Project Software Engineering
Tungrorum

Groepsleden (groep 4):

- Sam van Rijn (groepsverantwoordelijke)
- Sam Van Gucht
- Sibren Polders
- Sibrand Staessens

Probleemstelling:

Het project bestaat uit het implementeren van een variant van Carcassonne, een bordspel voor 2 tot 5 spelers. Elke niet-menselijke speler is een AI-gestuurde robot.
Het spel bestaat uit een vast aantal tegels, elke speler beschikt over 7 pionnen. Elke tegel bestaat uit een aantal landschapselementen (in dit spel: weg, klooster, weide, stad, kruispunt).

Spelverloop:
Eerst worden tegels getrokken, totdat een tegel getrokken wordt die 3 landeigenschappen bevat. Deze dient als starttegel van het spel en wordt op de tafel geplaatst.
Vervolgens trekken spelers om de beurt een tegel, en deze tegel dient als volgt op het spelbord gelegd te worden: de te plaatsen tegel moet aan zijn grenzen landschapselementen bevatten die overeenkomen met de landschapselementen van de daarbij aangrenzende tegels (wei aan wei, stad aan stad, etc.). Indien de tegel geen geldige plaats op het bord kan krijgen, wordt een nieuwe tegel getrokken totdat de getrokken tegel een geldige plaats kan krijgen. Onplaatsbare tegels worden terug bij de nog niet geplaatste tegels gevoegd.
De plaatsing heeft als gevolg dat gebieden op het spelbord worden uitgebreid, wanneer aangrenzende elementen van hetzelfde type zijn. De speler heeft de mogelijkheid om na plaatsing van een tegel, één van zijn pionnen op een landeigenschap van die tegel te plaatsen. Een pion wordt benoemd naargelang de landeigenschap waarop hij geplaatst is (wei → boer, stad → ridder, klooster → monnik, weg → struikrover). Beperkingen worden opgelegd bij het plaatsen van de pion: indien het landschapselement aan een gebied toegevoegd wordt waar reeds een pion op staat, dan mag er geen pion op geplaatst worden. Indien een gebied niet meer kan uitgebreid worden, worden de punten van dat gebied aan de speler zijn totaal toegekend (zie hieronder voor de puntentelling) en mag hij deze pion terugnemen (alleen als dit geen boer op een weide is, die moeten tot het einde van het spel geplaatst blijven).
Als geen tegels meer geplaatst kunnen worden, is het spel gedaan en worden de punten als volgt toegekend (deze regels gelden ook voor wanneer in het spel zelf een gebied vervolledigd wordt):

- Indien meerdere spelers pionnen hebben op een afgewerkt gebied, worden de punten toegekend aan de speler met de meeste pionnen op dat gebied. Indien er een gelijk aantal pionnen is, worden de punten aan elke speler toegekend.
- Ridders, struikrovers en monniken die nog op onafgewerkte gebieden staan op het einde van het spel zijn waardeloos. De boeren verzorgen enkel afgewerkte steden langs hun weiland. Voor elke afgewerkte stad krijgt de speler met de meeste boeren in dat weiland 4 punten.
- Ridder (stad): Per stadstegel komen er twee punten bij.
- Monnik (klooster): Een klooster is afgewerkt wanneer het omringd wordt door 8 tegels. Hiervoor krijgt de speler 9 punten.
- Struikrover (weg): Een weg kan eindigen bij een stad, kruispunt of klooster. Per tegel krijgt de speler hier 1 punt voor.

Systeemattributen:

- Robuustheid
- Vlot speelbaar
- Duidelijke en responsieve interface
- Plezierigheid
- Geen zware systeemeisen
- Object-Oriented (C++ en Qt)
- Portability

Systeemfuncties:

R1. Programma opstarten
R2. Programma afsluiten
R3. Nieuw lokaal spel starten
- aantal spelers instellen
- aantal en niveau van computergestuurde spelers instellen
- weergave van de verschillende spelers instellen
- aantal landtegels instellen
- standaardinstellingen instellen

R4. Nieuwe starttegel genereren
R5. Spel beëindigen
R6. Nieuw spel starten met de vorige instellingen (herstarten)
R7. Opslaan van een spel
R8. Voortzetten van een vorig spel (laden van een spel)
R9. Help raadplegen
R10. Van speler wisselen
R11. Aangeven welke speler aan beurt is
R12. Spelers die het spel verlaten, computergestuurd instellen/maken
R13. Tegel van stapel nemen
R14. Tegel terug bij stapel plaatsen
R15. Tegel draaien
R16. Tegel op tafel plaatsen
- validatie plaatsing
- melding bij verkeerde plaatsing

R17. Pion plaatsen
- validatie plaatsing
- type toewijzen bij correcte plaatsing
- melding bij verkeerde plaatsing

R18. Pion terugnemen
- validatie terugname
- melding bij verkeerde terugname
- punten toekennen bij correcte terugname

R19. Zicht over geplaatste tegels kunnen veranderen
R20. Informatie over het programma weergeven
R21. Spel pauzeren
R22. Punten per speler bijhouden
R23. Spelerstatistieken bijhouden

- punten
- pionnen
- positie in de ranglijst

R24. Aantal resterende tegels weergeven
R25. Duur van het spel weergeven
R26. Einde van het spel vaststellen
- Punten tellen

R27. Highscoretabellen bijhouden (alleen voor menselijke spelers)
R28. Geluid afspelen als
- ongeldige zet gedaan
- wisselen speler
- bijtelling punten
- einde spel
- pion plaatsen
- landtegel plaatsen
- draaien landtegel
- pion terugnemen
- starttegel genereren
- achtergrondmuziek

R29. Geluid aan- en uitzetten
- achtergrond
- effectgeluiden

R30. Achtergrond instellen
R31. Achtergrond inladen
R32. Cheats toelaten (vooral voor het testen van de software)
R33. Animatie weergeven bij
- draaien tegels
- pion plaatsen

Brief Use Cases:

UC 1: Nieuw spel starten
Actoren : Speler(s)
Beschrijving : Als de applicatie is opgestart kan de speler een nieuw spel configureren en opstarten. De speler voert allerlei specificaties in naargelang zijn voorkeur. Hierna start de speler het spel.

UC 2: Een beurt afleggen
Actoren : Huidige speler
Beschrijving : Tijdens het verloop van het spel trekt de huidige speler een tegel van de stapel. Deze plaatst hij dan op de tafel. Waarna de speler een pion kan plaatsen op de zojuist gelegde tegel en/of een kan terugnemen. Daarna wordt de beurt gegeven aan de volgende speler.

UC 3: Een pion plaatsen
Actoren : Huidige speler
Beschrijving : Nadat de speler een tegel geplaatst heeft, kiest hij (indien mogelijk) of hij een pion op de net geplaatste tegel legt of niet. Hierna gaat de beurt naar de volgende speler.

UC 4: Een pion terugnemen
Actoren : Huidige speler
Beschrijving : Als de speler aan de beurt is, kan hij ervoor kiezen eerder geplaatste pionnen terug van het veld te nemen. Hiervoor moet de pion op een volledig veld staan en mag de pion geen boer zijn.

UC 5: Help raadplegen
Actoren : Een speler
Beschrijving : Tijdens het uitvoeren van de applicatie wenst een speler meer informatie te bekomen over spelregels, programmainstellingen, ... Hiertoe roept hij het systeem op om hulp. Na het ingeven van bepaalde zoektermen, geeft het systeem de gevraagde informatie weer. Nadat de gebruiker al het gewenste is te weten gekomen, laat hij het systeem weten dat hij de hulpfunctie mag afsluiten.

UC 6: Programmainstellingen veranderen
Actoren : Een speler
Beschrijving : Tijdens het uitvoeren van de applicatie wenst een speler bepaalde programmainstellingen te wijzigen (zoals geluid aan/uit, achtergrond, animatie aan/uit, grootte van de tafel). Hiertoe laat hij het programma weten dat hij een bepaalde instelling wil veranderen. Hij geeft het programma de gewenste vernieuwing en het programma keurt dat dan goed of niet. Na de goedkeuring worden de veranderingen doorgevoerd.

UC 7: Spel laden
Actoren : Een speler
Beschrijving : Tijdens het uitvoeren van de applicatie kan een speler een vorige spelsituatie inladen. De gebruiker geeft dan de gewenste bestandslocatie en -naam door aan het programma. Het programma laadt het opgegeven bestand in en hervat het opgeslagen spel.

UC 8: Spel opslaan
Actoren : Een speler
Beschrijving : Tijdens het spelverloop kan een speler het huidige spel naar een bestand opslaan. De speler geeft dan een gewenste bestandslocatie door aan het programma. Het programma slaat de huidige spelsituatie op in een bestand met een standaardnaam.

UC 9: Speler verwijderen
Actoren : Een speler
Beschrijving : Tijdens het spelverloop kan een speler, die aan de beurt is, zichzelf uit het spel verwijderen. Dit laat hij aan de applicatie weten, en na bevestiging door de gebruiker wordt deze in het spel vervangen door een computergestuurde speler. Het niveau van die speler wordt dan bepaald a.h.v. de tot dan behaalde punten.

UC 10: Tafeloverzicht wijzigen
Actoren : Een speler
Beschrijving : De speler wenst tijdens het spelverloop het tafeloverzicht te wijzigen. Dit doet hij door het programma beweegrichtingen aan te geven. De applicatie visualiseert deze beweging met een wijzing van het tafeloverzicht als gevolg.

Fully Dressed Use Cases:

UC 1: Nieuw spel starten
Cross References: R3, R4, R8, R10, R11, R25, R28
Primary Actor: Speler
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: De applicatie is succesvol opgestart. (R1)
Succes Guarantee (Postconditions): Het spel start met de gewenste configuratie, de initiële starttegel is geplaatst. De timer is gestart en de eerste speler kan beginnen.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
● Vraagt om een nieuw spel te starten.	● Geeft alle instel mogelijkheden weer, met standaard waarden voor aantal menselijke en computergestuurde spelers, hun respectievelijke kleuren, namen, niveaus en het aantal tegels dat het spel bevat.(R3)
● Start het spel met de standaard instellingen.	● Stelt starttegel voor (R4) + speelt een passend geluidje af (R28).
● Keurt starttegel goed.	● Geeft de beurt aan de eerste speler.(R10 + R11) + speelt een passend geluidje af (R28) + start de timer (R25).

Extensions (or Alternitive Flows):

Actor Action (or Intention)	System Responsibility
*a. Op elk ogenblik kan de speler kiezen om het spel niet te starten. 1. Kiest ervoor om het proces voortijdig te beëindigen.	2a. Indien er een vorige spel bestaat wordt dit hervat (R8 -> spel van geheugen i.p.v. van elders inlezen). 2b. Anders terug naar vorige situatie.
*b. Op elk ogenblik kan het systeem falen. 1. Start het systeem opnieuw op	
3a. Verandert instellingen 1a. Laat het standaard aantal menselijke spelen staan.	

1b. Stelt aantal menselijke spelers in. 2a. Laat het standaard aantal computergestuurde spelen staan. 2b. Stelt aantal computergestuurde spelers in. 3a. Laat de standaard kleuren staan. 3b. Stelt de nieuwe kleuren in voor de verschillende spelers. 4a. Laat de standaard namen staan. 4b. Stelt de namen voor de verschillende spelers in. 5a. Laat de standaard niveaus van de computergestuurde spelers staan. 5b. Stelt de niveaus van de computergestuurde spelers in. 6a. Laat het standaard aantal tegels staan. 6b. Stelt het aantal tegels in. 5a[1]. Keurt starttegel af	5a[2]. Herstart stap 4 in de main flow.

<u>UC 2:</u> Beurt afleggen
Cross References: R5, R6, R10, R11, R13, R14, R15, R16, R17, R18, R21, R22, R24, R26, R28, R32, R33, UC3, UC4
Primary Actor: Speler
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: Een spel is succesvol gestart. De huidige speler is aan de beurt.
Succes Guarantee (Postconditions): Er is een tegel op de tafel gelegd.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. Vraagt om een tegel van de stapel te nemen. 3. Plaatst tegel.	2.Geeft een tegel aan speler (R13 en R24) + speelt een passend geluidje af (R28). 4. Verifieert de gedane zet. (R16) 5. Geeft aan dat de zet ok is en speelt geluid af. (R28) 6. UC3 uitvoeren. 7. Systeem geeft de beurt door aan een volgende speler. (R10 + R11)

Extensions (or Alternitive Flows):

Actor action or intention	System Responsibility
*a. De speler wenst het spel te herstarten met de huidige instellingen.	
	1. Vraagt om bevestiging.
2a. Bevestigt.	
	3a. Herstart het spel met de huidige instellingen. (R6)
2b. Bevestigt niet.	
	3b. Het huidige spel wordt verdergezet.
*b. Kiest ervoor om het spel te pauzeren.	
	1. Pauzeert het spel (R21) (stopt de tijdsmeting) en wacht op invoer van de speler.
2. Wenst te hervatten.	
	3. Hervat spel en tijdsmeting.
*c. Kiest ervoor om een cheat in te voeren.	
	1. Voert de cheat uit. (R32)
*d. Kiest ervoor om een spel in te laden. (Use Case 5)	
*e. Kiest ervoor om het spel op te slaan. (Use Case 6)	
*f. Kiest ervoor om de help te raadplegen. (Use Case 3)	
*g. Kiest ervoor om een speler te verwijderen. (Use Case 7)	
*h. Kiest ervoor om het speloverzicht te wijzigen. (Use Case 8)	
*i. Kiest ervoor om het spel te beëindigen. (R5)	
*j. Op elk moment kan UC4 uitgevoerd worden.	
	1a. Geeft geen tegel aan speler, wegens geen tegels meer. (R26 en R24) Bereken de scores (R22) en duid de winnaar aan (R28). (R5)
2a. Vraagt om tegel te draaien. 1a. Vraagt om naar links te draaien 1b. Vraagt om naar rechts te draaien	
	2. Draait tegel (R15), speelt geluid af (R28), toont animatie. (R33)
3a. Herhaal stap 1a of 1b.	

3b. Ga naar stap 2 in de basic flow.	
	4a. Accepteert de plaatsing niet en speelt een geluid af. (R28) 1. Geeft de tegel terug.
2-4a. Stelt vast dat de tegel niet plaatsbaar is. 1. Vraagt om de tegel terug in de stapel te leggen.	
	2. Legt de tegel terug in de stapel. (R14 en R24) 3. Terug naar stap 1 in de basic flow.
2-4b. Stelt vast dat de tegel op een andere plaats wel plaatsbaar is.	
	1. Geeft de tegel terug.
2. Terug naar stap 2 in de basic flow.	

<u>UC 3:</u> Pion plaatsen
Cross References: R17, R28
Primary Actor: Speler
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: Een spel is succesvol gestart. De huidige speler is aan de beurt. De speler heeft zojuist een tegel geplaatst.
Succes Guarantee (Postconditions): Er is een pion op de tegel geplaatst.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. Kiest een pion te plaatsen. 1a. Plaatst de pion op de weide. 1b. Plaatst de pion op de weg. 1c. Plaatst de pion op het stadsdeel. 1d. Plaatst de pion op het klooster.	
	2. Verifieert de plaatsing. (R17.1) 1a. Als de zet geldig is, speelt het systeem een geluid (R28) af naargelang het landsdeel waar de pion op geplaatst werd. (R17.2) 1b. Als de zet niet geldig is, geeft het systeem de pion terug en wordt een geluid afgespeeld. (R28 + R17.3)

Extensions (or Alternitive Flows):

Actor action or intention	System Responsibility
1a. Kiest ervoor geen pion te plaatsen	

<u>UC 4:</u> Pion terugnemen
Cross References: R17, R28
Primary Actor: Speler
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: Een spel is succesvol gestart. De huidige speler is aan de beurt. De speler heeft ooit een pion op een tegel geplaatst.
Succes Guarantee (Postconditions): Er is een pion van een tegel genomen en de puntentelling is aangepast.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. Neemt de pion terug.	2. Verifieert het terugnemen van de pion. (R18.1) 1a. Als het terugnemen geldig is, werkt het systeem de punten bij (R18.3 en R23 en R27) en wordt er een geluid afgespeeld (R28). 1b. Als het terugnemen niet geldig is, gebeurt er niets en wordt er een geluid afgespeeld (R28 + R18.2).

Extensions (or Alternitive Flows):

Actor action or intention	System Responsibility
1a. Neemt geen pion terug.	

<u>UC 5:</u> Help raadplegen
Cross References: R1, R9, R21
Primary Actor: Speler
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: De applicatie is succesvol opgestart. (R1)
Succes Guarantee (Postconditions): De speler heeft de gewenste informatie verkregen en gebruikt deze om verder te gaan.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. Roept de hulpfunctie van de applicatie op.	2. Laat weten dat hij wacht op input van de gebruiker. (R9)
3. Geeft de gewenste zoektermen op.	4. Geeft informatie die het beste overeenkomt met de zoektermen.
5. Gebruikt de informatie die nuttig is voor hem/haar. Waarna hij het spel hervat.	6. Het programma/spel wordt hervat.

Extensions (or Alternitive Flows):

Actor Action (or Intention)	System Responsibility
*a. De gebruiker kan op elk moment de hulp van het programma beëindigen.	1. Ga naar stap 6.
	*b. Indien help opgeroepen wordt tijdens het spelverloop, wordt het spel gepauzeerd. (R21)
	4a. Het programma vindt geen informatie bij de opgegeven zoekterm.
1. Ga naar stap 3.	
5a. De speler krijgt niet zijn gewenste informatie. 1. Ga naar stap 3.	
5b. De gebruiker wil opnieuw zoeken. 1. Ga naar stap 3.	

<u>UC 6:</u> Programmainstellingen veranderen
Cross References: R1, R31
Primary Actor: Gebruiker
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: De applicatie is succesvol opgestart. (R1)
Succes Guarantee (Postconditions): De instelling die de gebruiker heeft ingegeven worden merkbaar.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. De speler vraagt de mogelijkheid om instellingen te wijzigen.	
	2. Het programma geeft de mogelijkheid om instellingen te wijzigen.
3. De speler verandert instellingen. 1a. Verandert de achtergrondafbeelding. (R30)	
	2. Controleer de geldigheid van het opgegeven bestand. Het bestand is geldig en wordt ingeladen. (R31)
1b. Zet Geluid aan of uit. (R29) 1c. Zet animatie aan of uit. 1d. Verandert de tafelgrootte.	
	2. Controleer of de tafelgrootte geldig is.
4. Speler keurt de instellingen goed.	
	5. Voert de veranderingen door. 6. Hervat het verloop van het programma.

Extensions (or Alternitive Flows):

Actor Action (or Intention)	System Responsibility
*a. De gebruiker kan op elk moment het wijzigen van de instellingen annuleren.	
	1. Ga naar stap 6.
*b. De gebruiker kan op elk moment de standaardwaarden terug verkrijgen.	
*c. De gebruiker kan op elk moment de vorige instellingen terug verkrijgen.	
3a. Ga naar stap 3.	
	3(1a).2.a. Het opgegeven bestand is ongeldig. 1. Geeft melding en gebruikt het opgegeven bestand niet.
2. Ga naar stap 3	
	3(1d).2.a. De opgegeven tafelgrootte is ongeldig. 1. Geeft melding en gebruikt de opgegeven tafelgrootte niet.
2. Ga naar stap 3.	
4a. De speler keurt de instellingen af.	

	1. Ga naar stap 6.

<u>UC 7:</u> Spel laden
Cross References: R1, R8, R27
Primary Actor: Gebruiker
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: De applicatie is succesvol opgestart (R1), er is ooit een spel opgeslaan.
Succes Guarantee (Postconditions): De gebruiker hervat het opgeslagen spel.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. De gebruiker vraagt de mogelijkheid om een spel te laden.	
	2. Geeft de mogelijkheid om een spel te zoeken en te laden.
3. De gebruiker kiest een bestandslocatie.	
	4. Controleer geldigheid van de bestandslocatie.
	5. De locatie is geldig, geeft de inhoud van de opgegeven locatie weer.
6. De gebruiker kiest een bestandsnaam.	
	7. Controleert geldigheid van het bestand
	8. De opgegeven bestandsnaam is geldig, het spel wordt ingeladen en hervat. (R8 en R27)

Extensions (or Alternitive Flows):

Actor Action (or Intention)	System Responsibility
*a. Op elk moment kan de gebruiker het laden van een spel annuleren.	1. Hervat wat het bezig was.
3a. Ga naar stap 3.	
3b. Ga naar stap 6, gebruik makende van een voorgestelde locatie.	
	5a. De opgegeven bestandslocatie is ongeldig. 1. Geeft een melding. 2. Ga naar stap 3.
6a. Ga naar stap 6. 6b. Ga naar stap 3.	

	8a. Het opgegeven bestand is ongeldig. 1. Geeft een melding. 2. Ga naar stap 6.

<u>UC 8:</u> Spel opslaan
Cross References: R1, R7, R27
Primary Actor: Gebruiker
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: De applicatie is succesvol opgestart (R1), er is een spel gestart.
Succes Guarantee (Postconditions): De gebruiker hervat het huidige spel en dit spel werd naar een bestand weggeschreven.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. Vraagt de mogelijkheid om een spel op te slaan.	
	2. Geeft de mogelijkheid om een bestandslocatie te zoeken.
3. Kiest een bestandslocatie.	
	4. Controleert geldigheid van de bestandslocatie.
	5. De locatie is geldig, geeft de inhoud van de opgegeven locatie weer.
6. Kiest een bestandsnaam.	
	7. Controleert geldigheid van de extentie van het bestand en het uniek zijn van de bestandsnaam.
	8. De opgegeven bestandsnaam is geldig, het spel wordt opgeslagen en voortgezet. (R7 en R27)

Extensions (or Alternitive Flows):

Actor Action (or Intention)	System Responsibility
*a. Op elk moment kan de gebruiker het opslaan van een spel annuleren.	1. Hervat wat het bezig was.
*b. Op elk ogenblik kan het opslagmedium plaatsgebrek hebben.	1. Geeft een melding. 2. Keert terug naar de vorige stap.
3a. Ga naar stap 3.	

	5a. De opgegeven bestandslocatie is ongeldig. 1. Geeft een melding. 2. Ga naar stap 3.
6a. Ga naar stap 6. 6b. Ga naar stap 3.	
	7a. De opgegeven bestandsnaam bestaat al. 1. Geeft een melding, of hij mag overschrijven of niet.
2a. Kiest ervoor niet te overschrijven.	1. Ga naar stap 6.
2b. Kiest ervoor wel te overschrijven.	1. Ga naar stap 8.
	8a. De opgegeven extentie is ongeldig. 1. Geeft een melding. 2. Ga naar stap 6.

<u>UC 9:</u> Speler verwijderen
Cross References: R1, R12
Primary Actor: Gebruiker
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: De applicatie is succesvol opgestart (R1), er is een actief spel.
Succes Guarantee (Postconditions): De speler is vervangen door een computergestuurde speler.

Main Succes Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. Vraagt om zichzelf uit het spel te verwijderen.	
	2. Vraagt om bevestiging.
3. Bevestigt.	
	4. Stelt standaard niveau van de computergestuurde speler voor.
5. Aanvaardt het voorgestelde niveau.	
	6. Vervangt de speler met een computergestuurde speler en hervat het spel. (R12)

Extensions (or Alternitive Flows):

Actor Action (or Intention)	System Responsibility

*a. Op elk moment kan de gebruiker het verwijderen van de speler annuleren.	1. Hervat wat het bezig was.
3a. Bevestigt niet.	1. Hervat wat het bezig was.
5a. Aanvaardt het voorgestelde niveau niet. 1. Kiest ander niveau uit een aantal mogelijkheden.	2. Ga naar stap 6.

<u>UC 10:</u> Tafeloverzicht wijzigen
Cross References: R1, R19, R28
Primary Actor: Gebruiker
Stakeholders and Interests:
- Mogelijke menselijke medespelers
- Supporters

Preconditions: De applicatie is succesvol opgestart (R1), er is een spel gestart.
Succes Guarantee (Postconditions): De gebruiker hervat het huidige spel, met een nieuw overzicht.

