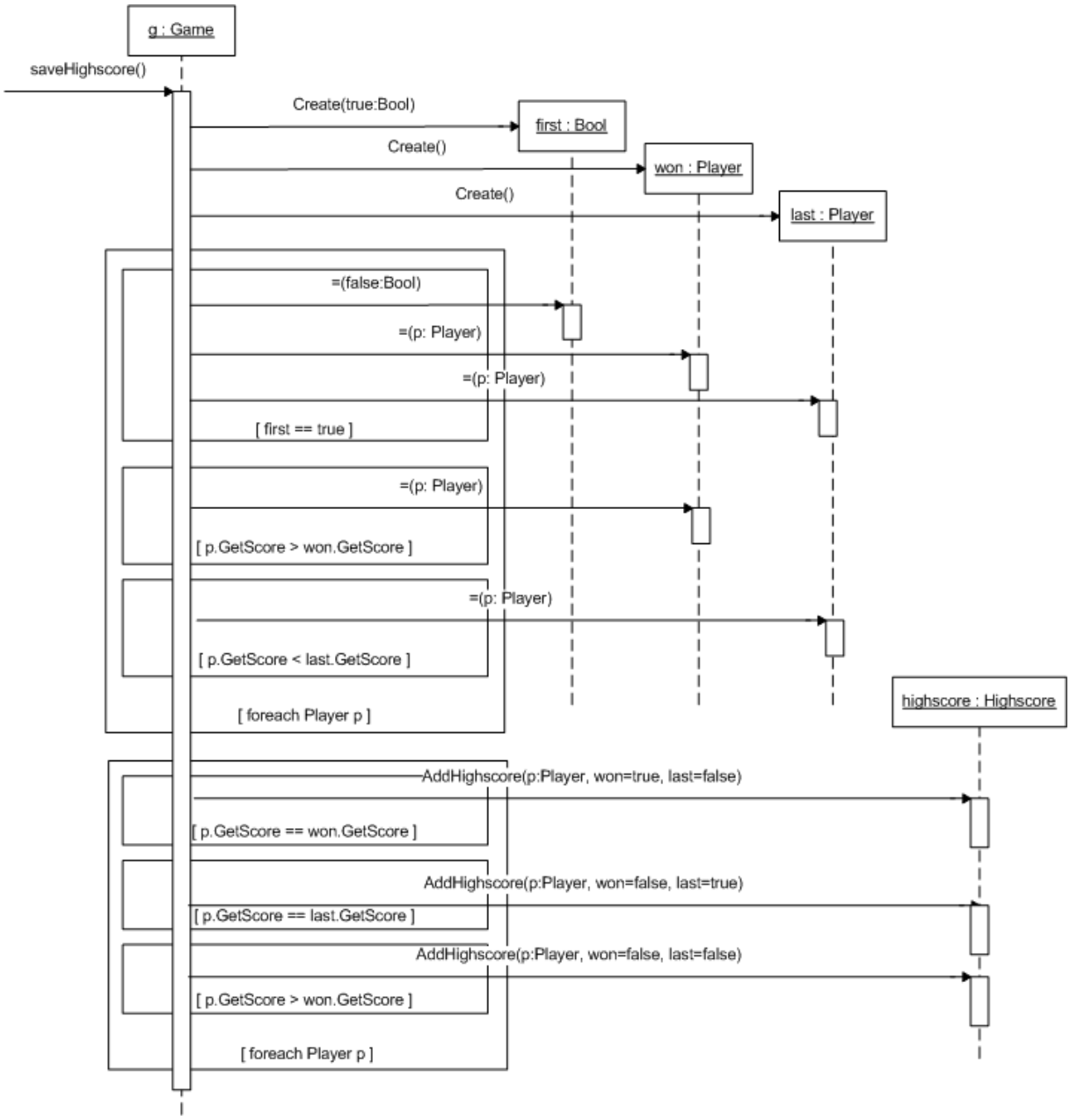
10.23changeOptions(Contract 9.23)