Main Succes Scenario (or Basic Flow):

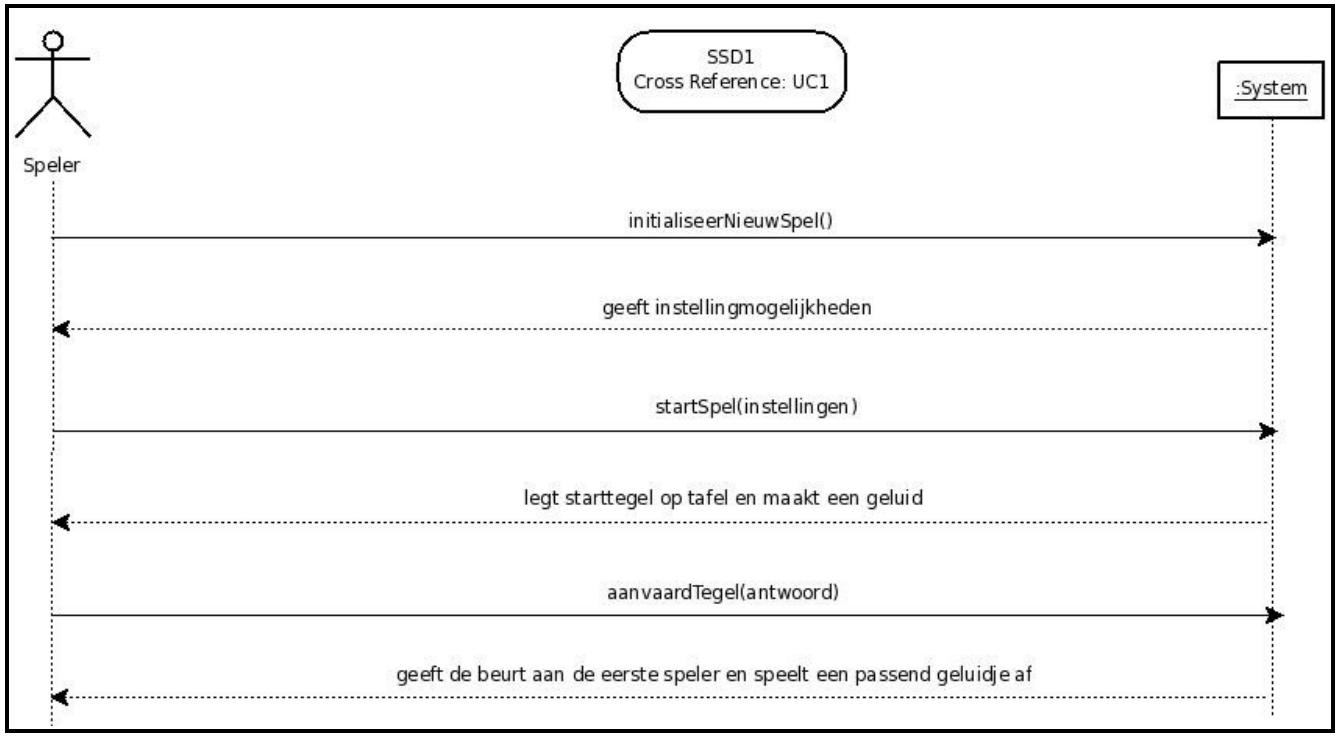
Actor Action (or Intention)	System Responsibility
1. Wijzigt het overzicht. 1a. Beweegt het veld naar links. 1b. Beweegt het veld naar rechts. 1c. Beweegt het veld naar boven. 1d. Beweegt het veld naar onder. 1e. Roteert het veld naar links. 1f. Roteert het veld naar rechts. 1g. Zoomt in. 1h. Zoomt uit.	2. Controleert of de wijziging mogelijk is. 3. Indien de wijzing mogelijk is, voert deze uit. (R19) 4. Het spel wordt hervat.

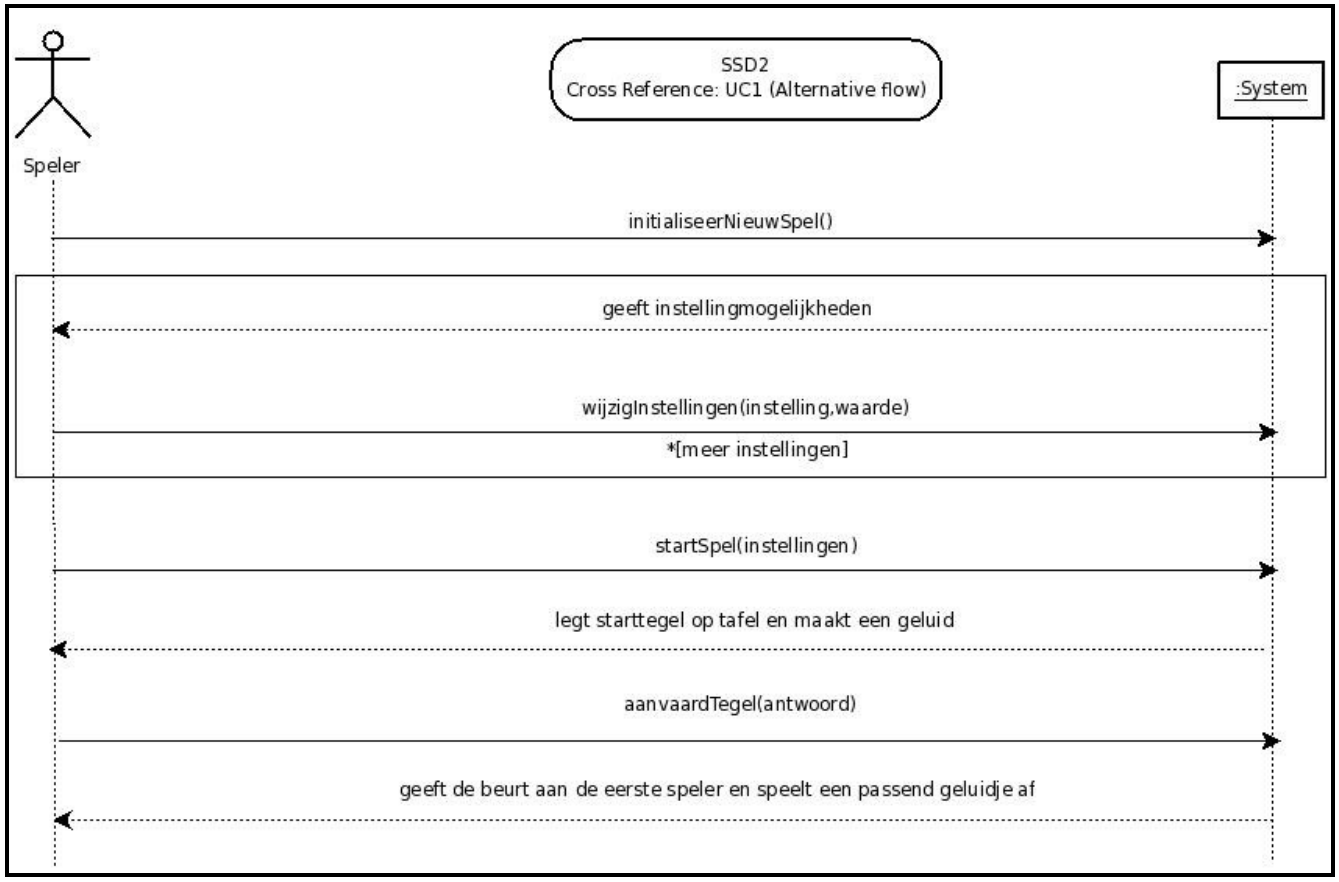
Extensions (or Alternitive Flows):

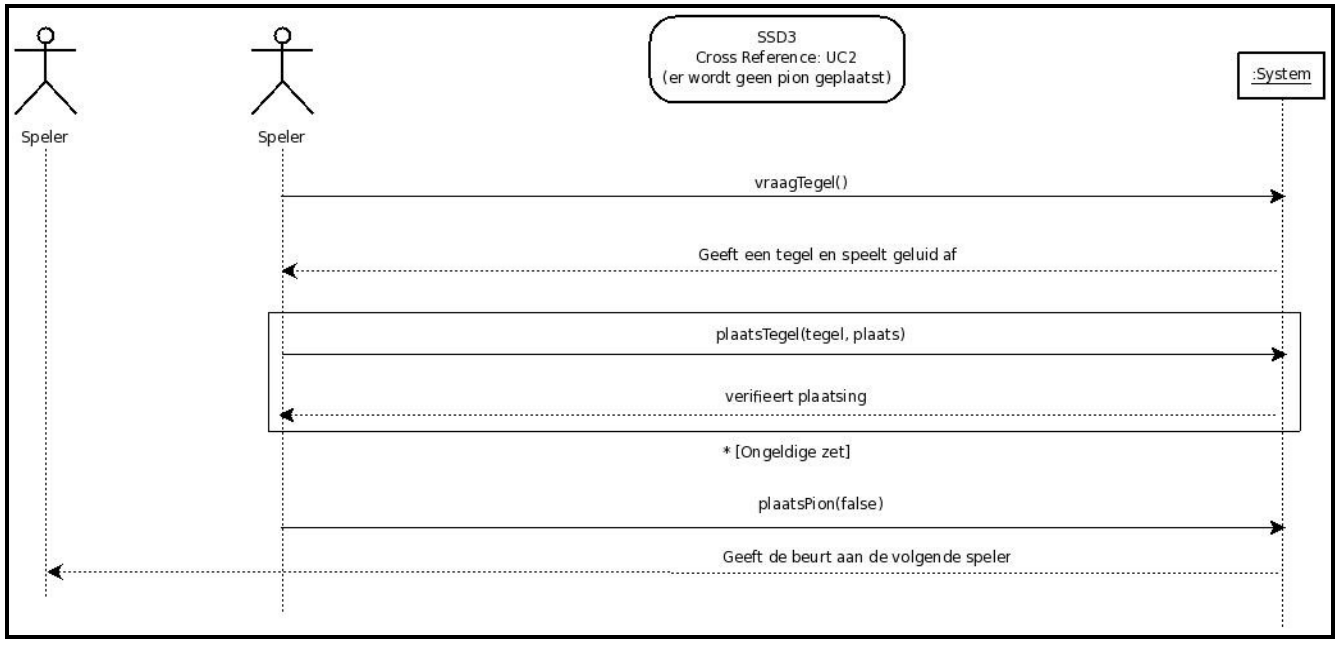
Actor Action (or Intention)	System Responsibility
*a. Op elk moment kan de gebruiker het overzicht herzetten naar het standaard beeld.	1. Herzet het overzicht naar standaardwaarden.

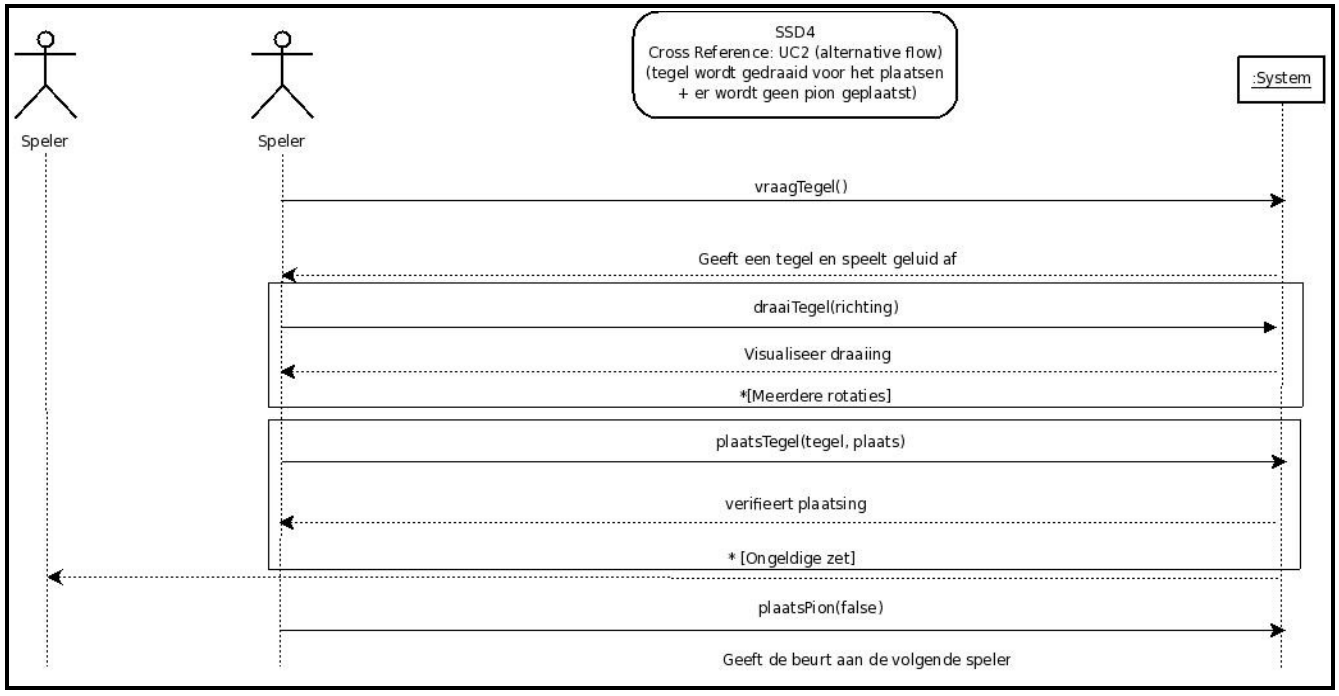
*b. Op elk ogenblik kan de gebruiker zijn laatste camerabewerking ongedaan maken. 1a. Ga naar stap 1.	1. Herzet het overzicht naar de vorige waarden. 3a. Indien de overzichtswijziging niet mogelijk is. 1. Geeft melding met geluid. (R28) 2. Ga naar stap 4.

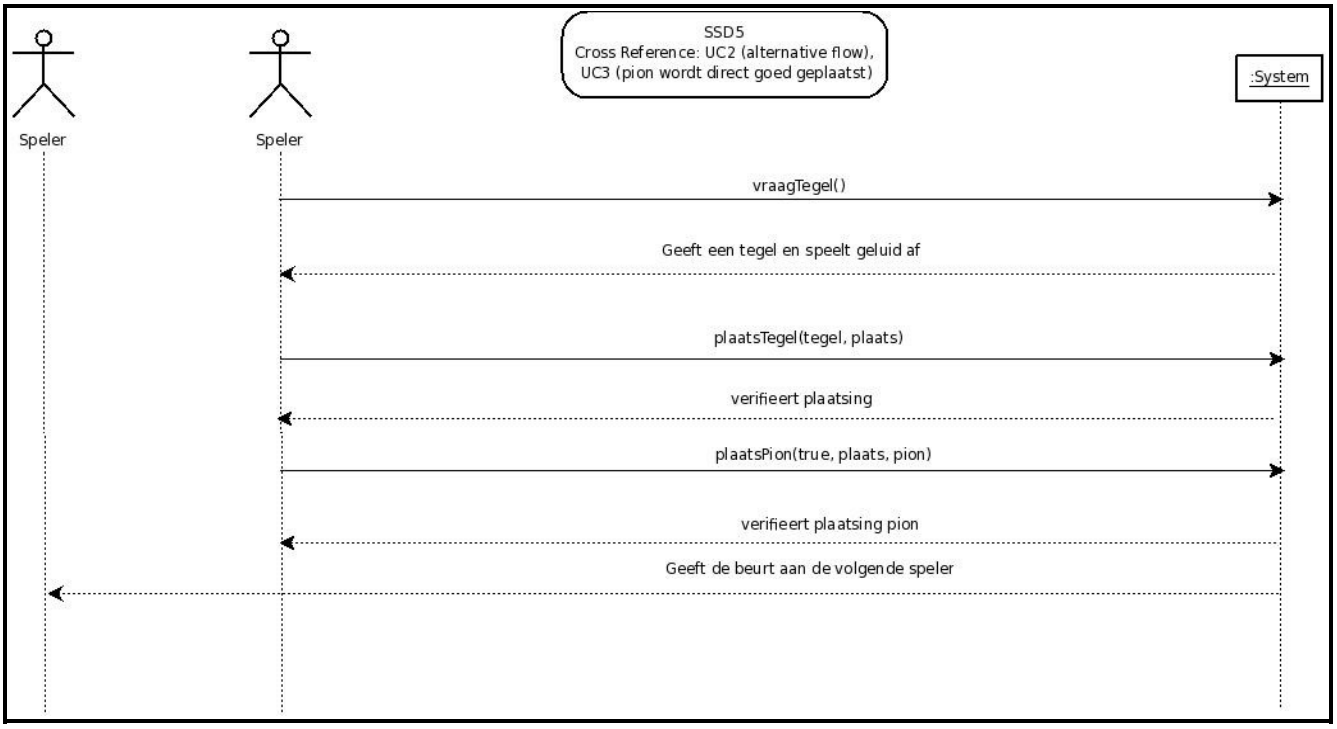
Systeemsequentiediagrammen:

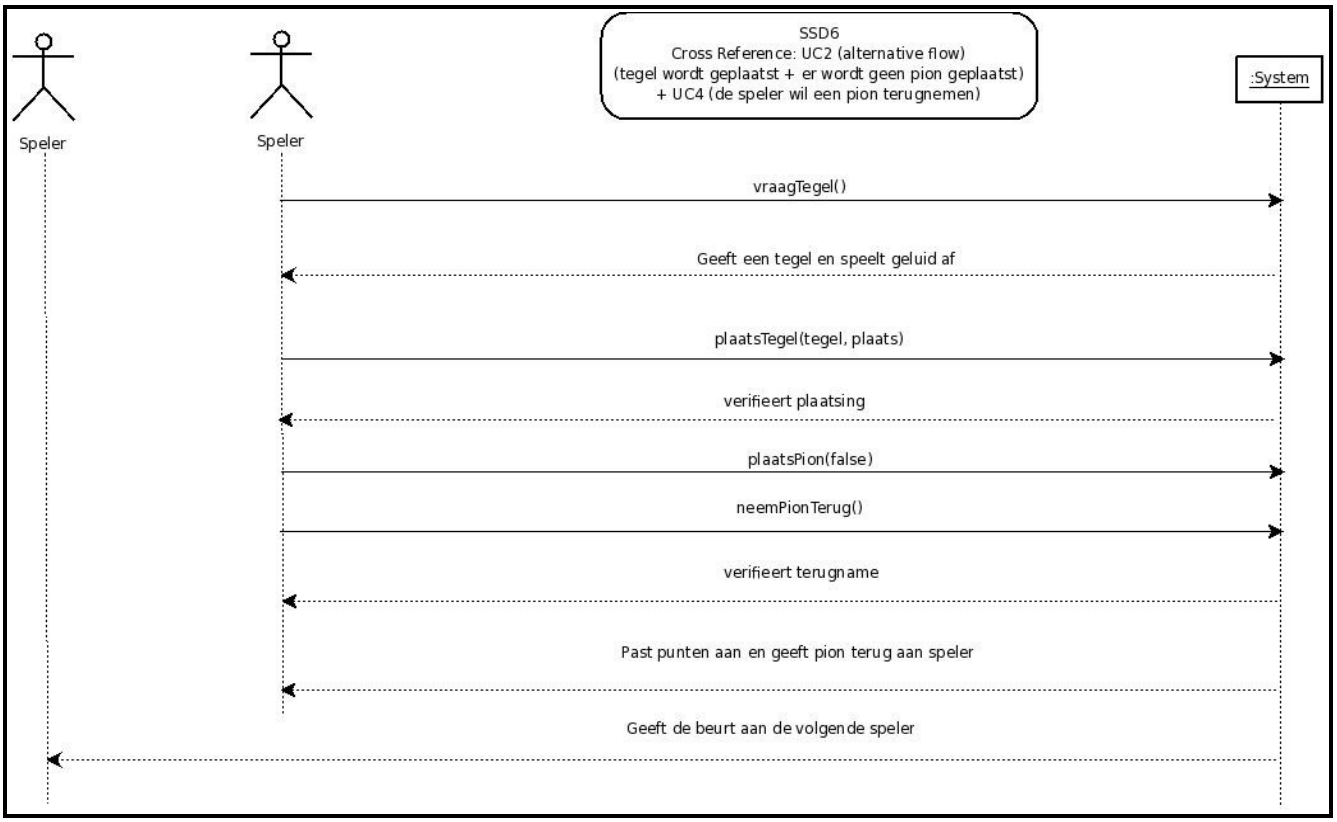


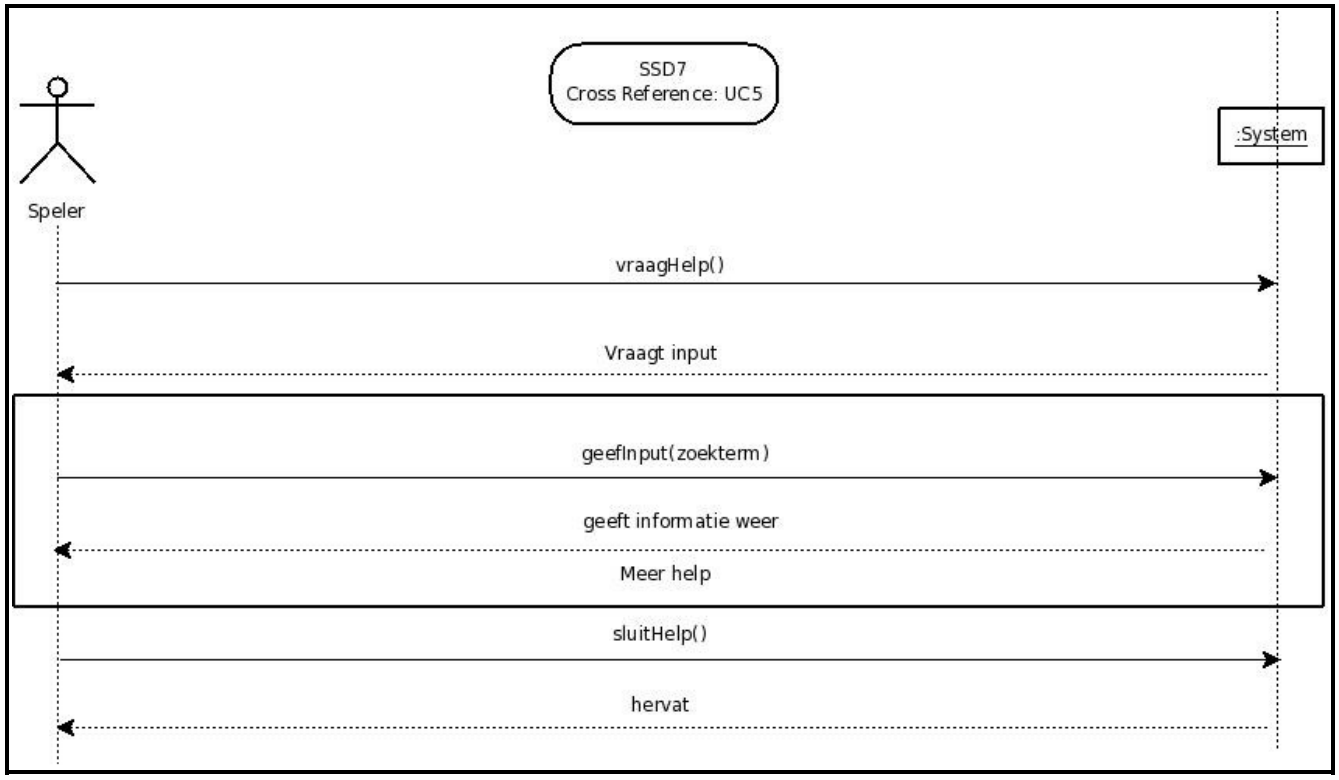


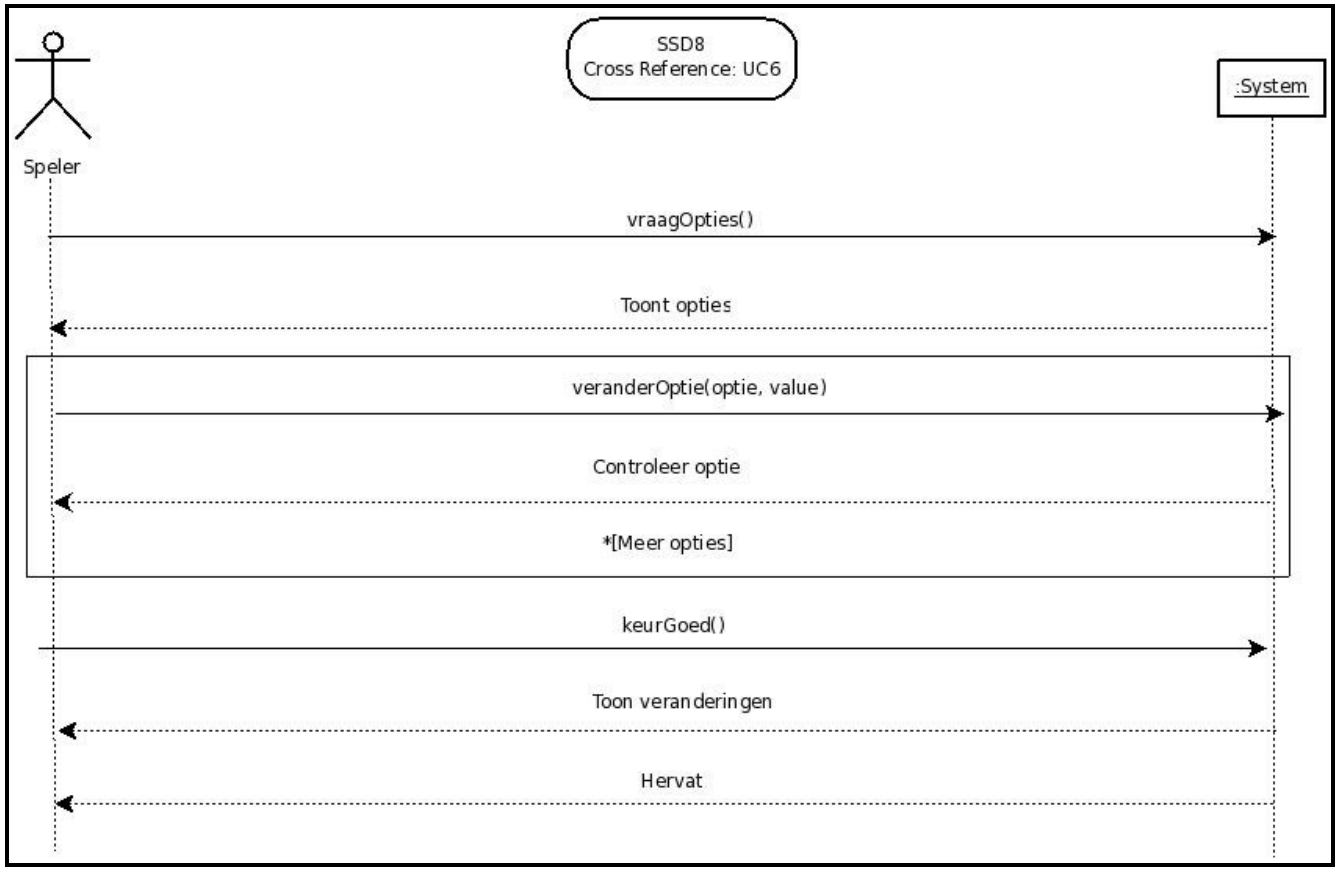

SSD3
Cross Reference: UC2
(er wordt geen pion geplaatst)
Speler
Speler
:System
vraagTegel()
Geeft een tegel en speelt geluid af
plaatsTegel(tegel, plaats)
verifieert plaatsing
* [Ongeldige zet]
plaatsPion(false)
Geeft de beurt aan de volgende speler

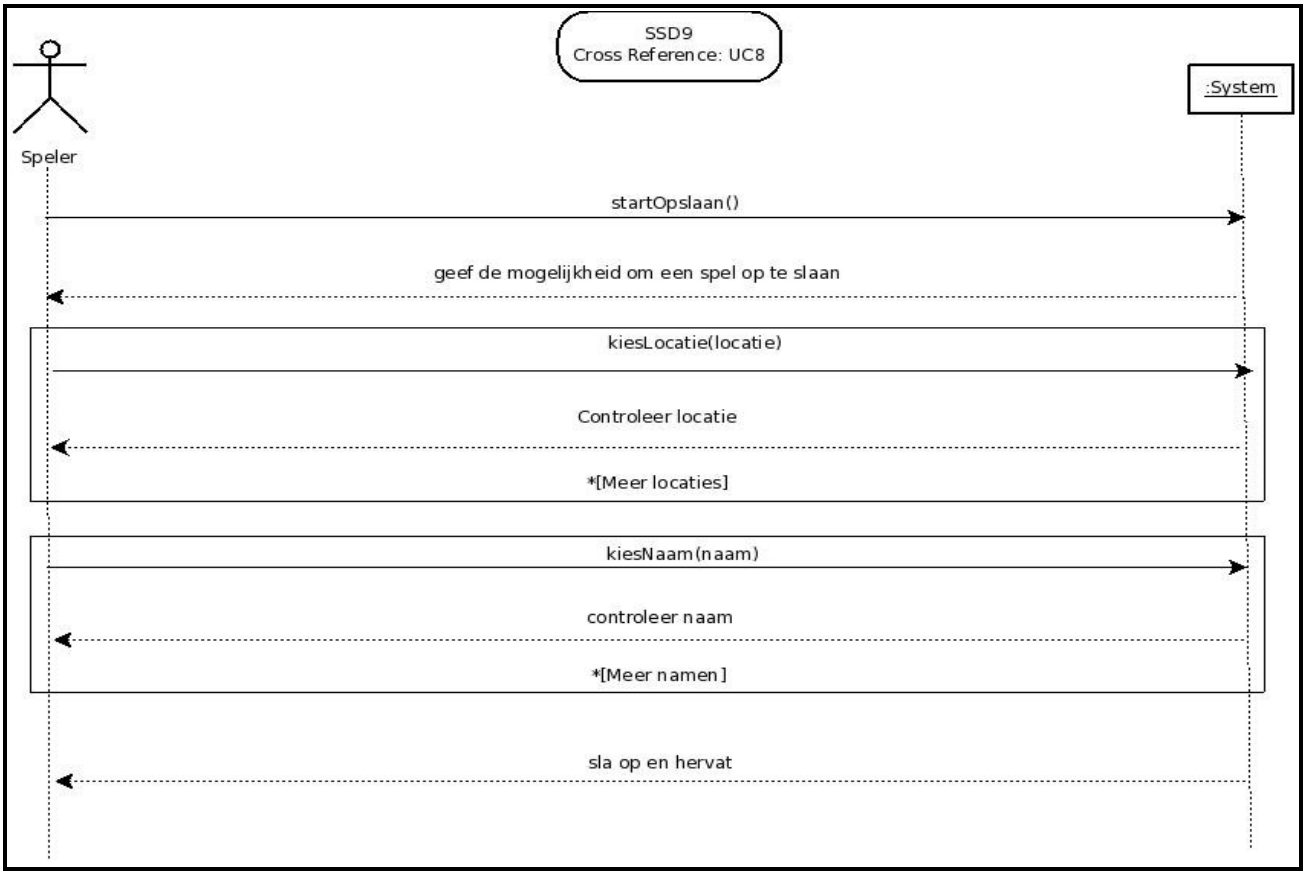

SSD4
Cross Reference: UC2 (alternative flow)
(tegel wordt gedraaid voor het plaatsen
+ er wordt geen pion geplaatst)
Speler
Speler
:System
vraagTegel()
Geeft een tegel en speelt geluid af
draaiTegel(richting)
Visualiseer draaiing
*[Meerdere rotaties]
plaatsTegel(tegel, plaats)
verifieert plaatsing
* [Ongeldige zet]
plaatsPion(false)
Geeft de beurt aan de volgende speler

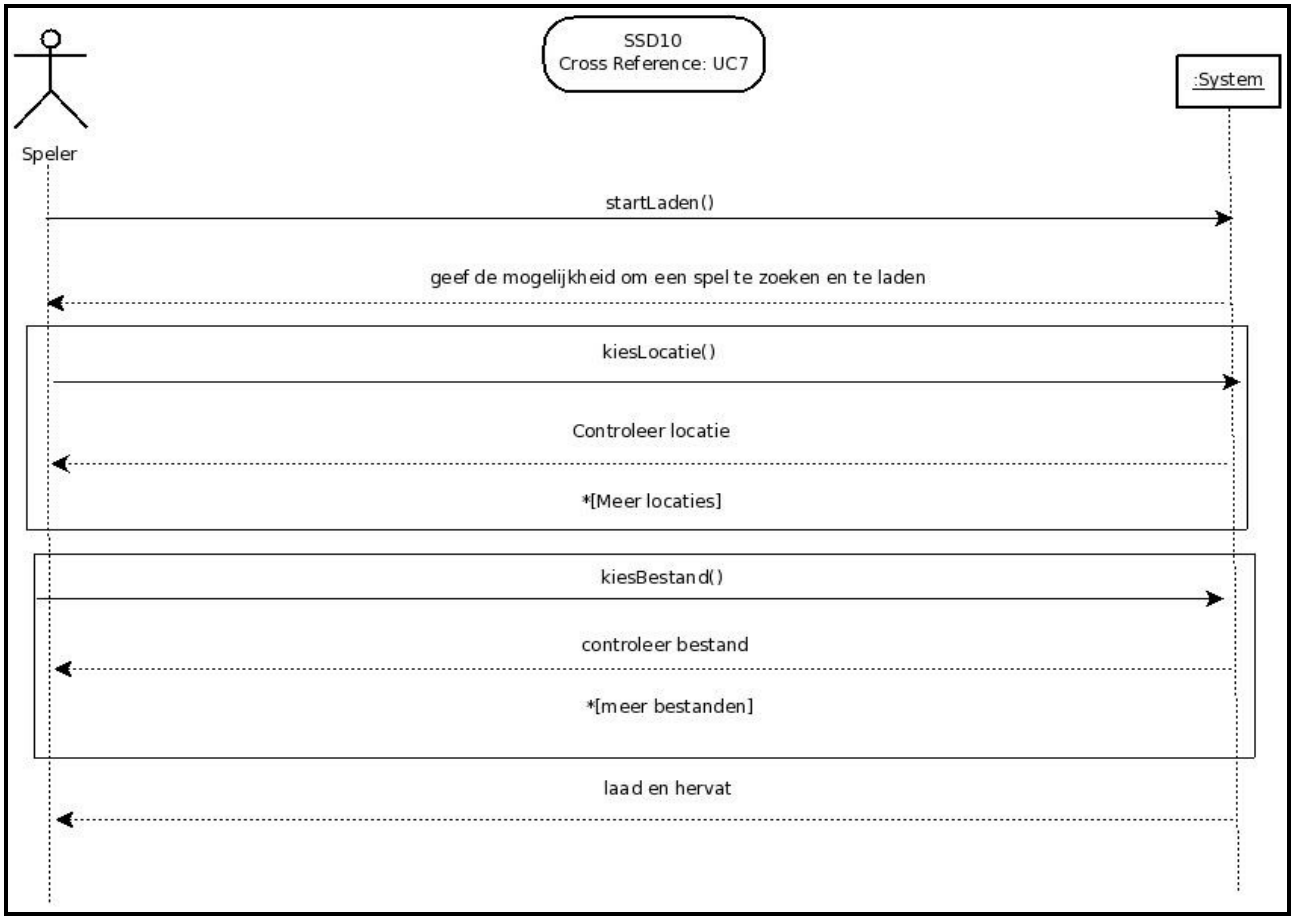

SSD5
Cross Reference: UC2 (alternative flow),
UC3 (pion wordt direct goed geplaatst)
Speler
Speler
:System
vraagTegel()
Geeft een tegel en speelt geluid af
plaatsTegel(tegel, plaats)
verifieert plaatsing
plaatsPion(true, plaats, pion)
verifieert plaatsing pion
Geeft de beurt aan de volgende speler

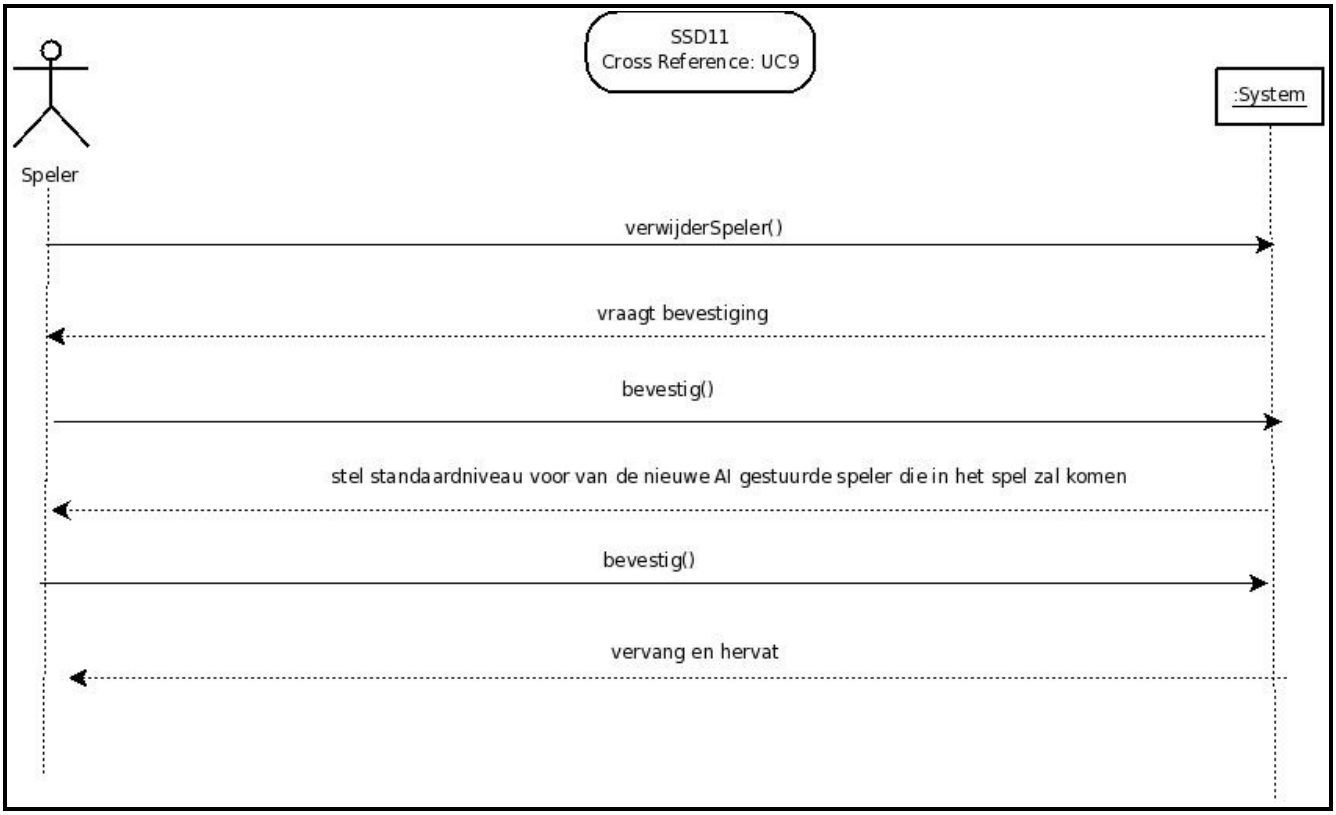

SSD6
Cross Reference: UC2 (alternative flow)
(tegel wordt geplaatst + er wordt geen pion geplaatst)
+ UC4 (de speler wil een pion terugnemen)
Speler
Speler
:System
vraagTegel()
Geeft een tegel en speelt geluid af
plaatsTegel(tegel, plaats)
verifieert plaatsing
plaatsPion(false)
neemPionTerug()
verifieert terugname
Past punten aan en geeft pion terug aan speler
Geeft de beurt aan de volgende speler

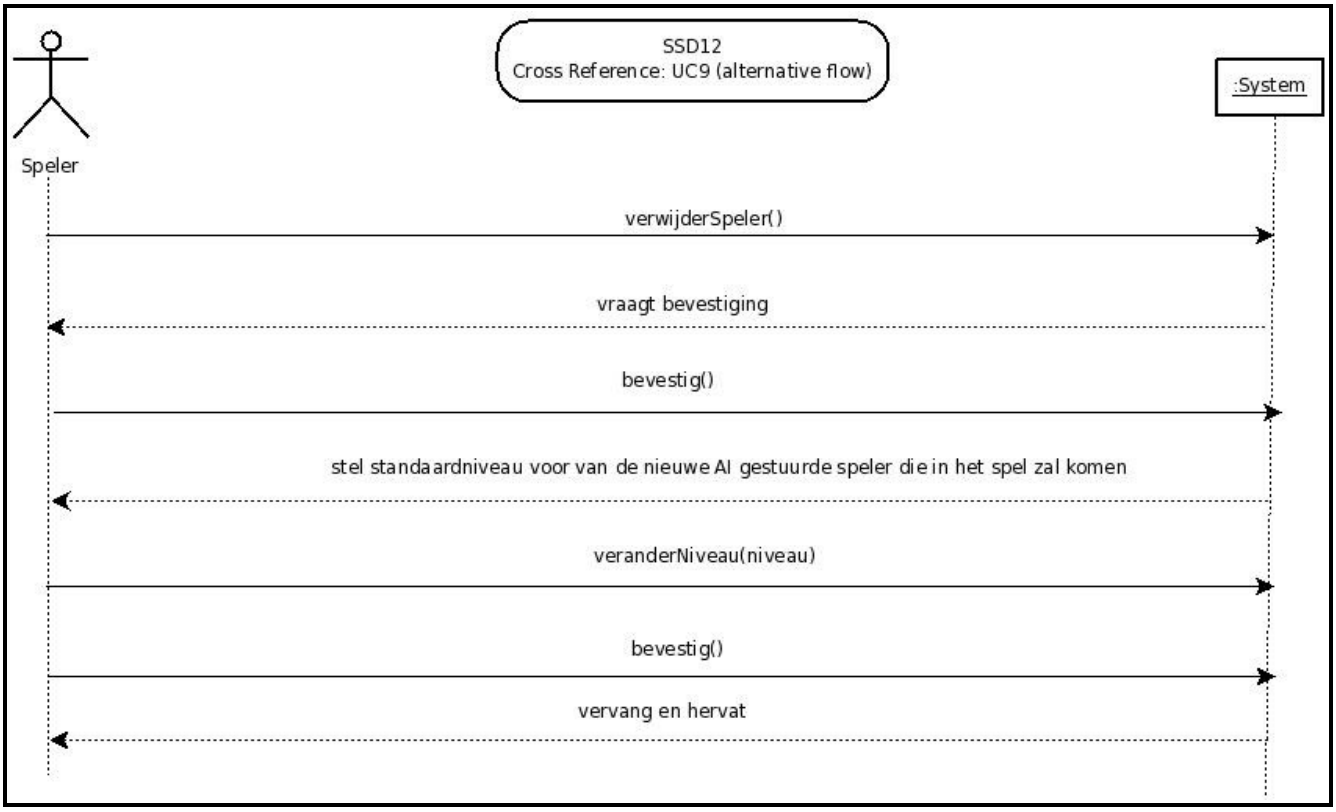

SSD7
Cross Reference: UC5
:System
Speler
vraagHelp()
Vraagt input
geefInput(zoekterm)
geeft informatie weer
Meer help
sluitHelp()
hervat

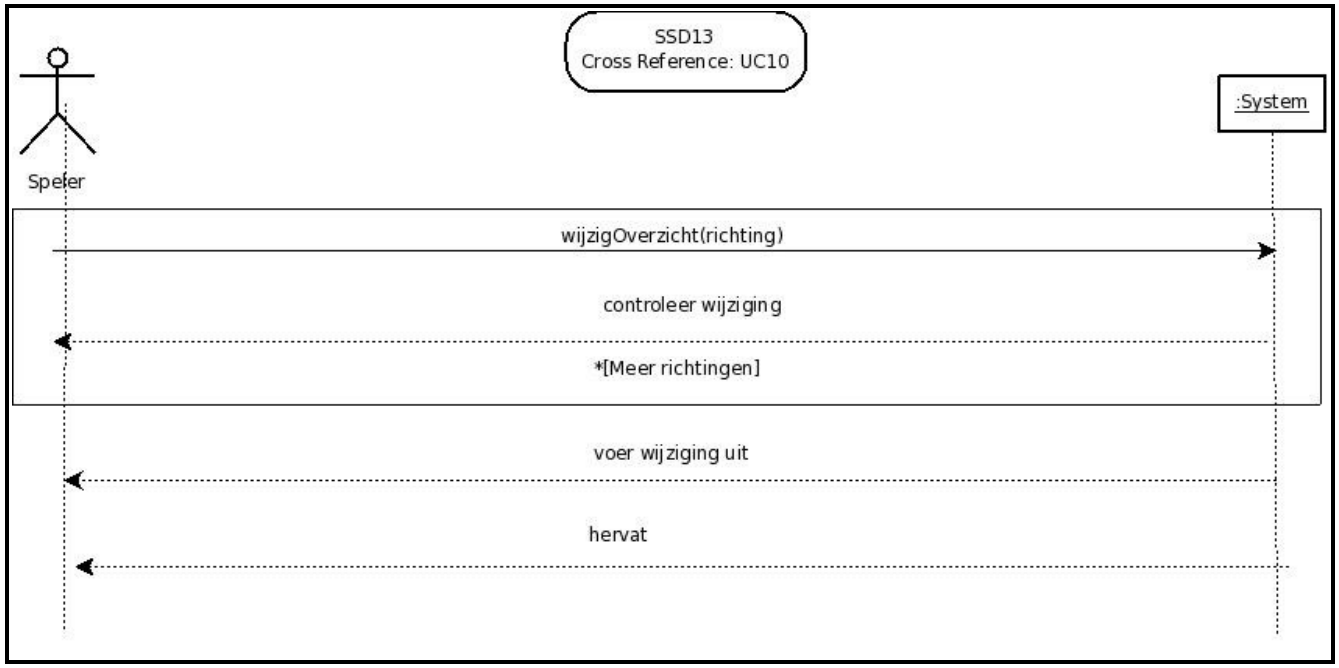

SSD8
Cross Reference: UC6
:System
Speler
vraagOpties()
Toont opties
veranderOptie(optie, value)
Controleer optie
*[Meer opties]
keurGoed()
Toon veranderingen
Hervat


SSD9
Cross Reference: UC8
Speler
:System
startOpslaan()
geef de mogelijkheid om een spel op te slaan
kiesLocatie(locatie)
Controleer locatie
*[Meer locaties]
kiesNaam(naam)
controleer naam
*[Meer namen]
sla op en hervat