10.24sendHighscore(Contract 9.24)

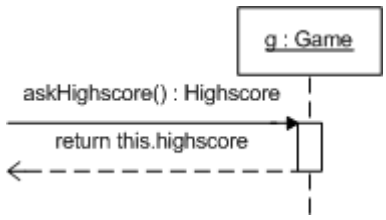


10.25saveHighscore(Contract 9.25)

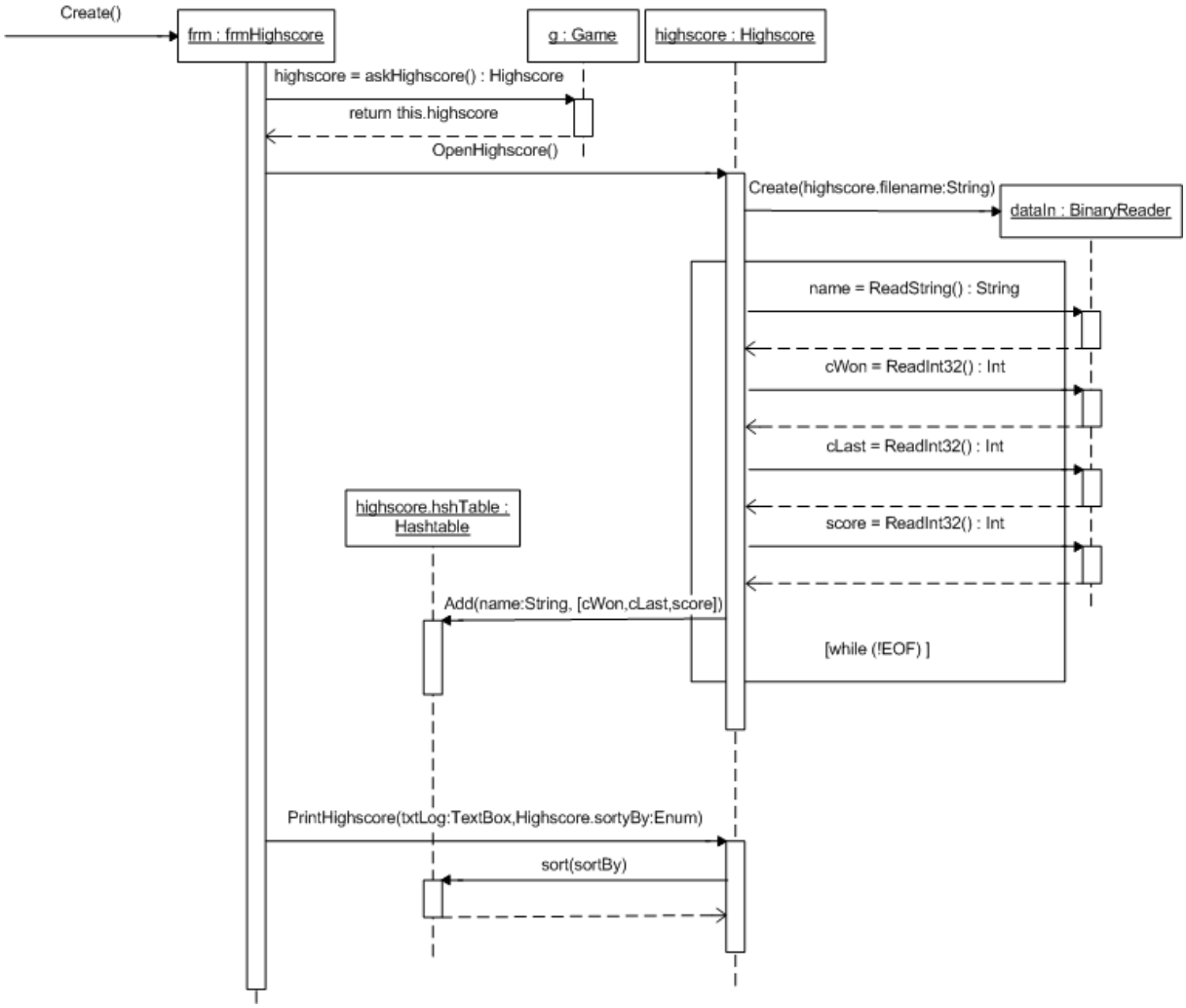


AddHighscore is niet verder uitgediept omdat deze enkel een container bevat van een hashtabel.