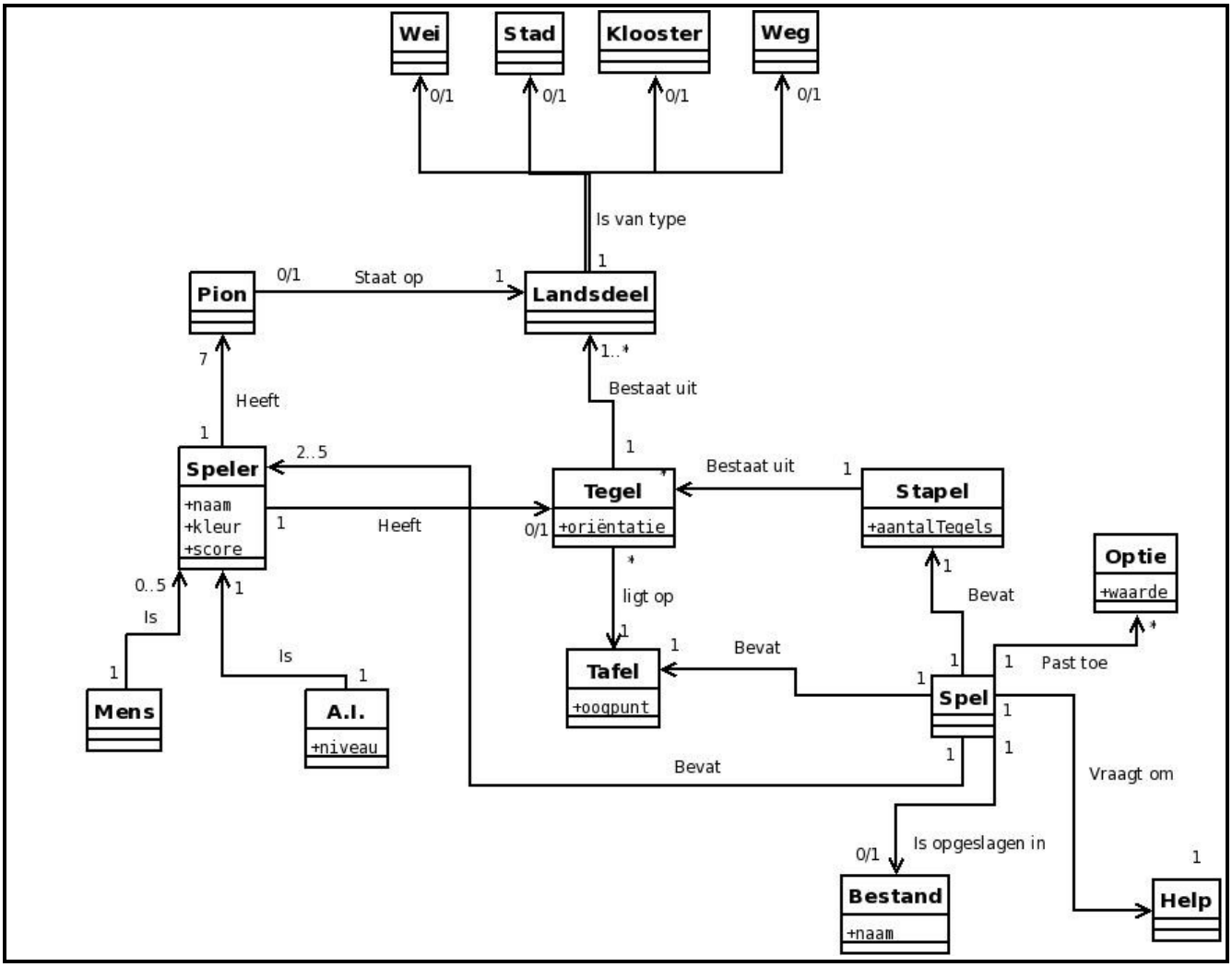

SSD10
Cross Reference: UC7
Speler
:System
startLaden()
geef de mogelijkheid om een spel te zoeken en te laden
kiesLocatie()
Controleer locatie
*[Meer locaties]
kiesBestand()
controleer bestand
*[meer bestanden]
laad en hervat


SSD11
Cross Reference: UC9
:System
Speler
verwijderSpeler()
vraagt bevestiging
bevestig()
stel standaardniveau voor van de nieuwe AI gestuurde speler die in het spel zal komen
bevestig()
vervang en hervat


SSD12
Cross Reference: UC9 (alternative flow)
:System
Speler
verwijderSpeler()
vraagt bevestiging
bevestig()
stel standaardniveau voor van de nieuwe AI gestuurde speler die in het spel zal komen
veranderNiveau(niveau)
bevestig()
vervang en hervat


SSD13
Cross Reference: UC10
:System
Speler
wijzigOverzicht(richting)
controleer wijziging
*[Meer richtingen]
voer wijziging uit
hervat

Domeinmodel:



Contracten:

Contract bij UC1: Nieuw spel starten	
Name	initialiseerNieuwSpel()
Cross References	UC1, SSD1 en SSD2
Pre-conditions	De applicatie is opgestart en de gebruiker heeft aangegeven een nieuw spel te willen starten.
Post-conditions	Een UI met de verschillende instellingsmogelijkheden is weergegeven. (instance creation) Een Spel is geïnitialiseerd. (instance creation)

Contract bij UC1: Nieuw spel starten	
Name	startSpel(instellingen)
Cross References	UC1, SSD1, SSD2
Pre-conditions	Applicatie is opgestart en spel is geïnitialiseerd.
Post-conditions	Wei, Stad, Klooster, Weg-instanties zijn aangemaakt (instance creation) Landsdeel-instanties zijn aangemaakt (instance creation, association formed (Wei, Stad, Klooster, Weg)) Tafel is aangemaakt. (instance creation, attribute modification (Tafel.oogpunt)) Tegel-instanties zijn aangemaakt. (instance creation, association formed (Landsdeel), attribute modification (Tegel.oriëntatie)) Stapel tegels is aangemaakt. (instance creation, association formed (Tegel), attribute modification (Stapel.aantalTegels)) Pion-instanties zijn aangemaakt. (instance creation) Speler-instanties zijn aangemaakt. (instance creation, association formed (Pion), attribute modification(Speler.naam, Speler.kleur, Speler.score)) Meerdere Mens-instanties zijn aangemaakt. (instance creation, association formed (Speler)) Meerdere AI-instanties zijn ingesteld. (instance creation, association formed (Speler), attribute modification (AI.niveau)) Optie-instanties zijn aangemaakt (instance creation, attribute modification (Optie.waarde)) Spel is aangemaakt met zijn instellingen. (instance creation, association formed (Optie), association formed (Stapel), association formed (Tafel), association formed (Speler)) Een starttegel is gelegd. (association broken (Stapel), association formed (Tafel))

Contract bij UC1: Starttegel niet aanvaarden	
Name	aanvaardTegel(false)
Cross References	UC1, SSD1, SSD2
Pre-conditions	Spel is gestart, maar er is nog geen spelbeurt afgelegd.
Post-conditions	De huidige starttegel is teruggelegd in Stapel. (association broken (Tafel), association formed (Stapel)) Een nieuwe starttegel is gelegd. (association broken (Stapel), association formed (Tafel))

Contract bij UC2: Beurt afleggen	
Name	vraagTegel()
Cross References	UC2, SSD3, SSD4, SSD5, SSD6
Pre-conditions	Spel is gestart en Speler is aan beurt.
Post-conditions	Stapel.aantalTegels is met 1 verminderd. (attribute modification) Speler heeft Tegel gekregen. (association broken (Stapel), association formed (Speler))

Contract bij UC2: Beurt afleggen	
Name	draaiTegel(richting)
Cross References	UC2, SSD4
Pre-conditions	Spel is gestart en Speler is aan beurt en Speler heeft tegel.
Post-conditions	Tegel.oriëntatie is veranderd conform de opgegeven richting. (attribute modification)

Contract bij UC2: Beurt afleggen	
Name	plaatsTegel(tegel,plaats)
Cross References	UC2, SSD3, SSD4, SSD5, SSD6
Pre-conditions	Spel is gestart en Speler is aan beurt en Speler heeft tegel. Plaats is geldig voor Tegel.
Post-conditions	Tegel is gelegd op Tafel. (association broken (Speler), association formed (Tafel))

Contract bij UC3: Pion plaatsen	
Name	plaatsPion(landsdeel)
Cross References	UC2, UC3, SSD3, SSD4, SSD5, SSD6
Pre-conditions	Spel is gestart en Speler is aan beurt en Speler heeft Tegel gelegd. Pion kan op het gewenste landsdeel staan.
Post-conditions	Pion is op een landsdeel gelegd. (association formed (Landsdeel))

Contract bij UC4: Pion terugnemen	
Name	neemPionTerug(landsdeel)
Cross References	UC2, UC3, UC4, SSD6
Pre-conditions	Spel is gestart en Speler is aan beurt en Speler heeft tegel gelegd en ooit eens een pion gezet. Een pion kan teruggenomen worden.
Post-conditions	Pion is teruggenomen. (association broken (Landsdeel)) Speler.score werd aangepast. (attribute modification)

Contract bij UC5: Help vragen	
Name	vraagHelp()
Cross References	UC5, SSD7
Pre-conditions	De help is opgestart en de speler wenst informatie over een bepaald onderwerp.
Post-conditions	De speler heeft zijn informatie gekregen (instance creation)

Contract bij UC5: Help vragen	
Name	geefInput(zoekterm)
Cross References	UC5, SSD7
Pre-conditions	De help instantie is aangemaakt en de speler heeft een zoekterm ingeven. Deze wordt doorgestuurd naar de database van de applicatie.
Post-conditions	De resultaten voor de opgegeven zoekterm zijn weergegeven. (attribute modification)

Contract bij UC5: Help vragen	
Name	sluitHelp()
Cross References	UC5, SSD7
Pre-conditions	Er is een help instantie aangemaakt. De speler heeft de gewenste informatie gekregen en wil hervatten waar hij/zij mee bezig was.
Post-conditions	De help instantie is afgesloten. (instance deletion)

Contract bij UC6: Opties instellen	
Name	keurGoed()
Cross References	UC6, SSD8
Pre-conditions	De applicatie is opgestart en de gebruiker heeft opties gewijzigd.
Post-conditions	De veranderingen werden toegepast op het huidige spel en de daarop volgende spellen. (attribute modification)

Contract bij UC6: Opties instellen	
Name	vraagOpties()
Cross References	UC6, SSD8
Pre-conditions	De applicatie is opgestart en de gebruiker heeft aangegeven om opties te willen wijzigen.
Post-conditions	Een UI met de verschillende instellingsmogelijkheden is weergegeven. (instance creation)

Contract bij UC6: Opties instellen	
Name	veranderOptie(optie, value)
Cross References	UC6 en SSD8
Pre-conditions	De applicatie is opgestart en de instellingsmogelijkheden zijn weergegeven.
Post-conditions	De veranderingen werden toegepast op het huidige spel en de daaropvolgende spellen. (attribute modification (Optie.waarde))

Contract bij UC7: Spel laden	
Name	startLaden()
Cross References	UC7 en SSD10
Pre-conditions	Een spel is gestart en speler wil spel laden en het bestand is geldig.
Post-conditions	Wei, Stad, Klooster, Weg-instanties zijn aangemaakt (instance creation) Landsdeel-instanties zijn aangemaakt (instance creation, association formed (Wei, Stad, Klooster, Weg)) Tafel is aangemaakt. (instance creation, attribute modification (Tafel.oogpunt)) Tegel-instanties zijn aangemaakt. (instance creation, association formed (Landsdeel), attribute modification (Tegel.oriëntatie)) Stapel tegels is aangemaakt. (instance creation, association formed (Tegel), attribute modification (Stapel.aantalTegels)) Pion-instanties zijn aangemaakt. (instance creation) Speler-instanties zijn aangemaakt. (instance creation, association formed (Pion), attribute modification(Speler.naam, Speler.kleur, Speler.score)) Meerdere Mens-instanties zijn aangemaakt. (instance creation, association formed (Speler)) Meerdere AI-instanties zijn ingesteld. (instance creation, association formed (Speler), attribute modification (AI.niveau)) Spel is aangemaakt met zijn instellingen. (instance creation, association formed (Optie), association formed (Stapel), association formed (Tafel), association formed (Speler)) Spel is gelinkt aan Bestand. (association formed (Spel <-> Bestand)) Tegels zijn gelegd. (association formed (Tafel)) Pionnen zijn geplaatst op landsdelen. (association formed (Pion <-> Landsdeel)) Speler.score-waarden zijn aangepast. (attribute modification)

Contract bij UC7: Spel laden	
Name	kiesLocatie()
Cross References	UC7 en SSD10
Pre-conditions	De speler wil een spel laden en heeft hiervoor de mogelijkheid gekregen van de applicatie. Hij kiest de locatie, waar het gewenste bestand zich bevindt.
Post-conditions	Bestand dat het spel bevat werd aangemaakt. (instance creation, association formed (Bestand <-> Spel), attribute modification (Bestand.naam))

Contract bij UC7: Spel laden	
Name	kiesBestand()
Cross References	UC7 en SSD10
Pre-conditions	Een speler wil een spel laden en heeft de laad-procedure opgestart en heeft reeds een bestandslocatie gekozen.
Post-conditions	Een controle op de geldigheid van het bestand (extentie) is uitgevoerd. Indien deze positief was: het spel dat opgeslagen was in het bestand, is ingeladen en het oude spel is afgebroken. (instance creation en deletion) Indien deze negatief was: een passende melding wordt gegeven. (mogelijk d.m.v. instance creation)

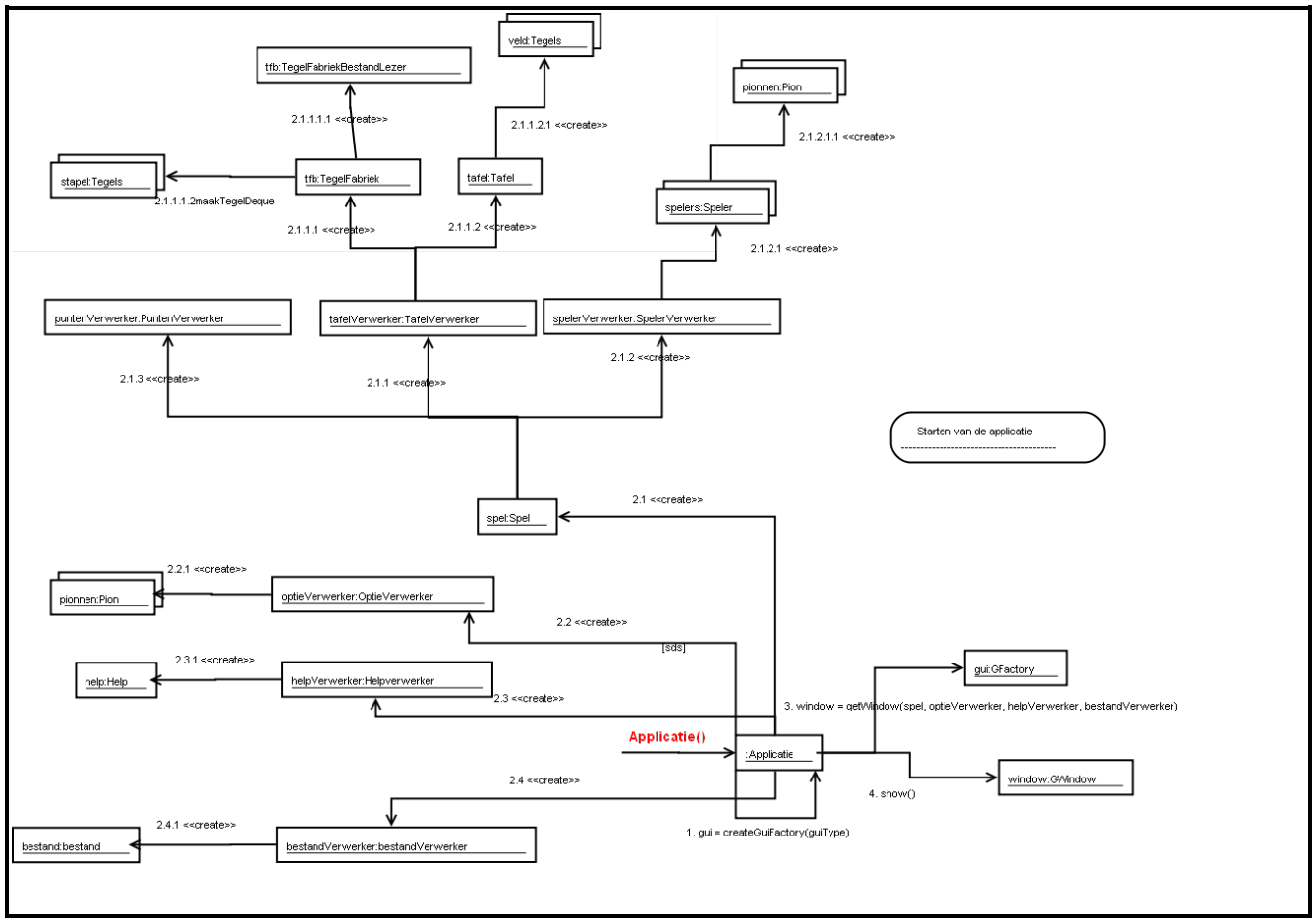
Contract bij UC8: Spel opslaan	
Name	startOpslaan()
Cross References	UC8 en SSD9
Pre-conditions	Een spel is bezig en Speler wil spel opslaan.
Post-conditions	Het systeem vraag om de naam van het bestand waar die mag opslaan.

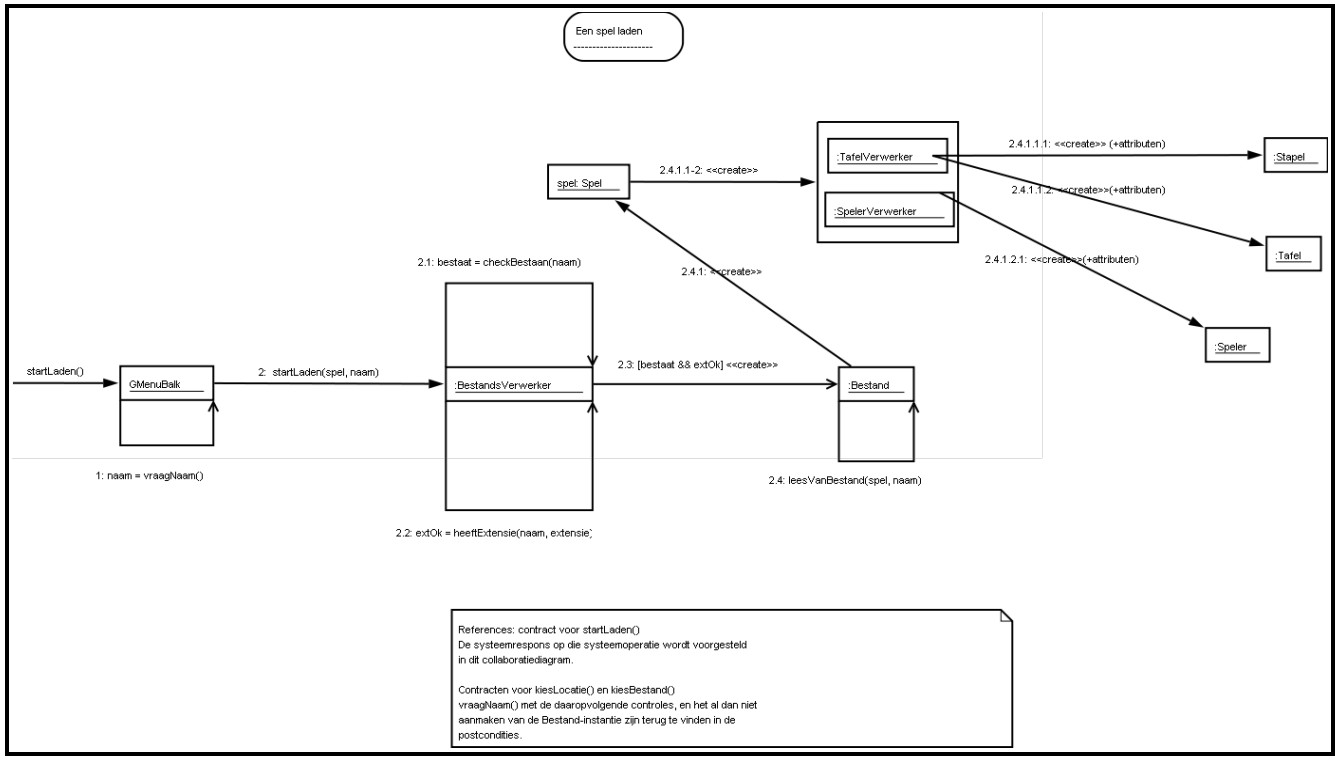
Contract bij UC8: Spel opslaan	
Name	kiesNaam(naam)
Cross References	UC8, SSD9
Pre-conditions	Een spel is bezig en een Speler wil Spel opslaan en de voor de gebruiker nodig zijnde UI is aanwezig.
Post-conditions	Een controle op de geldigheid van de naam is uitgevoerd. Indien deze positief was:een bestand-instantie wordt aangemaakt met naam (instance creation, attribute modification) Indien deze negatief was: er wordt een passende melding gegeven, als die positief is, wordt een bestand-instantie met naam aangemaakt. (instance creation, attribute modification)

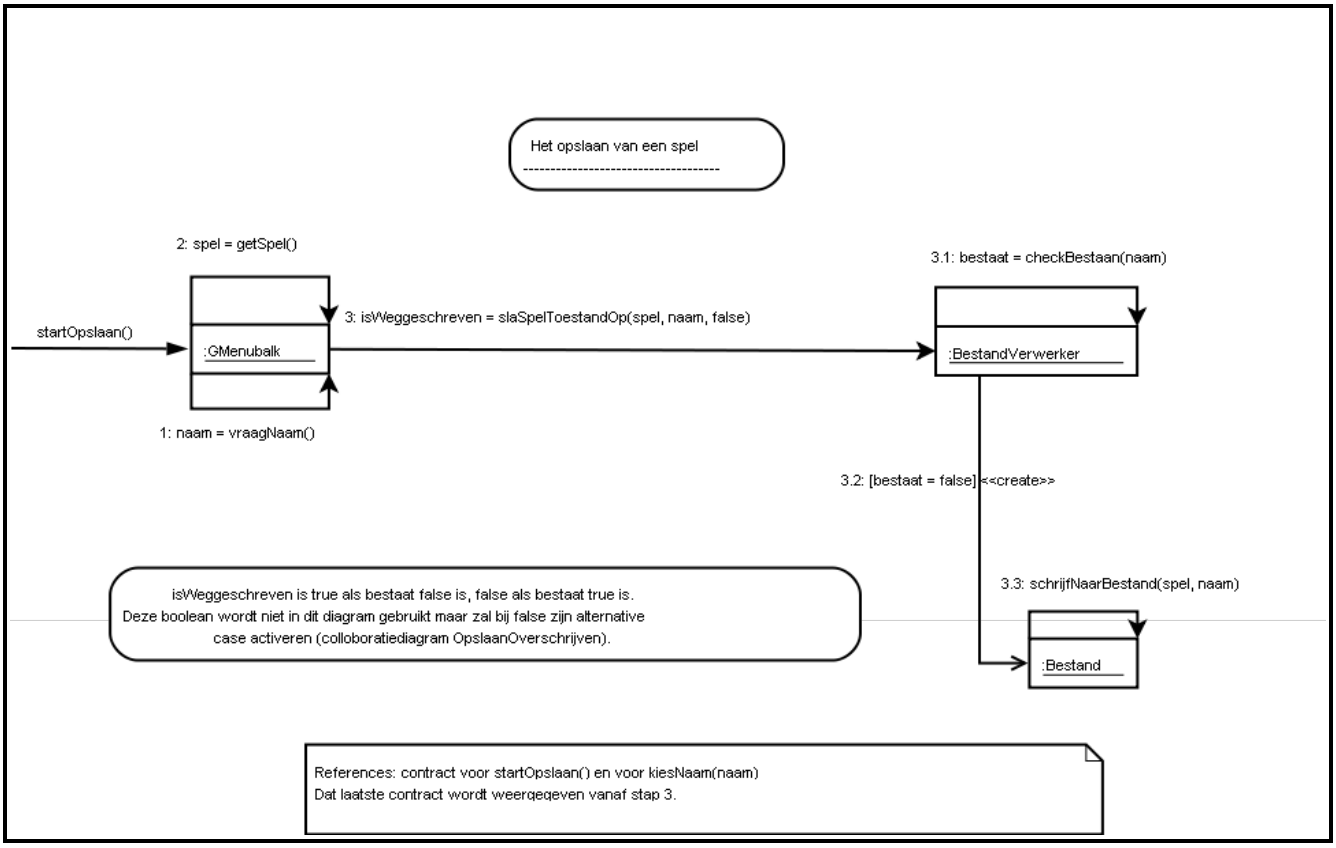
Contract bij UC9: Speler verwijderen	
Name	verwijderSpeler()
Cross References	UC9 en SSD11, SSD12
Pre-conditions	Een spel is bezig, en de huidige menselijke Speler wil zichzelf verwijderen.
Post-conditions	Nieuwe AI is aangemaakt met de attributen van de menselijke speler en een welbepaald niveau. (instance creation, attribute modification (AI.niveau)) AI is gelinkt met de Speler-instantie gelinkt aan Mens. (association formed (AI <-> Speler)) Mens is verwijderd. (instance deletion)

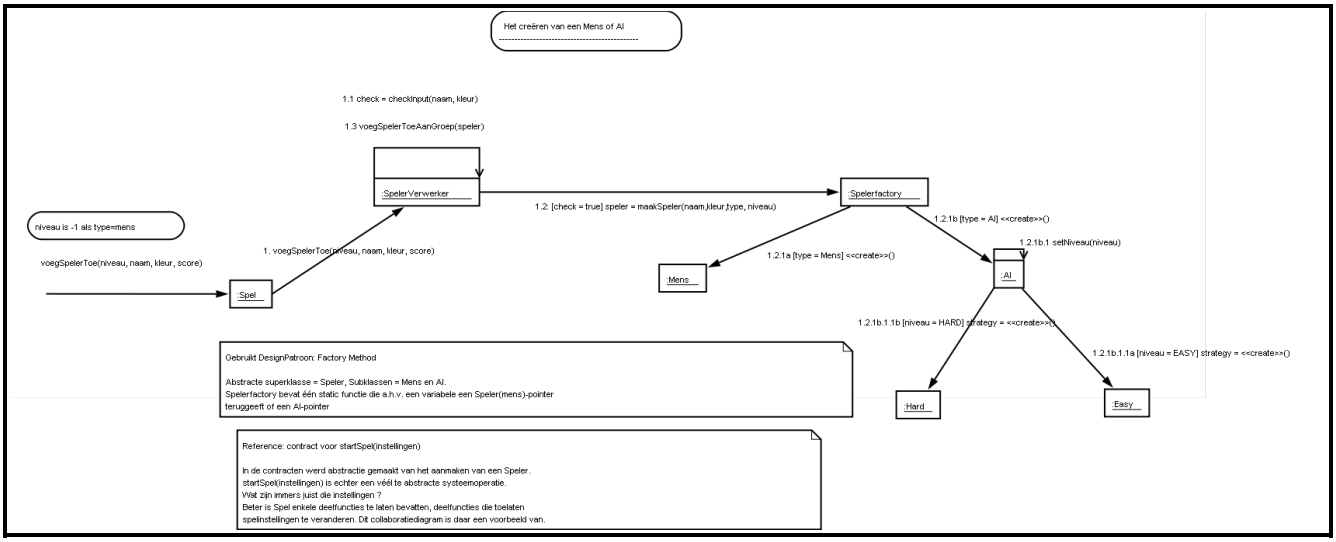
Contract bij UC10: Tafeloverzicht wijzigen	
Name	wijzigOverzicht(richting)
Cross References	UC10 en SSD13
Pre-conditions	Een spel is bezig en een speler wenst het overzicht over de tafel te wijzigen.
Post-conditions	Tafel.oogpunt is aangepast. (attribute modification)

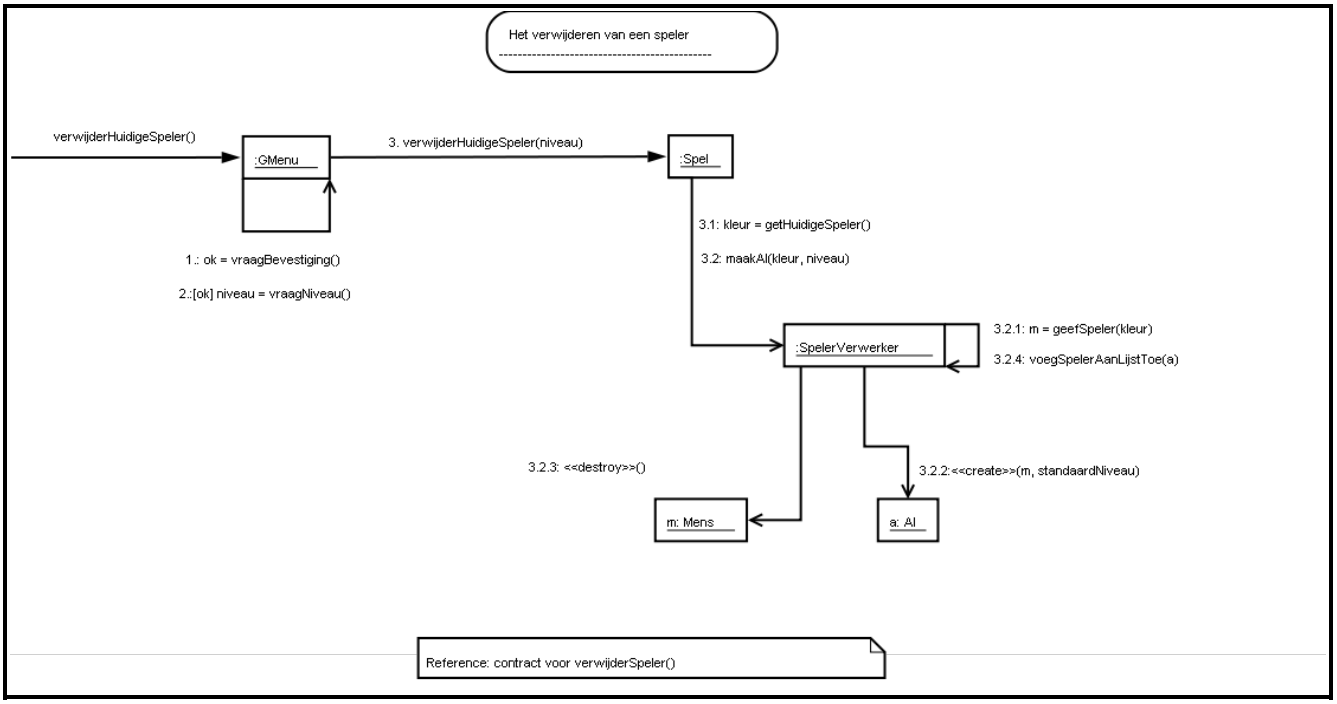
Collaboratiediagrammen:

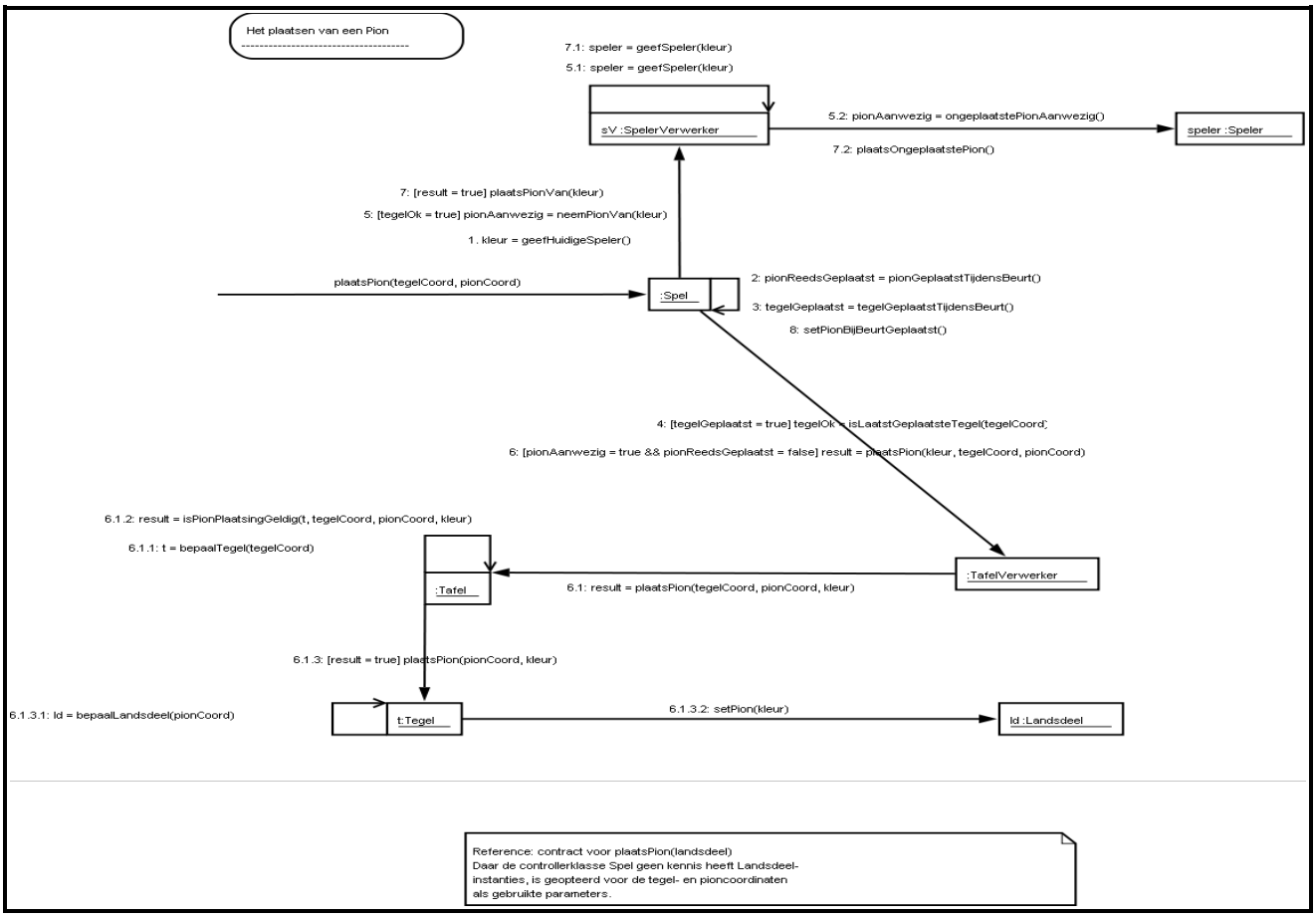


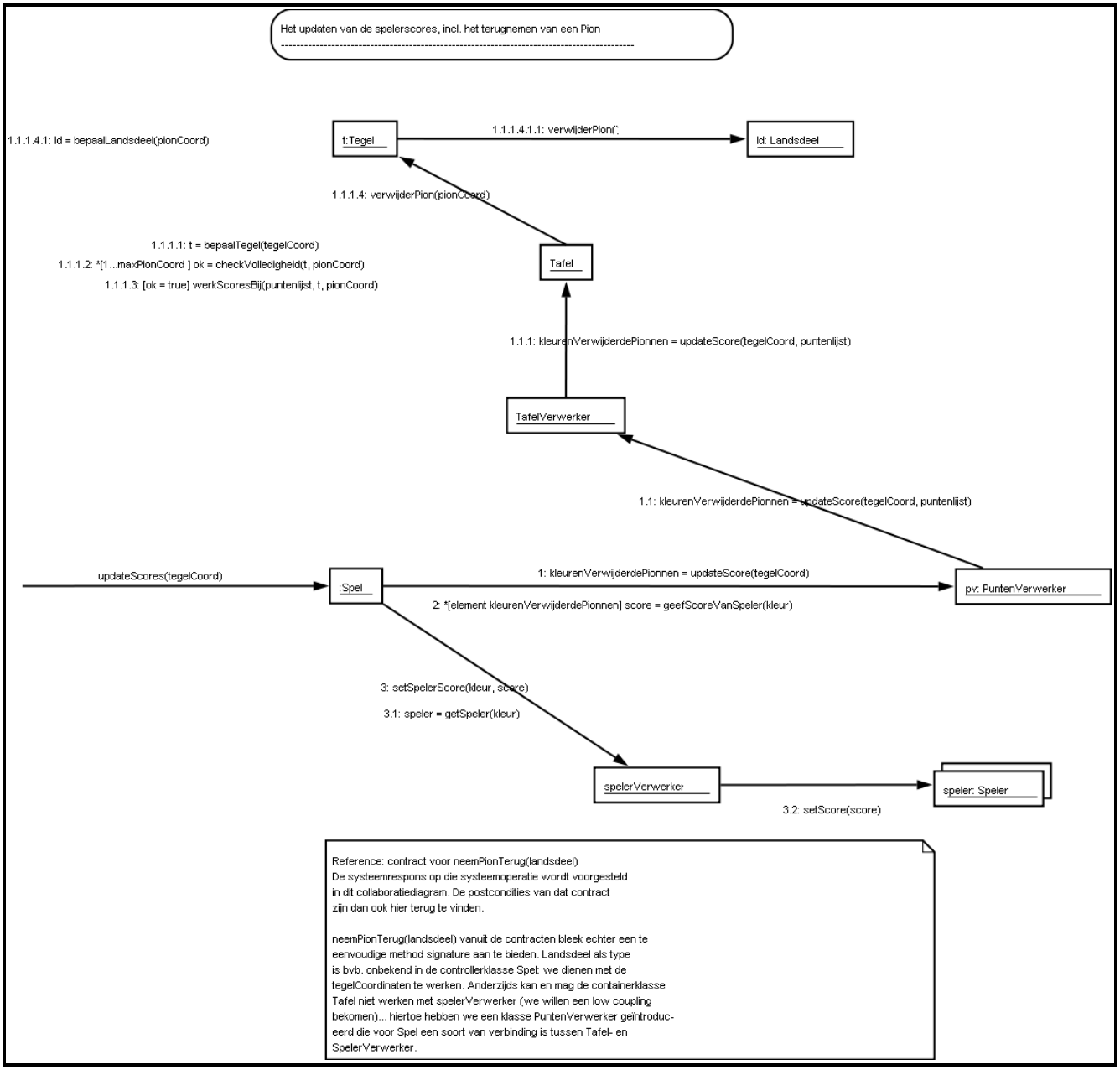


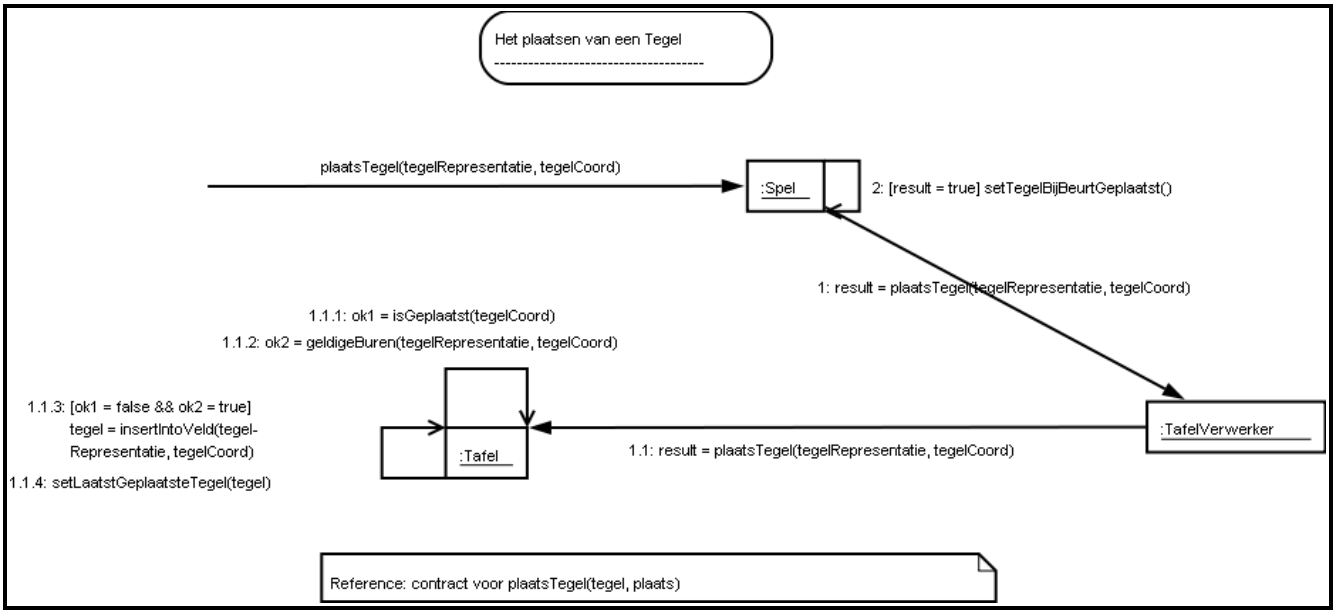

Het opslaan van een spel
2: spel = getSpel()
3.1: bestaat = checkBestaan(naam)
startOpslaan()
:GMenubalk
3: isWeggeschreven = slaSpelToestandOp(spel, naam, false)
:BestandVerwerker
1: naam = vraagNaam()
3.2: [bestaat = false] <<create>>
isWeggeschreven is true als bestaat false is, false als bestaat true is.
Deze boolean wordt niet in dit diagram gebruikt maar zal bij false zijn alternative
case activeren (colloboratiediagram OpslaanOverschrijven).
3.3: schrijfNaarBestand(spel, naam)
:Bestand
References: contract voor startOpslaan() en voor kiesNaam(naam)
Dat laatste contract wordt weergegeven vanaf stap 3.

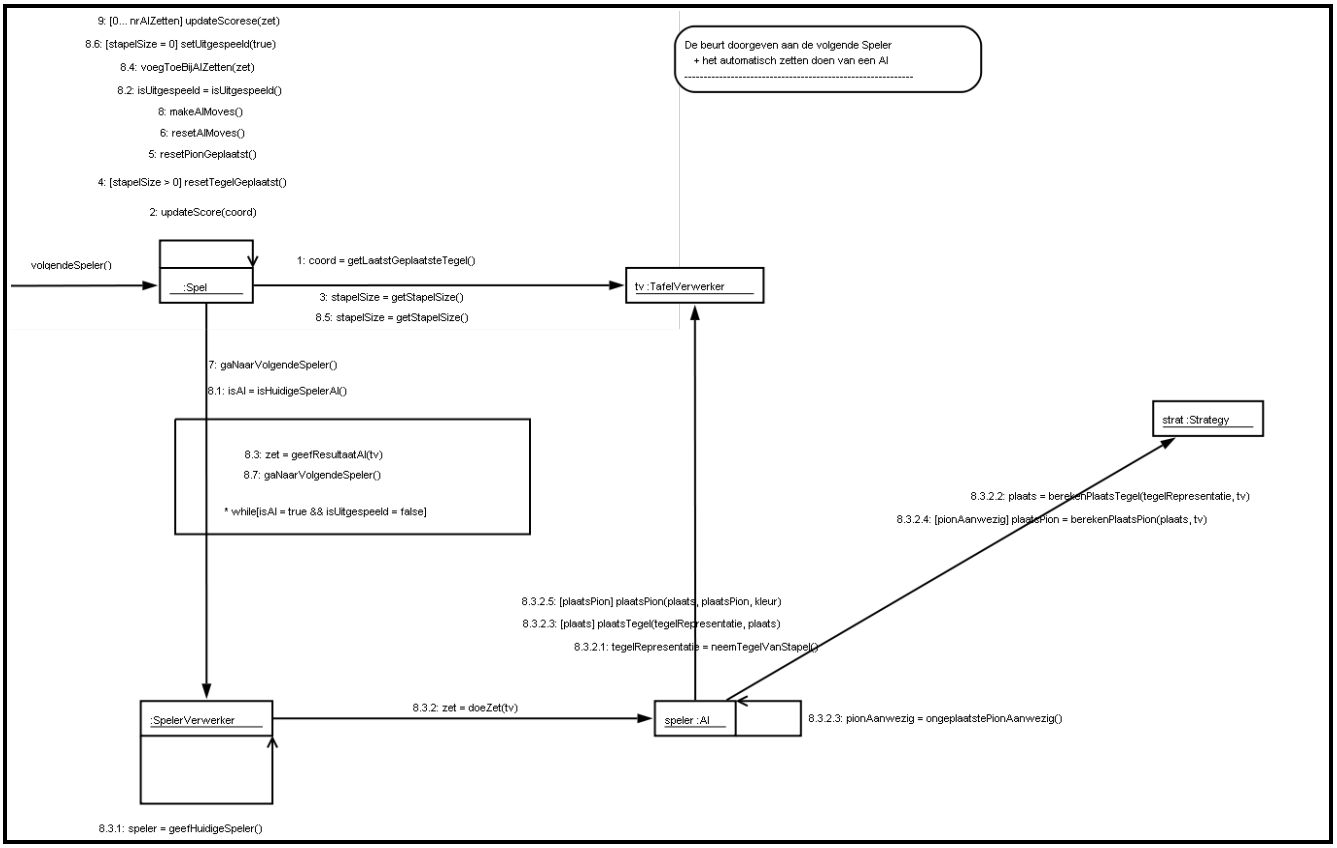

Het creëren van een Mens of AI
1.1 check = checkInput(naam, kleur)
1.3 voegSpelerToeAanGroep(speler)
:SpelerVerwerker
1.2: [check = true] speler = maakSpeler(naam,kleur,type, niveau)
:Spelerfactory
niveau is -1 als type=mens
1.2.1b [type = AI] <<create>>()
1.2.1b.1 setNiveau(niveau)
1. voegSpelerToe(niveau, naam, kleur, score)
1.2.1a [type = Mens] <<create>>()
voegSpelerToe(niveau, naam, kleur, score)
:Mens
:AI
:Spel
1.2.1b.1.1b [niveau = HARD] strategy = <<create>>()
1.2.1b.1.1a [niveau = EASY] strategy = <<create>>()
Gebruikt DesignPatroon: Factory Method
Abstracte superklasse = Speler, Subklassen = Mens en AI.
Spelerfactory bevat één static functie die a.h.v. een variabele een Speler(mens)-pointer
teruggeeft of een AI-pointer
:Hard
:Easy
Reference: contract voor startSpel(instellingen)
In de contracten werd abstractie gemaakt van het aanmaken van een Speler.
startSpel(instellingen) is echter een véél te abstracte systeemoperatie.
Wat zijn immers juist die instellingen ?
Beter is Spel enkele deelfuncties te laten bevatten, deelfuncties die toelaten
spelinstellingen te veranderen. Dit collaboratiediagram is daar een voorbeeld van.