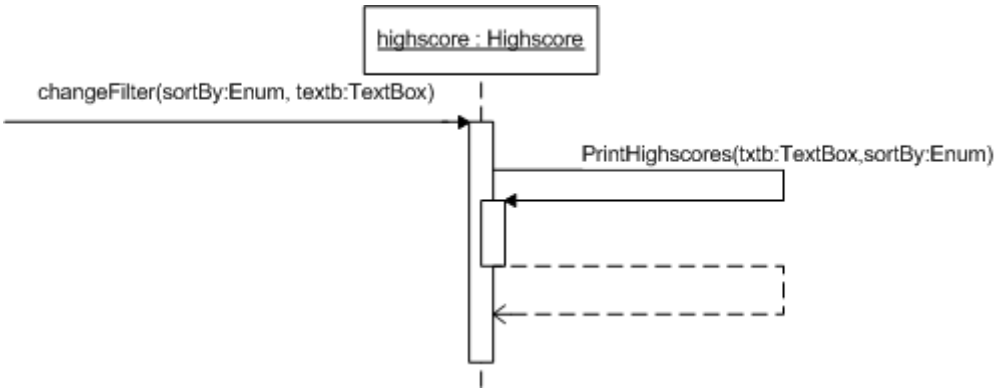
10.26askHighscore(Contract 9.26)



Deze interactiediagram zegt niet veel. Na onze implementatie hiervan, hebben we kunnen zien dat een interactiediagram via de user interface veel meer zegt. Daarom hebben we deze achteraf toegevoegd:



10.27changeFilter (Contract 9.27)



11 Design Patterns

11.1 Singleton

Het verzekeren dat tijdens de uitvoering van onze applicatie een instantie van een bepaalde klasse ten hoogste maar één keer voorkomt. Deze is bereikbaar via een centraal punt.

11.1.1 Participerende klassen

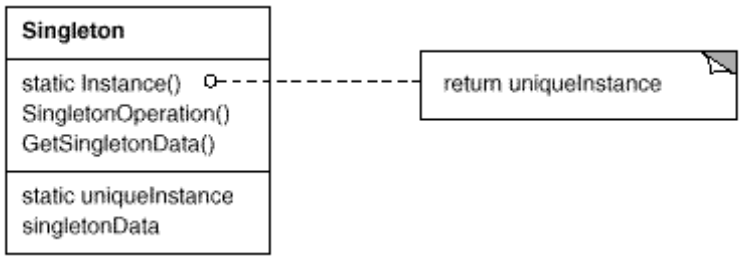
- Game
- Globale static variabele van Game

Game
-_instance : Game = null -_lock : object = new object() -_board : Board -_players : List<Tungrorum.Player> -_iterator : IEnumerator<Tungrorum.Player> -_fieldsFilled : int -_form : frmMain
+Instance() : Game +Game() +InitGame() +SetForm(in form : frmMain) +NextPlayer() +LayTile(in i : int, in j : int, in tile : Field) +board() : Board

11.1.2 Motivatie

In onze applicatie is het belangrijk dat er maar één game tegelijk kan gespeeld worden. We kunnen hier zeker van zijn door doorheen de applicatie maar 1 instantie van de klasse ‘Game’ aan te maken. In onze applicatie gaan we dit verzekeren door globaal een static instantie aan te maken van ‘Game’. Een singleton patroon zou voor meerdere klasses gebruikt kunnen worden in dit project, maar men moet oppassen met het globaal declareren van variabelen. In vele gevallen is dit niet gewenst en maakt dit het alleen maar onoverzichtelijker. Daarom dat alleen de variabele ‘Game’ globaal gedeclareerd zal worden.

11.1.3 Diagram



11.1.4 Externe referentie

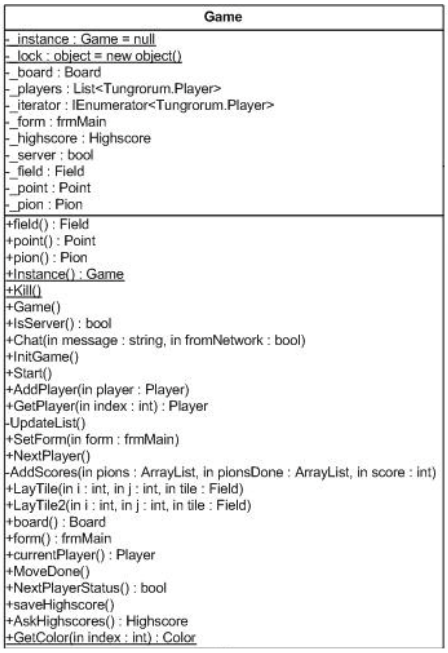
Boek Design Patterns van de GoF, p. 127-134.[1]

11.2 Iterator

Een manier om elementen van een object aan te roepen op een sequentiële manier zonder de onderliggende representatie te kennen.

11.2.1 Participerende klassen

- Player
- Game

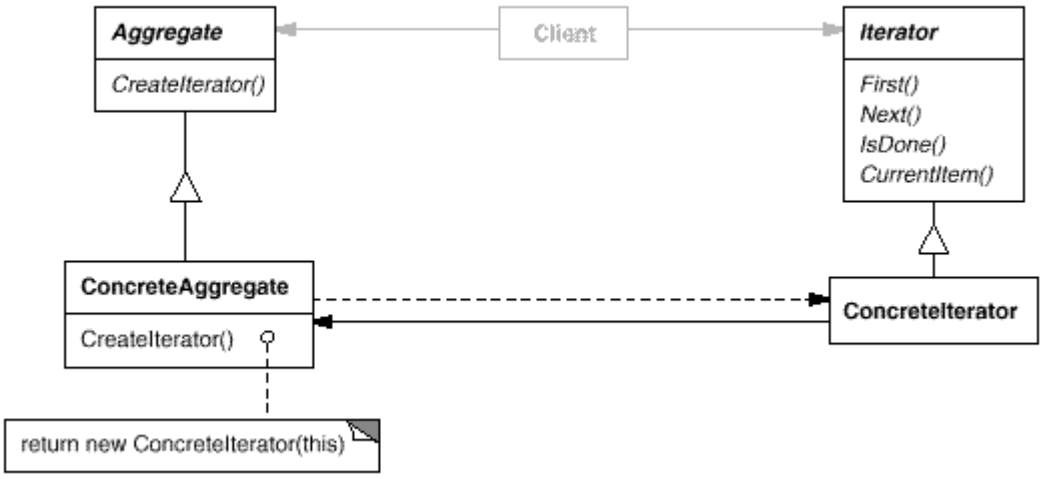


11.2.2 Motivatie

Door middel van een iterator kunnen we sneller en gemakkelijker door onze lijst van spelers lopen. Hoewel we hier steeds met een eindige lijst van spelers werken kunnen we circulair blijven itereren.

[1] Design Patterns, Elements of Reusable Object-Oriented Software, Gamma.E, Helm R., Johnson R., Vlissides J., 36th printing, 2008, p. 127-134

11.2.3 Diagram



11.2.4 Externe referentie

Boek Design Patterns van de GoF, p. 257-271.[2]

11.3 Observer

Een one-to-many afhankelijkheid tussen objecten zodat wanneer een object is veranderd al de afhankelijke objecten dit te weten komen en worden geupdate.