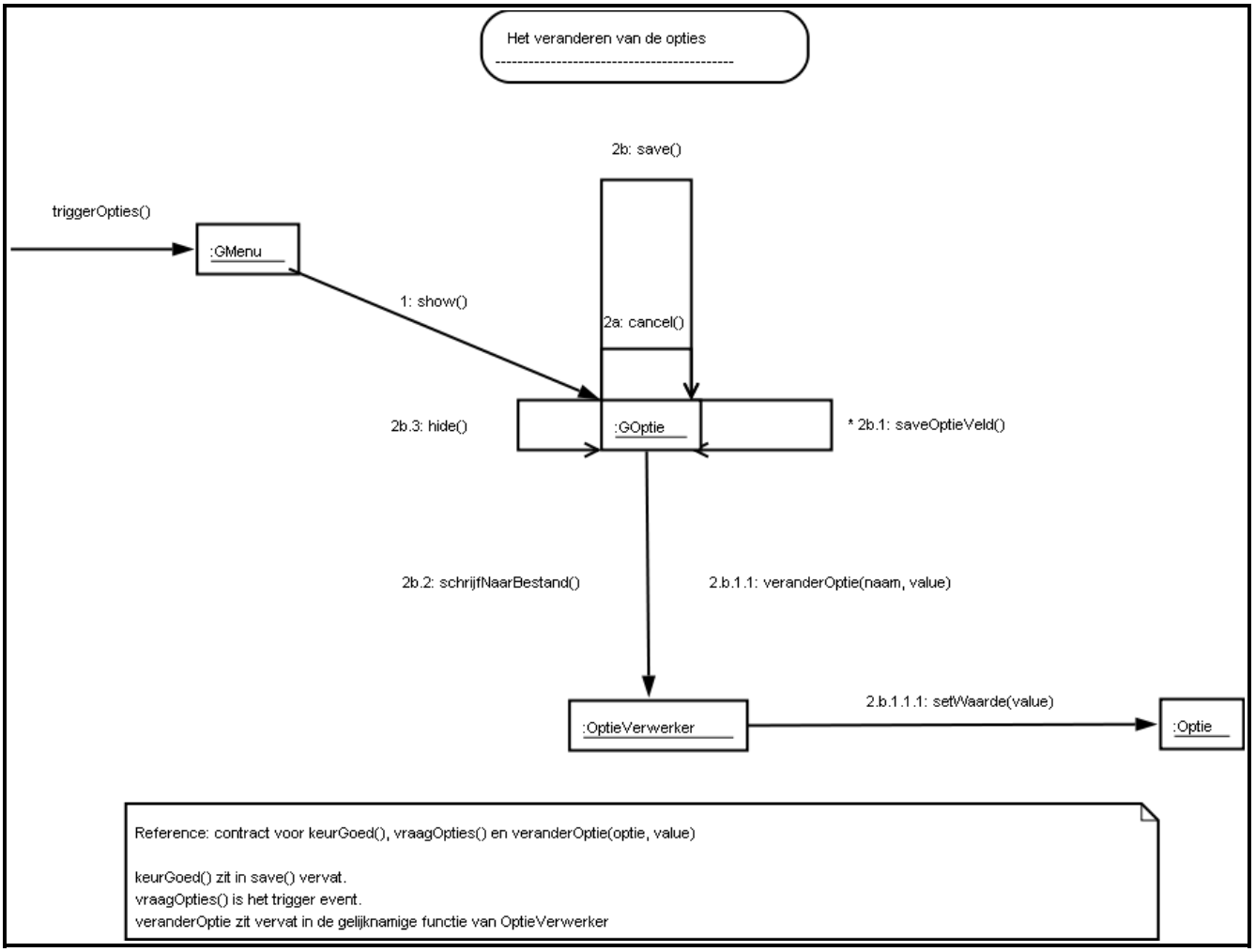

Het verwijderen van een speler
verwijderHuidigeSpeler()
:GMenu
3. verwijderHuidigeSpeler(niveau)
:Spel
1.: ok = vraagBevestiging()
2.:[ok] niveau = vraagNiveau()
3.1: kleur = getHuidigeSpeler()
3.2: maakAI(kleur, niveau)
:SpelerVerwerker
3.2.1: m = geefSpeler(kleur)
3.2.4: voegSpelerAanLijstToe(a)
3.2.3: <<destroy>>()
3.2.2:<<create>>(m, standaardNiveau)
m: Mens
a: AI
Reference: contract voor verwijderSpeler()

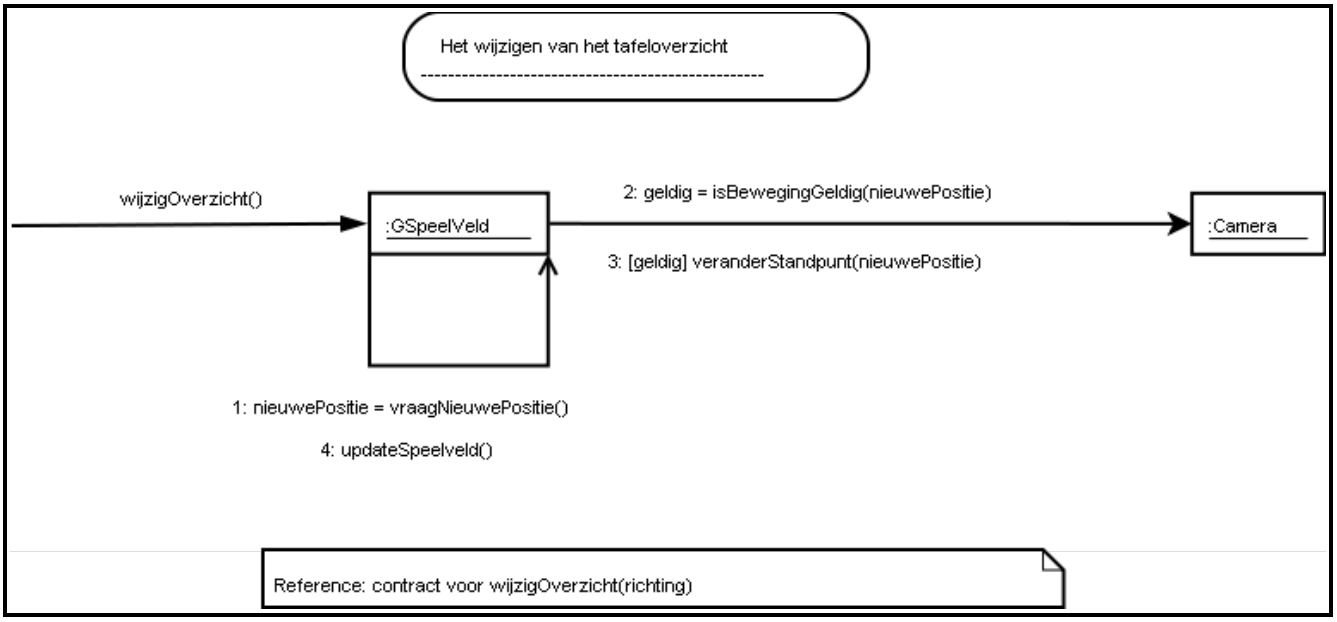

Het plaatsen van een Pion
7.1: speler = geefSpeler(kleur)
5.1: speler = geefSpeler(kleur)
sV :SpelerVerwerker
5.2: pionAanwezig = ongeplaatstePionAanwezig()
speler :Speler
7.2: plaatsOngeplaatstePion()
7: [result = true] plaatsPionVan(kleur)
5: [tegelOk = true] pionAanwezig = neemPionVan(kleur)
1. kleur = geefHuidigeSpeler()
plaatsPion(tegelCoord, pionCoord)
:Spel
2: pionReedsGeplaatst = pionGeplaatstTijdensBeurt()
3: tegelGeplaatst = tegelGeplaatstTijdensBeurt()
8: setPionBijBeurtGeplaatst()
4: [tegelGeplaatst = true] tegelOk = isLaatstGeplaatsteTegel(tegelCoord)
6: [pionAanwezig = true && pionReedsGeplaatst = false] result = plaatsPion(kleur, tegelCoord, pionCoord)
6.1.2: result = isPionPlaatsingGeldig(t, tegelCoord, pionCoord, kleur)
6.1.1: t = bepaalTegel(tegelCoord)
:Tafel
6.1: result = plaatsPion(tegelCoord, pionCoord, kleur)
:TafelVerwerker
6.1.3: [result = true] plaatsPion(pionCoord, kleur)
6.1.3.1: ld = bepaalLandsdeel(pionCoord)
t:Tegel
6.1.3.2: setPion(kleur)
ld :Landsdeel
Reference: contract voor plaatsPion(landsdeel)
Daar de controllerklasse Spel geen kennis heeft Landsdeel-
instanties, is geopteerd voor de tegel- en pioncoordinaten
als gebruikte parameters.

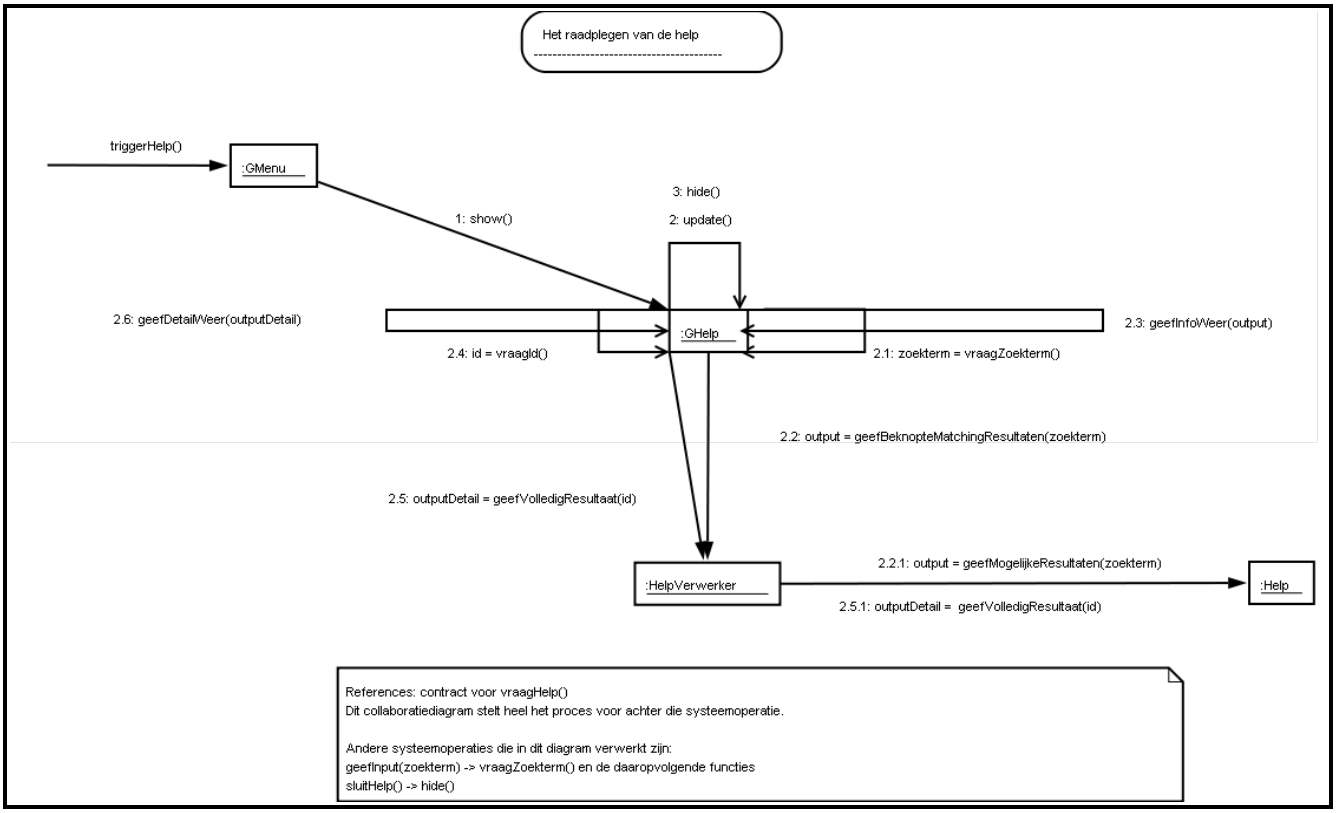

Het updaten van de spelerscores, incl. het terugnemen van een Pion
1.1.1.4.1: Id = bepaalLandsdeel(pionCoord)
t:Tegel
1.1.1.4.1.1: verwijderPion(
Id: Landsdeel
1.1.1.4: verwijderPion(pionCoord)
1.1.1.1: t = bepaalTegel(tegelCoord)
1.1.1.2: *[1...maxPionCoord] ok = checkVolledigheid(t, pionCoord)
1.1.1.3: [ok = true] werkScoresBij(puntenlijst, t, pionCoord)
Tafel
1.1.1: kleurenVerwijderdePionnen = updateScore(tegelCoord, puntenlijst)
TafelVerwerker
1.1: kleurenVerwijderdePionnen = updateScore(tegelCoord, puntenlijst)
updateScores(tegelCoord)
:Spel
1: kleurenVerwijderdePionnen = updateScore(tegelCoord)
pv: PuntenVerwerker
2: *[element kleurenVerwijderdePionnen] score = geefScoreVanSpeler(kleur)
3: setSpelerScore(kleur, score)
3.1: speler = getSpeler(kleur)
spelerVerwerker
speler: Speler
3.2: setScore(score)
Reference: contract voor neemPionTerug(landsdeel)
De systeemrespons op die systeemoperatie wordt voorgesteld
in dit collaboratiediagram. De postcondities van dat contract
zijn dan ook hier terug te vinden.
neemPionTerug(landsdeel) vanuit de contracten bleek echter een te
eenvoudige method signature aan te bieden. Landsdeel als type
is bvb. onbekend in de controllerklasse Spel: we dienen met de
tegelCoordinaten te werken. Anderzijds kan en mag de containerklasse
Tafel niet werken met spelerVerwerker (we willen een low coupling
bekomen)... hiertoe hebben we een klasse PuntenVerwerker geïntroduc-
eerd die voor Spel een soort van verbinding is tussen Tafel- en
SpelerVerwerker.

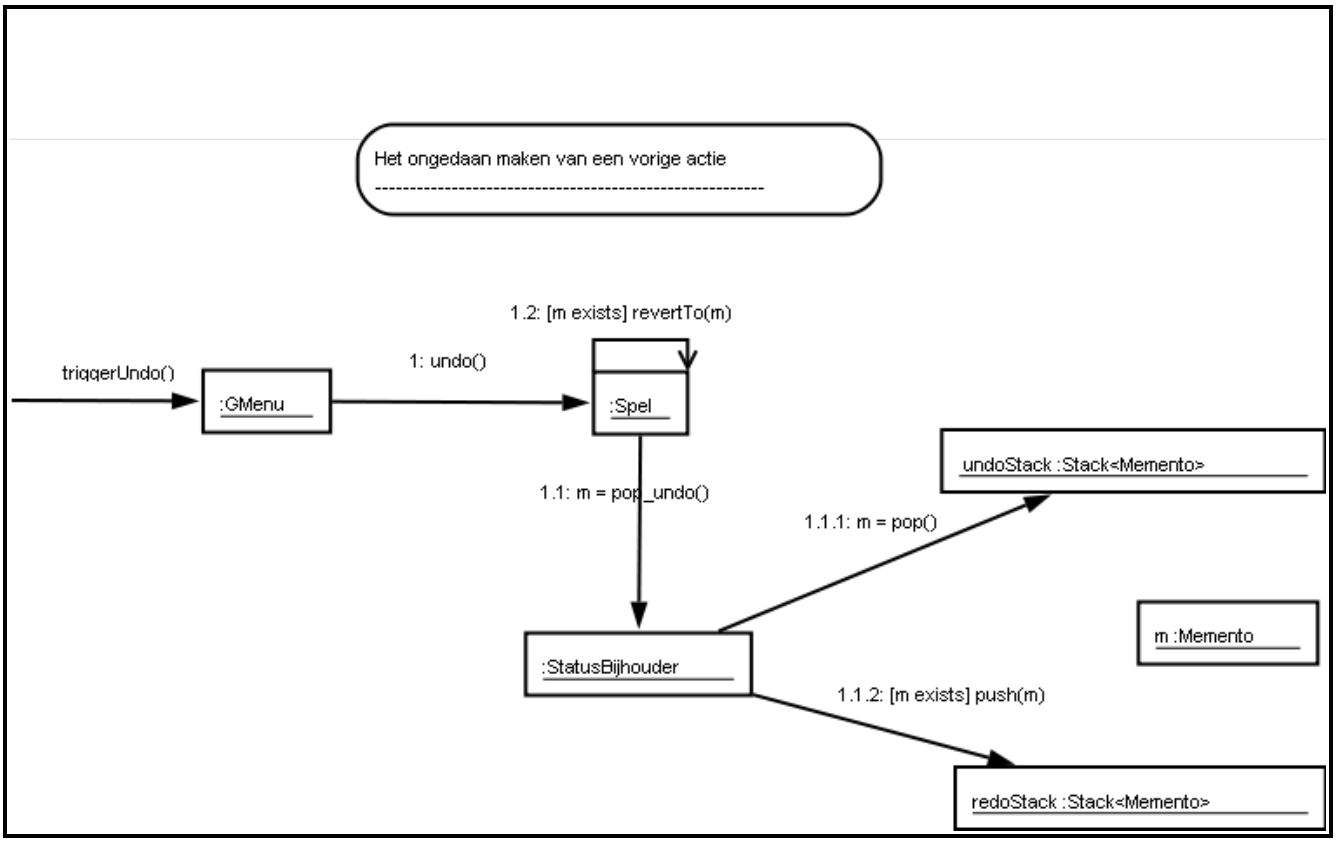

Het plaatsen van een Tegel
plaatsTegel(tegelRepresentatie, tegelCoord)
:Spel
2: [result = true] setTegelBijBeurtGeplaatst()
1: result = plaatsTegel(tegelRepresentatie, tegelCoord)
1.1.1: ok1 = isGeplaatst(tegelCoord)
1.1.2: ok2 = geldigeBuren(tegelRepresentatie, tegelCoord)
1.1.3: [ok1 = false && ok2 = true]
tegel = insertIntoVeld(tegel-
Representatie, tegelCoord)
1.1.4: setLaatstGeplaatsteTegel(tegel)
:Tafel
:TafelVerwerker
1.1: result = plaatsTegel(tegelRepresentatie, tegelCoord)
Reference: contract voor plaatsTegel(tegel, plaats)

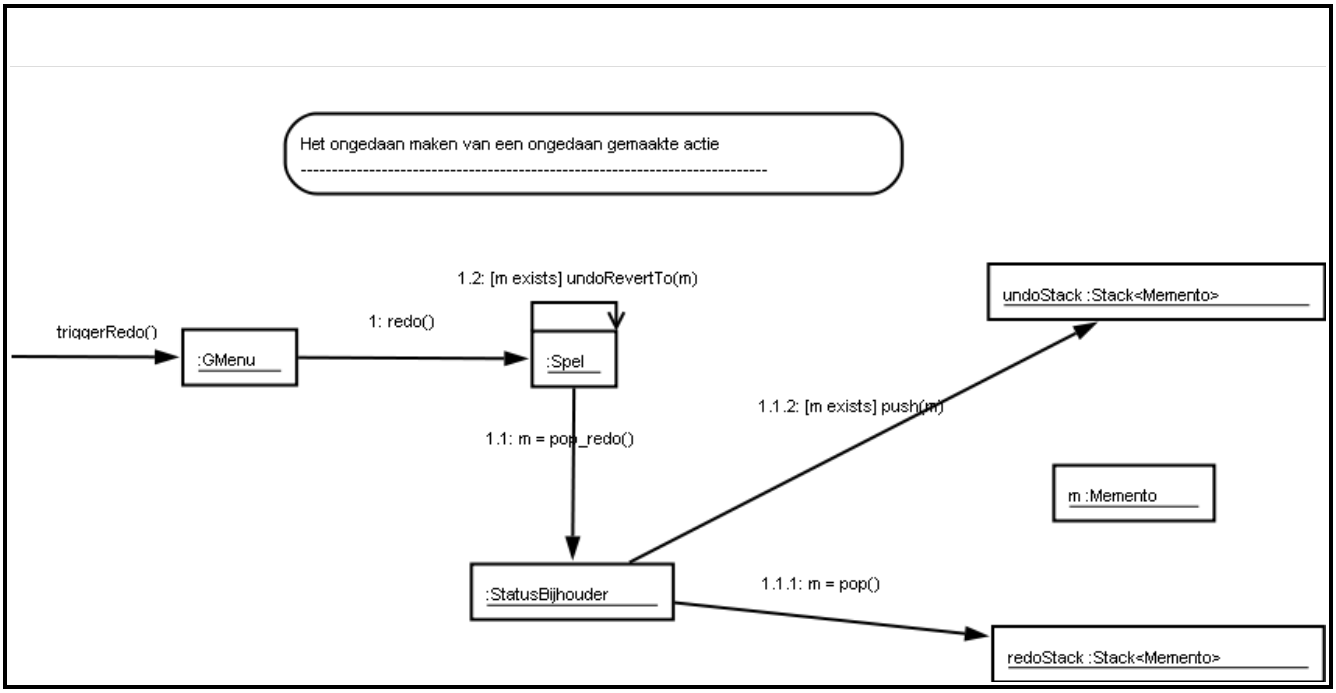

9: [0... nrAIZetten] updateScorese(zet)
8.6: [stapelSize = 0] setUitgespeeld(true)
8.4: voegToeBijAIZetten(zet)
8.2: isUitgespeeld = isUitgespeeld()
8: makeAIMoves()
6: resetAIMoves()
5: resetPionGeplaatst()
4: [stapelSize > 0] resetTegelGeplaatst()
2: updateScore(coord)
De beurt doorgeven aan de volgende Speler
+ het automatisch zetten doen van een AI
volgendeSpeler()
:Spel
1: coord = getLaatstGeplaatsteTegel()
tv :TafelVerwerker
3: stapelSize = getStapelSize()
8.5: stapelSize = getStapelSize()
7: gaNaarVolgendeSpeler()
8.1: isAI = isHuidigeSpelerAI()
8.3: zet = geefResultaatAI(tv)
8.7: gaNaarVolgendeSpeler()
* while[isAI = true && isUitgespeeld = false]
strat :Strategy
8.3.2.2: plaats = berekenPlaatsTegel(tegelRepresentatie, tv)
8.3.2.4: [pionAanwezig] plaatsPion = berekenPlaatsPion(plaats, tv)
8.3.2.5: [plaatsPion] plaatsPion(plaats, plaatsPion, kleur)
8.3.2.3: [plaats] plaatsTegel(tegelRepresentatie, plaats)
8.3.2.1: tegelRepresentatie = neemTegelVanStapel()
:SpelerVerwerker
8.3.2: zet = doeZet(tv)
speler :AI
8.3.2.3: pionAanwezig = ongeplaatstePionAanwezig()
8.3.1: speler = geefHuidigeSpeler()


Het veranderen van de opties
2b: save()
triggerOpties()
:GMenu
1: show()
2a: cancel()
2b.3: hide()
:GOptie
* 2b.1: saveOptieVeld()
2b.2: schrijfNaarBestand()
2.b.1.1: veranderOptie(naam, value)
:OptieVerwerker
2.b.1.1.1: setWaarde(value)
:Optie
Reference: contract voor keurGoed(), vraagOpties() en veranderOptie(optie, value)
keurGoed() zit in save() vervat.
vraagOpties() is het trigger event.
veranderOptie zit vervat in de gelijknamige functie van OptieVerwerker

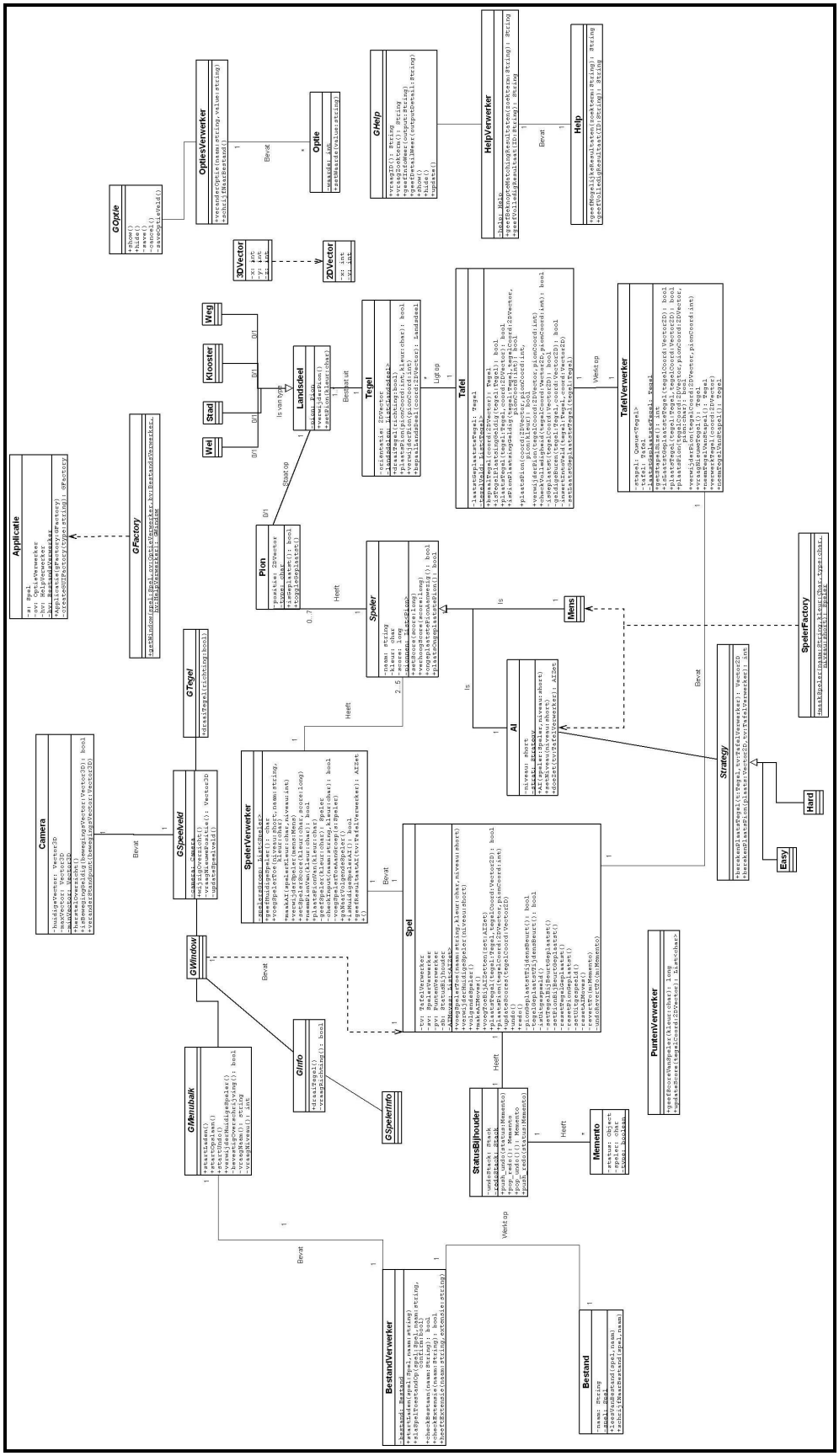

Het wijzigen van het tafeloverzicht
wijzigOverzicht()
:GSpeelVeld
2: geldig = isBewegingGeldig(nieuwePositie)
:Camera
3: [geldig] veranderStandpunt(nieuwePositie)
1: nieuwePositie = vraagNieuwePositie()
4: updateSpeelveld()
Reference: contract voor wijzigOverzicht(richting)