11.3.1 Participerende klassen

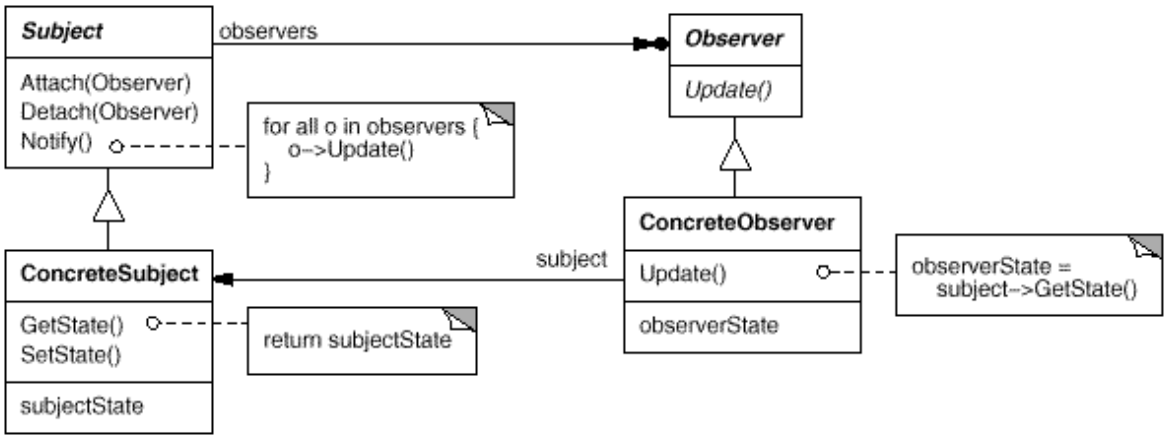
- Board
- Player

11.3.2 Motivatie

De observer klasse zal er voor zorgen wanneer er iets aan ons spel veranderd de observer automatisch de andere objecten dit laat weten. Zo hoeven we niet expliciet een update() functie aan te roepen bij elke verandering die het systeem doet maar wordt het automatisch door de observer opgevangen. Zo zal bijvoorbeeld het bord onmiddellijk aangepast worden wanneer er een tegel gelegd wordt, zodat alle spelers onmiddellijk de verandering kunnen waarnemen. Wanneer men van speler wisselt zal op het bord de huidige speler aangepast worden met de naam en kleur pion.

[2] Design Patterns, Elements of Reusable Object-Oriented Software, Gamma.E, Helm R., Johnson R., Vlissides J., 36th printing, 2008, p. 257-271

11.3.3 Diagram



11.3.4 Externe referentie

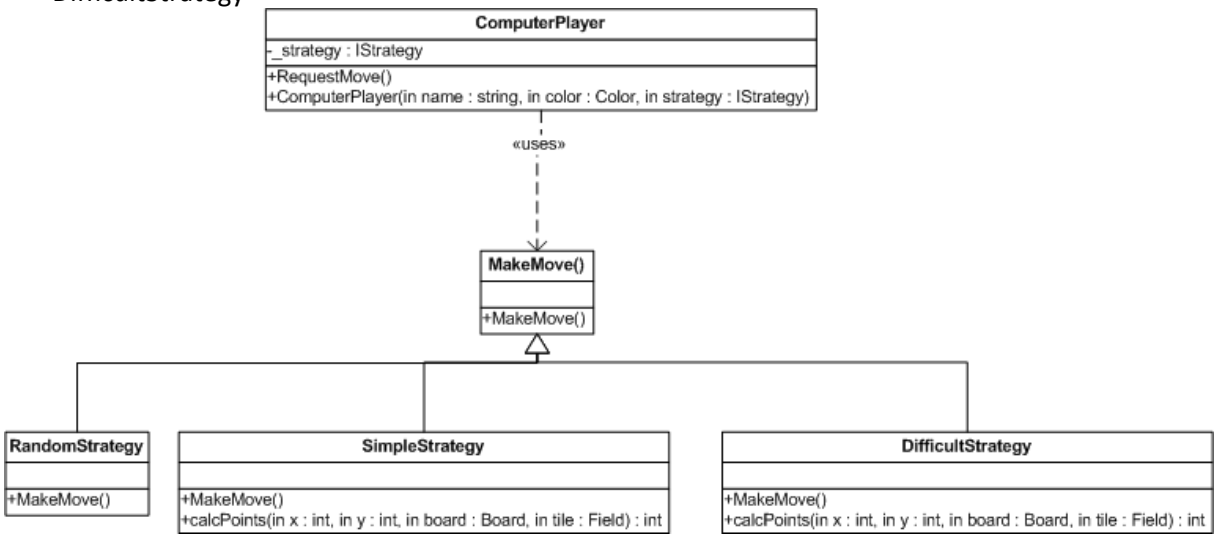
Boek Design Patterns van de GoF, p. 293-304.[3]

11.4 Strategy

Om de computerspeler te kunnen laten kiezen tussen verschillende speelstijlen hebben we gekozen voor het strategy-patroon. Dit patroon laat ons toe om praktisch met afgeleide klassen de gewenste strategie te gebruiken zonder veel omweggen.

11.4.1 Participerende klassen

- ComputerPlayer
- IStrategy
- RandomStrategy
- SimpleStrategy
- DifficultStrategy

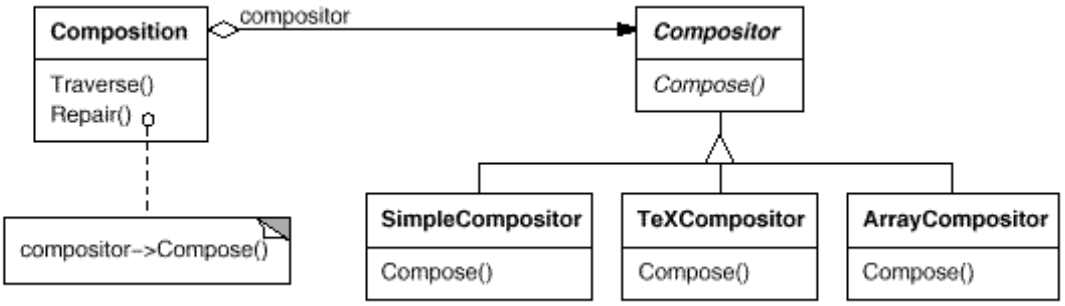


[3] Design Patterns, Elements of Reusable Object-Oriented Software, Gamma.E, Helm R., Johnson R., Vlissides J., 36th printing, 2008, p. 293-304

11.4.2 Motivatie

Bij het instellen van een moeilijkheidsgraad van een AI gebruik kunnen maken van een algoritme-interface die een verschillende strategy teruggeeft naargelang de instelling van de AI. Elke strategie bevat dan een algoritme dat dan de moeilijkheidsgraad bepaalt van de gekozen AI. Aangezien we momenteel beschikken over 3 verschillende strategieën stelt dit patroon ons in staat om zeer makkelijk te verwisselen tussen deze bij het opstarten van een nieuw spel.

11.4.3 Diagram



11.4.4 Externe referentie

Boek Design Patterns van de GoF, p. 315-324.[4]