Het raadplegen van de help
triggerHelp()
:GMenu
1: show()
3: hide()
2: update()
2.6: geefDetailVeer(outputDetail)
:GHelp
2.3: geefInfoWeer(output)
2.4: id = vraagId()
2.1: zoekterm = vraagZoekterm()
2.2: output = geefBeknopteMatchingResultaten(zoekterm)
2.5: outputDetail = geefVolledigResultaat(id)
:HelpVerwerker
2.2.1: output = geefMogelijkeResultaten(zoekterm)
:Help
2.5.1: outputDetail = geefVolledigResultaat(id)
References: contract voor vraagHelp()
Dit collaboratiediagram stelt heel het proces voor achter die systeemoperatie.
Andere systeemoperaties die in dit diagram verwerkt zijn:
geefInput(zoekterm) -> vraagZoekterm() en de daaropvolgende functies
sluitHelp() -> hide()


Het ongedaan maken van een vorige actie
1.2: [m exists] revertTo(m)
triggerUndo()
1: undo()
:GMenu
:Spel
undoStack :Stack<Memento>
1.1: m = pop_undo()
1.1.1: m = pop()
m :Memento
:StatusBijhouder
1.1.2: [m exists] push(m)
redoStack :Stack<Memento>

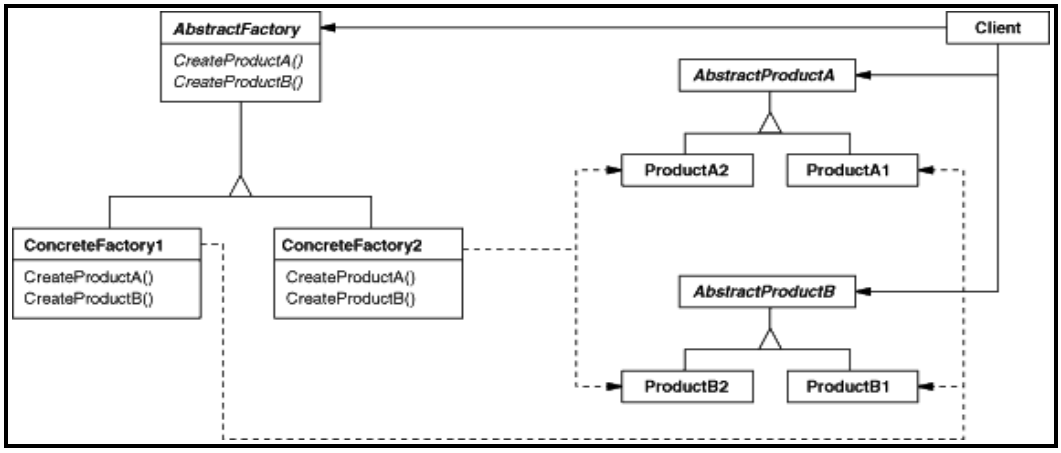

Het ongedaan maken van een ongedaan gemaakte actie
triggerRedo()
:GMenu
1: redo()
:Spel
1.2: [m exists] undoRevertTo(m)
undoStack :Stack<Memento>
1.1: m = pop_redo()
1.1.2: [m exists] push(m)
m :Memento
:StatusBijhouder
1.1.1: m = pop()
redoStack :Stack<Memento>

Design Klassediagram:



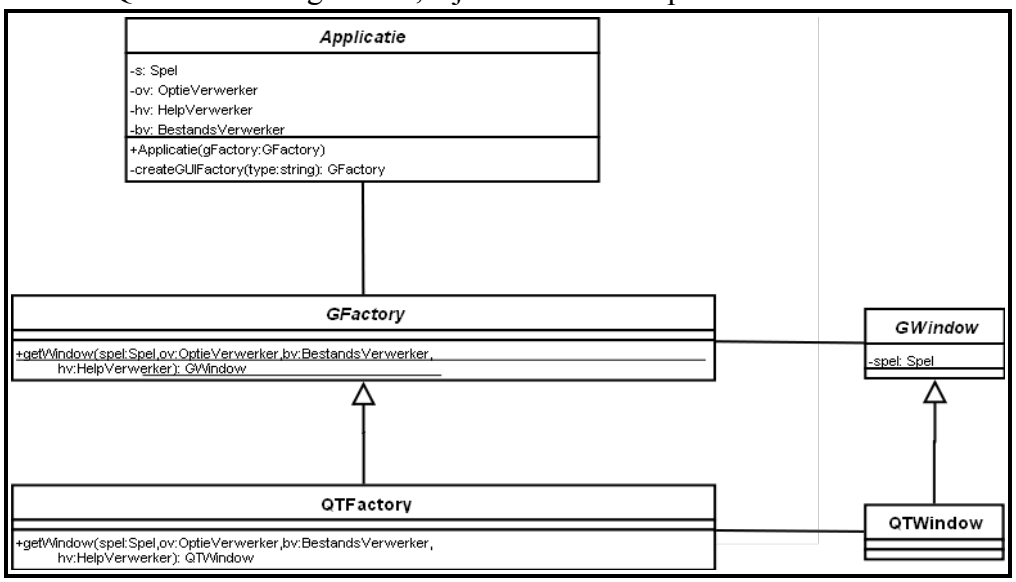
Gebruikte Design Patterns

Abstract Factory (Design Patterns p. 99):

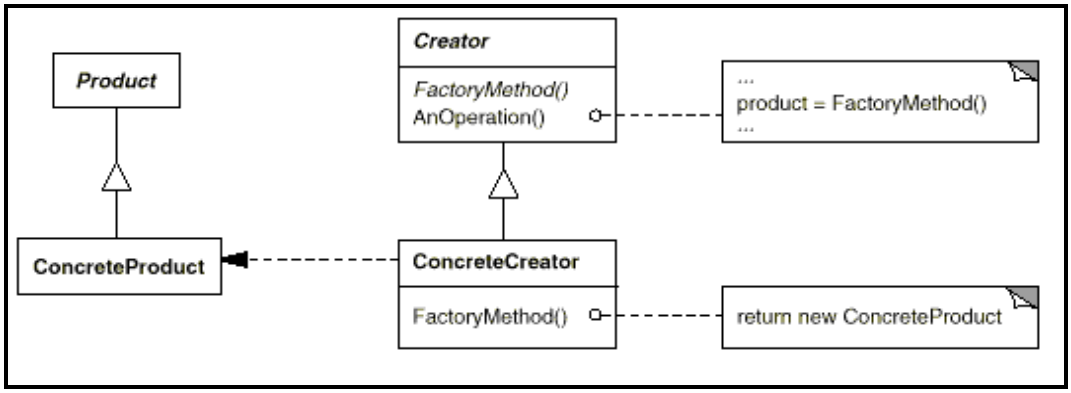


Motivatie:
Dit designpatroon laat ons toe eenvoudig nieuwe API-afhankelijke subklassen te gebruiken. Zo zal *AbstractFactory* een interface zijn, die API-afhankelijke *ConcreteFactories* implementeren. Elk van die concrete factories creëert op hun beurt concrete implementaties van abstracte *Product*-klassen, *Product*-klassen die *AbstractProduct*-interfaces implementeren. De *Client* kan m.b.v. een *ConcreteFactory* (zijnde een referentie naar een *AbstractFactory*) werken met *AbstractProduct*-referenties, gecreëerd doordat de *ConcreteFactory* allerlei concrete *Product*-implementaties bevat.

Betrokken klassen:
Applicatie (cfr. Client), GFactory (cfr. AbstractFactory), QTFactory (cfr. ConcreteFactory), GWindow/ GSpeelveld/... (cfr. AbstractProduct), QTWindow/QTSpeelveld/... (cfr. ConcreteProduct).
Applicatie krijgt in zijn constructor een GFactory-referentie mee. Die wordt bekomen d.m.v. het Factory Method pattern. De concrete implementatie van GFactory, in dit geval QTFactory, houdt een GWindow-referentie bij (zijnde in dit geval een QTWindow). Bij de creatie van een QTWindow wordt op zijn beurt allerlei QT-klassen aangemaakt, zijnde concrete implementaties van G-klassen.

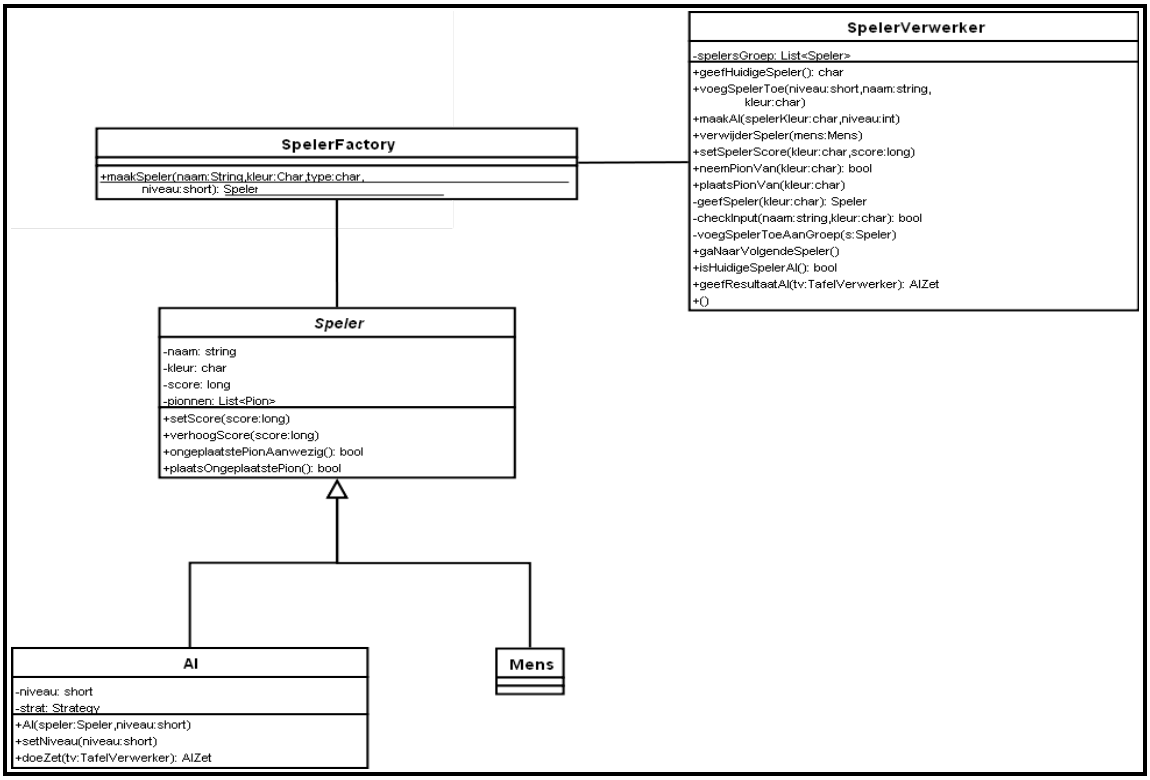


Factory Method (Design Patterns p. 121):

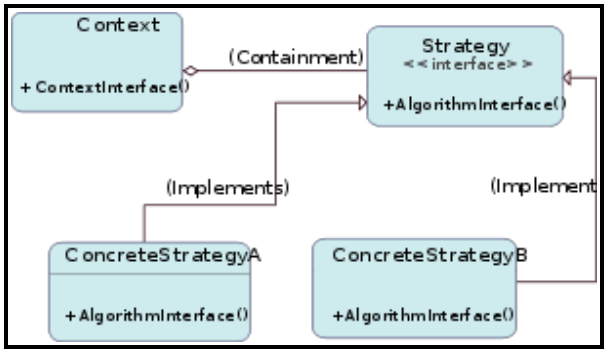


Motivatie:
Dit designpatroon abstraheert constructoren van super- en subklassen tot één methode, die aan de hand van een meegegeven parameter de constructor van de juiste klasse gebruikt om alzo een instantie van de juiste klasse aan te maken. Het resultaat van deze methode is een referentie naar de instantie, bekeken als een superklasse-referentie.

Betrokken klassen:
Speler (cfr. Product), Mens en AI (cfr. ConcreteProduct), SpelerFactory (cfr. ConcreteCreator). FactoryMethod wordt gemapt op maakSpeler(type).
We gebruiken de Speler-interface doorheen de andere klassen, doch moeten we in staat zijn Mens- én AI-instanties (subklassen van Speler) aan te maken. Het gebruik van het Factory Method designpatroon heeft als voordeel dat wanneer we ergens een Speler-instantie moeten aanmaken, we geen gebruik moeten maken van een if-else-structuur, maar van één factorymethode die dat allemaal afhandelt met behulp van een meegegeven parameter.

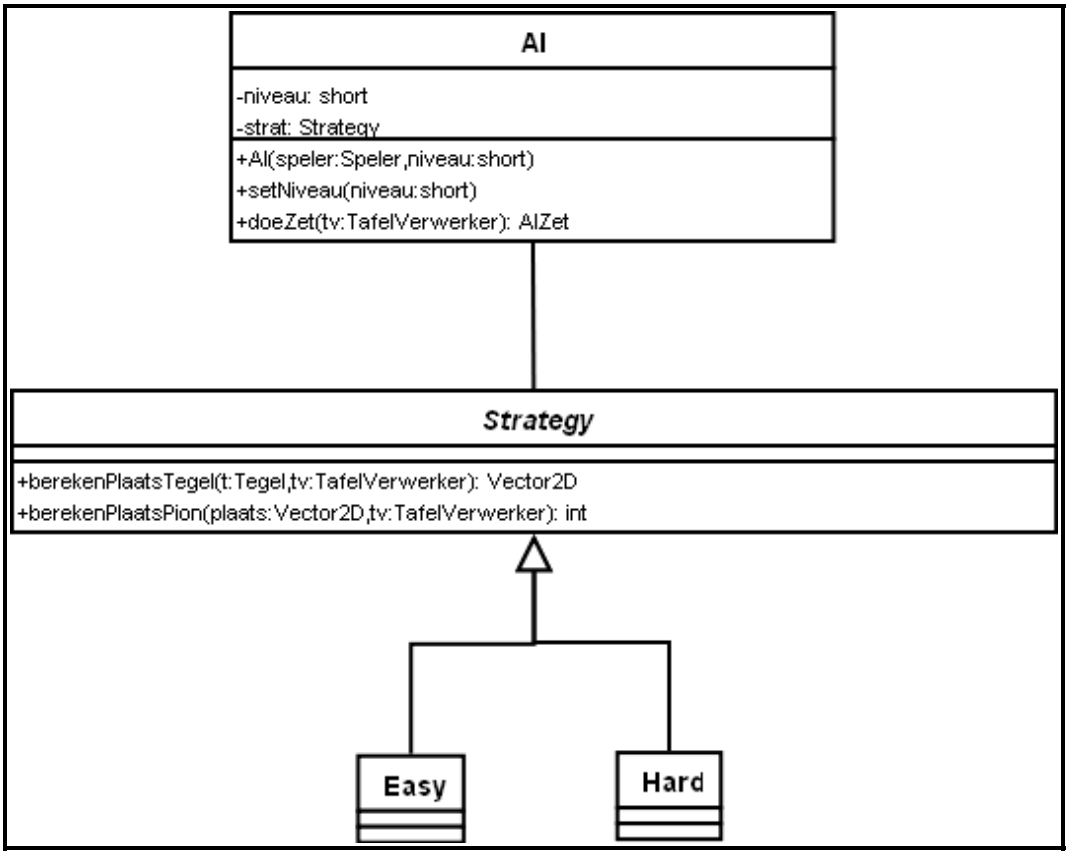


Strategy (Design Patterns p. 349):

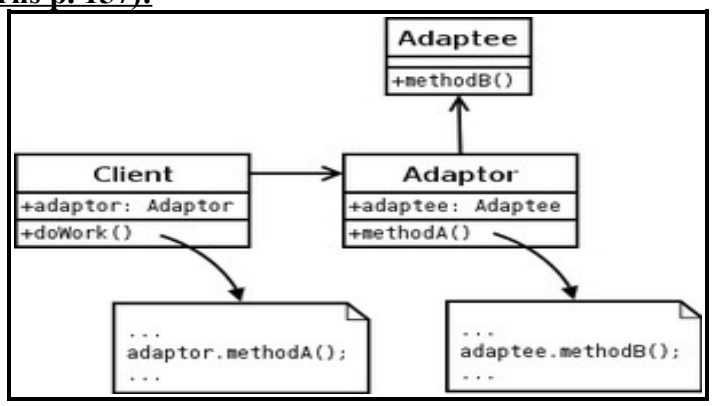


Motivatie:
Context houdt een referentie bij naar een *Strategy*-instantie. Afhankelijk van naar welke concrete instantie deze refereert, verandert het achterliggende algoritme (*AlgorithmInterface*) van een bepaalde methode uit de *Strategy*-interface.

Betrokken klassen:
AI (cfr. Context), Strategy (cfr. Strategy), Easy (cfr. ConcreteStrategy) en Hard (cfr. ConcreteStrategy). Het spreekt voor zich dat dit designpatroon bijzonder handig is voor de implementatie van verschillende AI-niveaus. In AI maken we afhankelijk van het meegegeven AI-niveau, een referentie naar een Strategy-instantie, zijnde een referentie naar Easy of Hard. Impulsief zouden we dit ook bekomen kunnen hebben d.m.v. if-else-structuren in AI in combinatie met een aantal extra hulpfuncties.

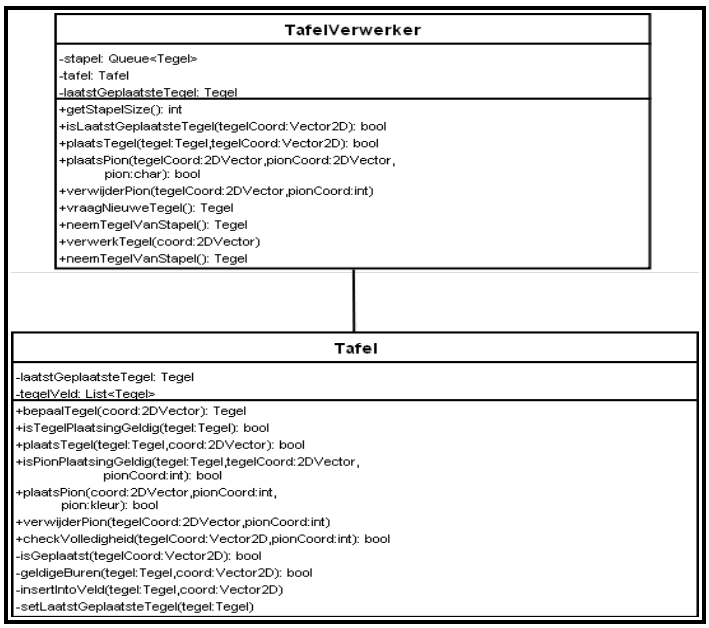


Adapter (Design Patterns p. 157):

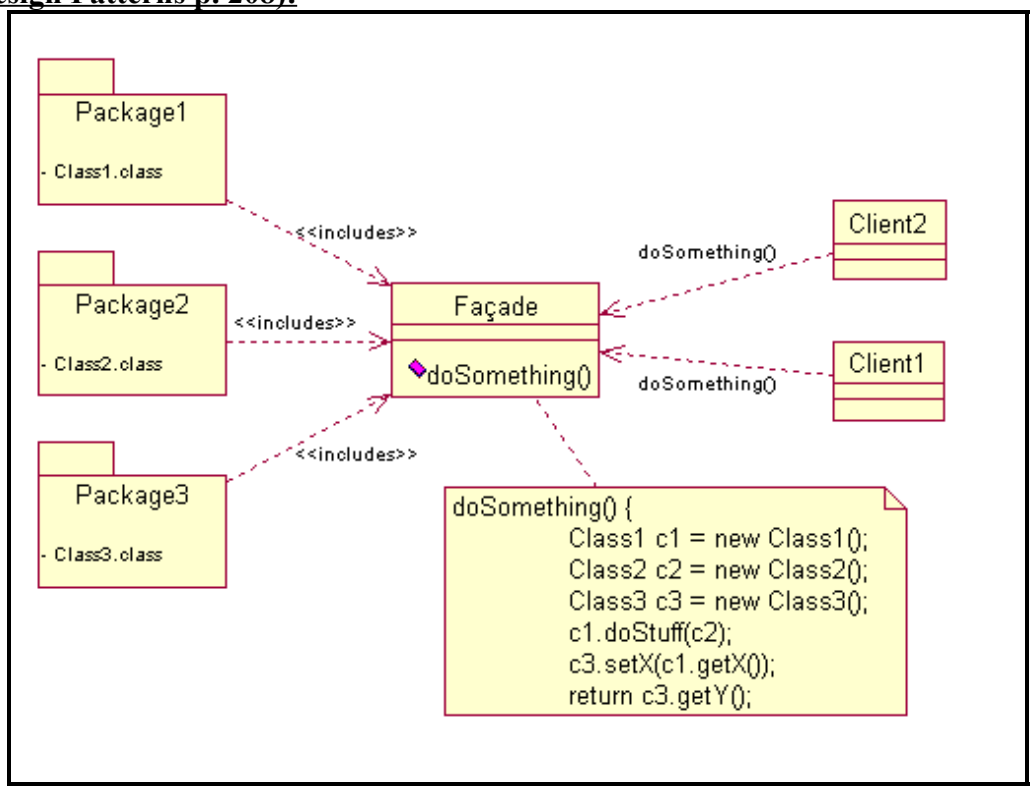


Motivatie:
Dit designpatroon laat ons toe een high-level interface (*Adaptor*) te creëren, een interface die door vele andere klassen (*Client*) kan gebruikt worden, zonder dat die daarvoor de interne structuur/opslag van de *Adaptee*-klassen moeten kennen. Met kan de *Adaptor*-klasse dus beschouwen als een soort van *translation-interface*, die a.h.v. door *Client*s gegeven data de functies van de *Adaptee*-klassen in de juiste volgorde en met de correcte parameters gaat aanroepen.

Betrokken klassen:
Tafel (cfr. Adaptee), TafelVerwerker (cfr. Adaptor), Spel (cfr. Client).
Speler (cfr. Adaptee), SpelerVerwerker (cfr. Adaptor), Spel (cfr. Client).
...
Tafel werkt bijvoorbeeld met een List van Tegels. Hoe die precies gemapt wordt op de visuele voorstelling van het tegelveld moeten andere klassen/clients helemaal niet weten. Daarom is de klasse TafelVerwerker gecreëerd: die wordt als het ware *bovenop* Tafel geplaatst, zodat Clients via TafelVerwerker alle acties op Tegels via tegenover de starttegel relatieve coördinaten kunnen aanspreken.