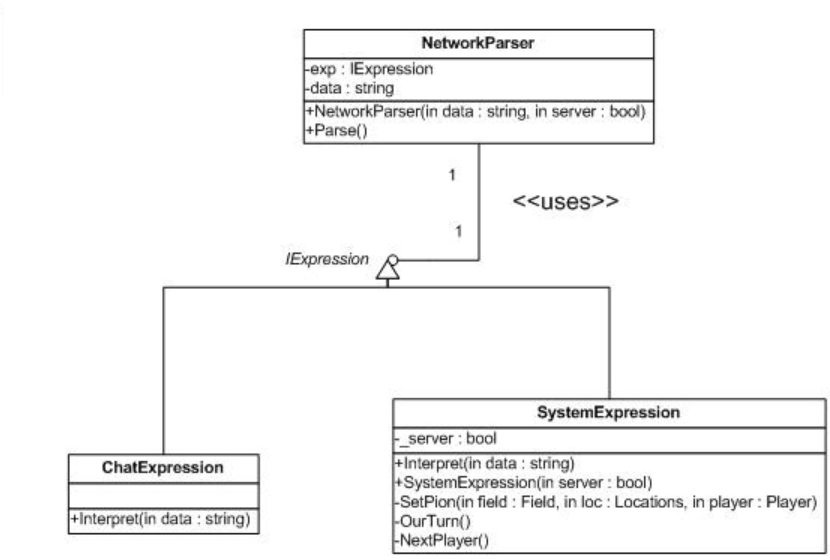
11.5 Interpreter

Het Interpreter design pattern stelt ons in staat om een specifieke taal te maken die ons in staat stelt om boodschappen over een netwerk door te sturen. Door dit patroon te gebruiken, kunnen we meerdere boodschap problemen op 1 manier oplossen.

11.5.1 Participerende klassen

- SystemExpression
- ChatExpression
- NetworkParser

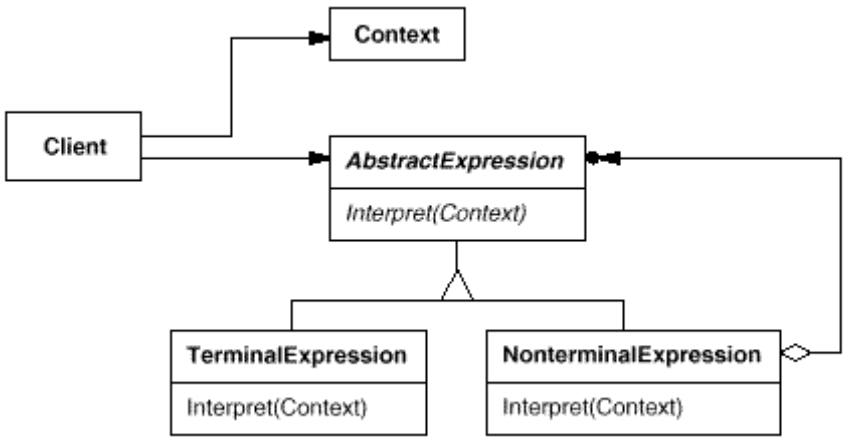
[4] Design Patterns, Elements of Reusable Object-Oriented Software, Gamma.E, Helm R., Johnson R., Vlissides J., 36th printing, 2008, p. 315-324



11.5.2 Motivatie

Omdat verschillende acties doorgestuurd moeten worden over het netwerk was het noodzakelijk om hier strikte regels aan op te leggen. Verkeerde interpretaties van doorgestuurde code kan immers voor foutmeldingen zorgen of onverwachte reacties. Om de foutmeldingen tot een minimum te beperken hebben we gekozen voor het Interpreter design pattern dat ons in staat stelt om de code op een ordelijke te interpreteren en vervolgens de gepaste reacties hier op toepassen.
NetworkParser zal naargelang de doorgestuurde code een SystemExpression of ChatExpression aanmaken waarin deze dan de correcte acties ondernemen.
Door het gebruik van dit patroon kunnen we meerdere boodschappen aanmaken en deze versturen via het netwerk waarbij deze dan op de juiste manier verder afgehandeld worden.

11.5.3 Diagram



11.5.4 Externe referentie

Boek Design Patterns van de GoF, p. 243-256.[5]

12 Opmerkingen

Onze implementatie is wegens tijdsbeperkingen licht verschillend van de analyse. De verschillen zijn:

- Fouten worden niet gelogd (ook niet in analyse).
- Geen ondersteuning voor meerder resoluties (ook niet in analyse).
- Spellen worden niet opgeslaan of ingeladen.
- Er kunnen geen bijzondere spelopties gekozen worden.
- Field- en pion bewerkingen zijn uiteindelijk commandos doorgegeven door de ui geworden.

[5] Design Patterns, Elements of Reusable Object-Oriented Software, Gamma.E, Helm R., Johnson R., Vlissides J., 36th printing, 2008, p. 243-256