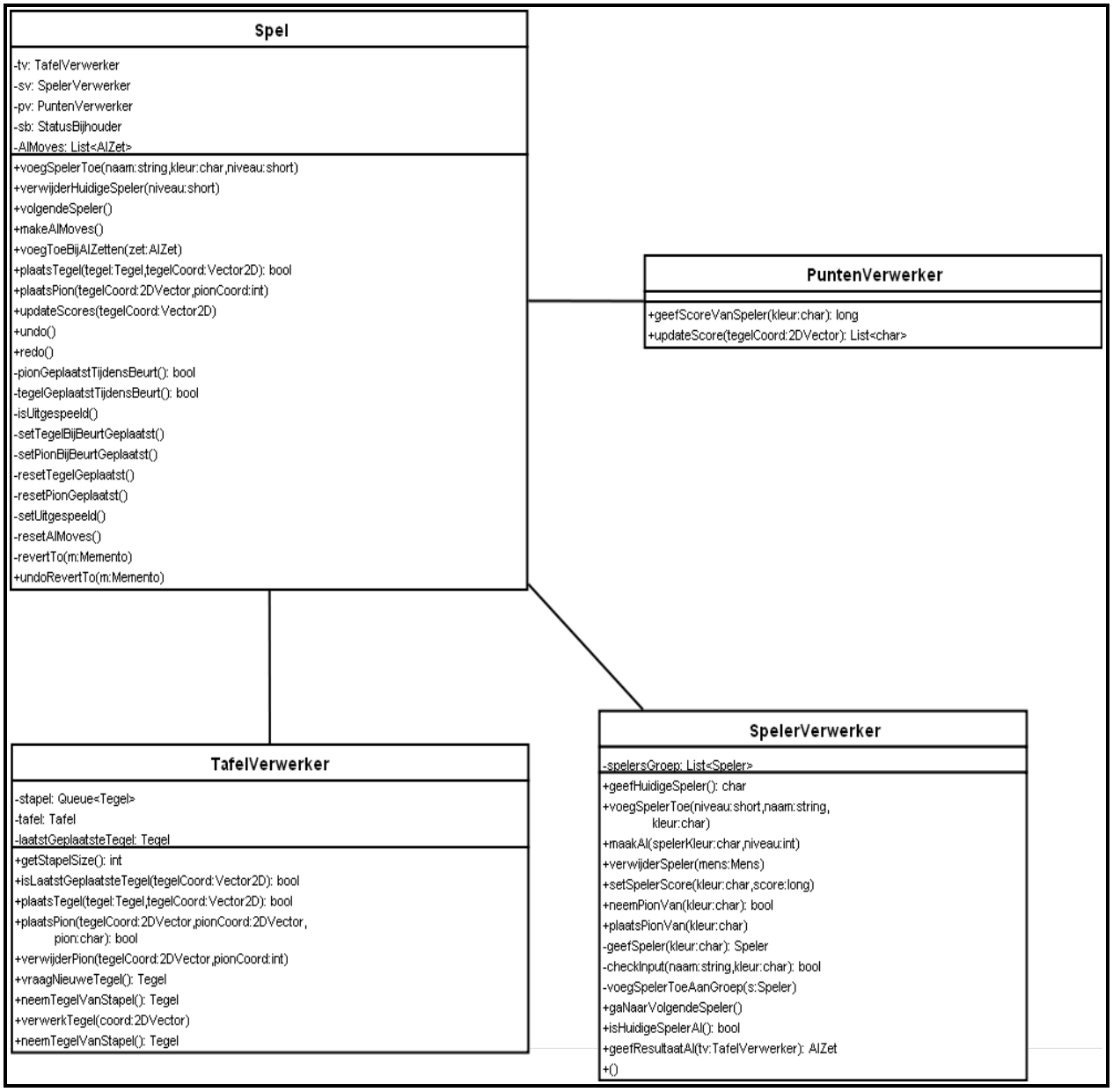


Facade (Design Patterns p. 208):



Motivatie:
Facade bevat een aantal high-levelfuncties, functies die 'buitenstaanders' van de *Facade* kunnen gebruiken. De implementatie van deze functies maakt gebruik van functies uit andere klassen. *Facade* brengt dus samenhorende functionaliteit uit diverse klassen samen in één klasse, één klasse waar 'buitenstaanders' aan genoeg hebben qua functionaliteit.

Betrokken klassen:
Spel (cfr. Facade), SpelerVerwerker (cfr. Class1), PuntenVerwerker (cfr. Class2), TafelVerwerker (cfr. Class), GSpeelVeld (cfr. Client1)
Spel brengt functionaliteit uit SpelerVerwerker, PuntenVerwerker en TafelVerwerker samen in één high-level interface, waarvan GSpeelVeld dan kan gebruik maken. Het automatisch afwerken van de spelbeurt van AI-speler (makeAIMoves()) heeft bvb. nood aan én SpelerVerwerker én TafelVerwerker én PuntenVerwerker.

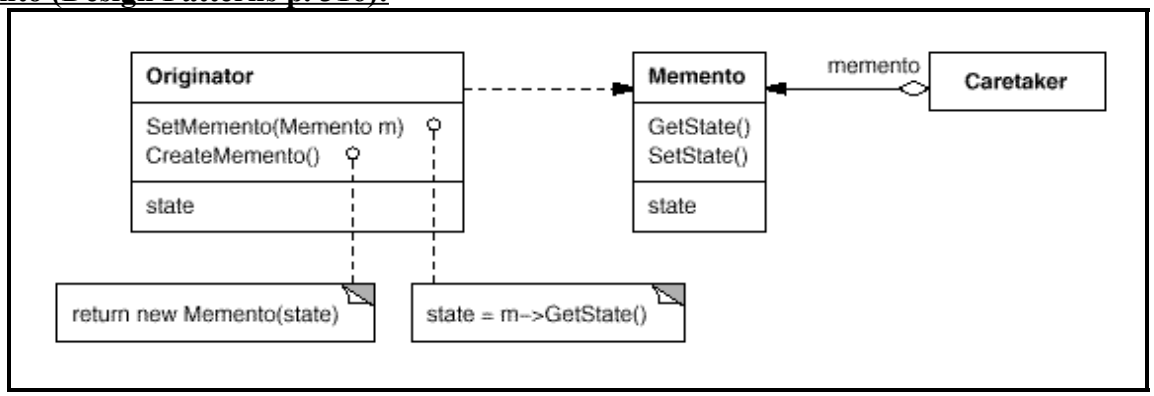


(Facade) Controller (GRASP p. 237):
(zie ook Facade hierboven)

Motivatie:
Facade Controller-klassen zijn klassen die een bepaald subsysteem voorstellen. Deze controllerklasse bevat vooral functies die vanuit de G-klassen moeten worden aangeroepen. Men kan controllerklassen dus bekijken als een communicerende laag tussen de GUI-klassen en de Core-klassen. Gebeurt er bijvoorbeeld een bepaald event in de GUI, dan kan daarop gereageerd worden door een passende functie(s) uit een Facade Controllerklasse aan te roepen. Deze functies zijn dus, net zoals die uit de facade klasse vermeld hierboven, een soort van high-level functies naar de 'buitenwereld' toe.

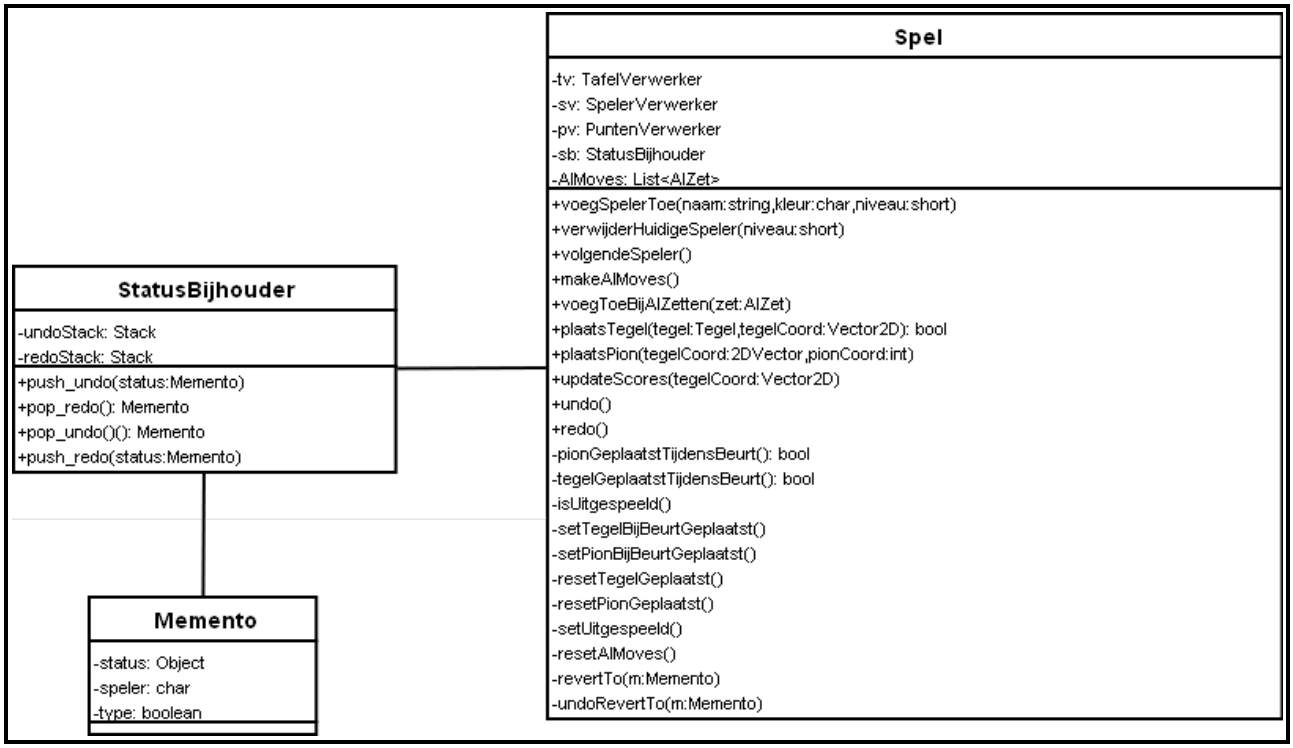
Betrokken klassen:
Spel (de Facede Controllerklasse), SpelerVerwerker (de geabstraheerde Core-klasse), GSpelerInfo (de GUI-klasse). (en gelijkaardige klassen)
GSpelerInfo kan bijvoorbeeld een event opvangen (pion nemen), en daaropvolgend functies uit SpelerVerwerker aanroepen door dat via Spel te doen.

Memento (Design Patterns p. 316):

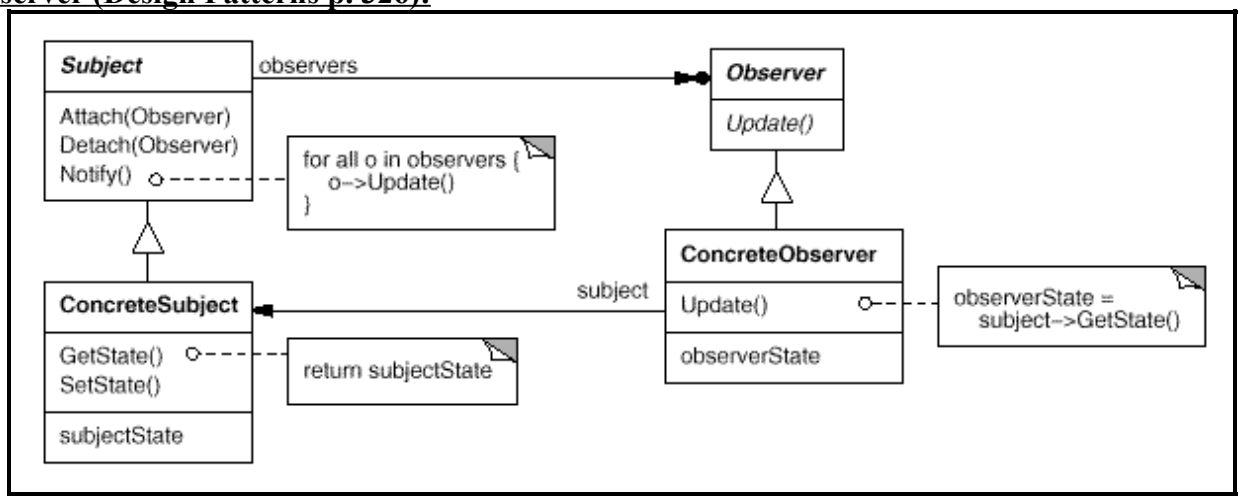


Motivatie:
Dit patroon geeft een eenvoudige structuur om toestanden/states van objecten bij te houden en terug te gebruiken indien gevraagd. *Memento* is een datastructuur die de toestand opslaat. *Originator* is het object waarvan we de toestand willen opslaan of willen terugzetten. *Caretaker* is de containerstructuur die alle *Memento*'s bijhoudt.

Betrokken klassen:
Memento (cfr. Memento), Spel (cfr. Originator), StatusBijhouder (cfr. Caretaker).
Memento is een datastructuur die de toestand van een Spel-instantie uniek kan bepalen. Spel bevat methodes die een status kan terugzetten m.b.v. een gegeven Memento, deze bevat eveneens methodes die zo'n Memento a.h.v. de huidige toestand kan genereren. StatusBijhouder is niet meer dan een container voor Memento-instanties.



Observer (Design Patterns p. 326):



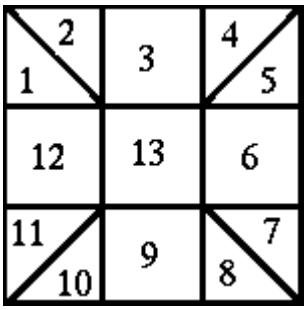
Motivatie:
Dit patroon laat toe een klasse te laten reageren nadat een andere klasse simpelweg kenbaar heeft gemaakt veranderd te zijn. Hiertoe dient *ConcreteSubject* de interface *Subject* te implementeren die functionaliteit aanbiedt zodanig dat andere klassen (bv. *ConcreteObserver*) zich bij *ConcreteSubject* kunnen *aanmelden,* opdat zij op de hoogte zouden kunnen gebracht worden van veranderingen van *ConcreteSubject.* De bij de *ConcreteSubject* aangemelde klassen dienen wel de interface *Observer* te implementeren opdat *ConcreteSubject* voor elk van die klassen een *update*()-functie zou kunnen uitvoeren.

Betrokken klassen:
Observable (= interface cfr. Subject), Observer (cfr. Observer), GSpeelveld (cfr. ConcreteObserver), Spel (cfr. ConcreteSubject). Ook andere GUI-klassen hebben zich als Observer bij Spel geregistreerd. Het is nu heel eenvoudig veranderingen in de Spel-gegevens (scoreverandering, tegelplaatsing, ...) in de GUI te laten zien. Spel dient enkel maar een gepast bericht naar zijn Observers te sturen, en de GUI-klassen kunnen daarna a.h.v. Spel-functies (functies die gegevens opvragen) de GUI updaten.

Nuttige implementatiedetails:

Nu worden enkele gebruikte technieken van de Core-klassen wat nader bestudeerd. De belangrijkste betrokken klasse is Tegel. Hoe wordt daarin nu precies een Tegel uniek gedefiniëerd ?

Dit wordt bekomen door een Tegel op te slaan als een dertiendelentellend veld. Dit raster ziet er intern als volgt uit (Figuur 1):



Figuur 1:
Tegelpresentatie

Met behulp van deze presentatie kan elke Tegel gemakkelijk uniek bepaald worden.

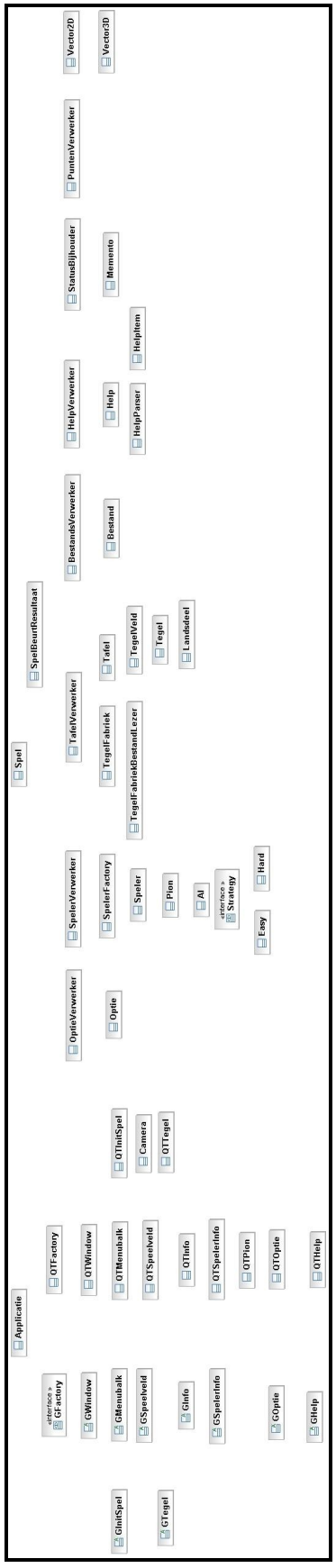
Hoe kunnen we nu een Pion plaatsen met deze voorstelling ? Om dit uit te leggen moet er wel eerst nog een belangrijk punt duidelijk gemaakt worden: namelijk hoe de Landsdelen op de Tegel worden gemapt, een voorbeeld aan de hand van een Tegel zoals (Figuur 2):

Figuur 2:
TegelIcoon

Deze Tegel bezit duidelijk slechts drie verschillende lansdelen: de stad links, de stad rechts en de weide die van boven naar onder doorloopt. Zo wordt dit ook gemapt op de Tegel: er worden dus drie Landsdeel-instanties aangemaakt. Twee Landsdeel-instanties van het type Stad en een Landsdeel-instantie van het type Wei. Deze instanties worden dan op de juiste positie op het veld geplaatst: elke rastercel dat het Landsdeel *onder zich heeft,* bevat een referentie naar die Landsdeel-instantie.

Wanneer er nu een pion op een veldpositie wordt geplaatst, dan wordt er eerst gekeken of op het overeenkomstige Landsdeel niet al een pion staat. Indien dat niet het geval is, dan wordt de pion geplaatst. Door de plaatsing van deze pion zal elke andere veldpositie die een referentie heeft naar dat Landsdeel, ook correct bijgewerkt worden (juist door die referentie). Als we dus bijvoorbeeld op positie twee een pion plaatsen, dan zal positie negen ook automatisch een pion bevatten (vermits dit dezelfde Landsdeel-instantie is).

Klassediagram na refactoring



Logboeken

Sibren Polders

03/10/2008 (8u gewerkt)
- Opstellen van de probleemstelling.
- Indificatie van de systeemfuncties en de systeemattributen.
- Eerste Use Case afgewerkt.

14/10/2008 (7u30 gewerkt)
- Alle Use Cases afgewerkt.

17/10/2008 (10u gewerkt)
- Alle SSDs gemaakt. Contracten opgesteld.
- Alles nog eens goed nagekeken voor de deadline.

17/10/2008
- jpeg en document gemaakt om op te sturen.

24/10/2008 (6u gewerkt)
- collaboratie diagrammen voor volgendeSpeler en creatieVanEenSpeler gemaakt.

30/10/2008 (10u gewerkt)
- Alle collaboratiediagrammen afgewerkt.
- Domein model gestart. Design patterns besproken/voorgesteld.

07/11/2008 (7u45 gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.

09/11/2008 (9u gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.
- Begin klassendiagram met design patterns.
- Verdeling van individueel te maken contracten en SSDs

11/11/2008 (1 uur gewerkt)
- individuele contracten gemaakt

13/11/2008 (1 uur gewerkt)
- individuele SSDs gemaakt (bevestig, veranderNiveau)

14/11/2008 (9u gewerkt)
- Samenvoegen van de individueel gemaakte contracten.
- Verder uitwerken van het klassendiagram.
- Design pattern ingevoegd. (Strategy)
- Collaboratiediagram volgendeSpeler en TafelOverzichtWijzigen aangemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

14/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Memento + Adaptor).
- Collaboratiediagram undo gemaakt.
- Collaboratiediagram redo aangepast.
- Collaboratiediagram plaatsPion, neemPionTerug en plaatsTegel gemaakt
- Andere collaboratiediagrammen waar nodig aangepast.

15/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Factory Method + AbstractFactory).
- Collaboratiediagram startApplicatie gemaakt.
- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

19/11/2008 (6u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Facade Controller).
- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

20/11/2008 (10u gewerkt)

- Verder uitwerken van het klassendiagram.
- Collaboratiediagram Laden en Opslaan gemaakt.
- Collaboratiediagram SpelerVerwijderen gemaakt.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

21/11/2008 (8u gewerkt)
- Verder uitwerken van het klassendiagram.
- Collaboratiediagram VeranderOpties en Help gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

26/11/2008 (3u gewerkt)
- Begonnen aan de Unit Tests in Junit.
- Klassendiagram bijgewerkt na enkele opmerkingen.

27/11/2008 (4u gewerkt)
- Unit Tests afwerken (AI, Easy, Hard, Memento, statusbijhouder)

28/11/2008 (4u gewerkt)
- Klassendiagram bijwerken.
- Collaboratiediagrammen waar nodig aangepast.
- Design Patterns in een document uitgelegd.

3/12/2008 (5u gewerkt)
- Begonnen aan implementatie.

4/12/2008 (6u gewerkt)
- Implementatie (tafel, tafelverwerker, camera)
- Implementatie QTMenubalk, QT/GInfo QT/GSpeelveld, QT/GWindow

5/12/2008 (4u gewerkt)
- Implementatie QT/GSpeelveld

8/12/2008 (7u gewerkt)
- implementatie Applicatie, GFactory, GWindow, Lansdeel, QTFactory, QTInfo, QTSpeelveld, QTTegel, QTWindow, Tafel, Tegel, TegelFabriek, TegelFabriekBestandLezer, Tegels.xm
- unit test voor Tafel, Tegel, TegelFrabriek, TegelFabriekBestandlezer aangepast.

9/12/2008 (4u gewerkt)
- Implementatie (QTSpelerInfo, Pion, Tafel)
- Unit test tafel aangepast

10/12/2008 (4u gewerkt)
- Implementatie (Tafel, Tegel, TegelFabriek, TegelFabriekBestandLezer)
- Unit test tafel, tegelFabriekBestandLezer aangepast.

11/12/2008 (6u gewerkt)
- Implementatie (Tafel, Tegel, Lansdeel, TafelVerwerker) + andere klassen geupdate
- Unit tests geupdate

12/12/2008 (5u gewerkt)
- Implementatie (Tafel, Tegel, TafelVerwerker, TegelFabriek)

13/12/2008 (4u gewerkt)
- Implementatie (Tafel, Camera, Vector2-3D)
- Implementatie van I/O.

15/12/2008 (3.50u gewerkt)
- Implementatie (Landsdeel, Puntenverwerker, Tafel)
- unit test puntenverwerker aangemaakt

15/12/2008 (3.50u gewerkt)
- Implementatie (Tegel, Tafel, TegelFabriek)

18/12/2008 (6u gewerkt)
- Implementatie (klassen gerefactored)
- Implementatie (GspeelVeld, QTInfo, QTTegel, Tafel,)
- Foutjes uit xml gehaald
- Presentatietje ineengestoken

19/12/2008 (3u gewerkt)
- Kleine foutjes uit collaboratiediagrammen gehaald (schrijffouten, pijlplaatsing, ..)
- Debug van TegelPlaatsing fout.
- Refactoring, bepaalde klassens werden te groot → opdeling in subklassens

21/12/2008 (5u gewerkt)

- Refactoring van Tafel klassen (een nieuwe subklasse TegelVeld gemaakt)
- Updaten van andere klassen

27/12/2008 (5u gewerkt)
- Implementatie (QT-klassen gerefactored)
- QTMenubalk meer functionaliteit gegeven.

29/12/2008 (5u gewerkt)
- Implementatie (QTSpeelVeld, QTTegel)
- Update andere klassen

01/01/2009 (3.5u gewerkt)
- Implementatie (Tafel, Tegel. TegelVeld)

06/01/2009 (5u gewerkt)
- Toelichting Tegel geschreven
- Debugging + refactoring

07/01/2009 (3u gewerkt)
- Eindverslag nagekeken + kleine aanpassingen gemaakt

<u>Sam Van Rijn (gewerkte uren zijn vooral facultatief)</u>

03/10/2008 (8u gewerkt)
- Opstellen van de probleemstelling.
- Indificatie van de systeemfuncties en de systeemattributen.
- Eerste Use Case afgewerkt.

14/10/2008 (7u30 gewerkt)
- Alle Use Cases afgewerkt.

17/10/2008 (10u gewerkt)
- Alle SSDs gemaakt. Contracten opgesteld.
- Alles nog eens goed nagekeken voor de deadline.

18/10/2008
- jpeg en document gemaakt om op te sturen.

24/10/2008 (6u gewerkt)
- collaboratie diagrammen voor volgendeSpeler en creatieVanEenSpeler gemaakt.

30/10/2008 (10u gewerkt)
- Alle collaboratiediagrammen afgewerkt.
- Domein model gestart. Design patterns besproken/voorgesteld.

07/11/2008 (7u45 gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.

09/11/2008 (9u gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.
- Begin klassendiagram met design patterns.

14/11/2008 (9u gewerkt)
- Samenvoegen van de individueel gemaakte contracten.
- Verder uitwerken van het klassendiagram.
- Design pattern ingevoegd. (Strategy)
- CollaboratieDiagram volgendeSpeler en TafelOverzichtWijzigen aangemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

14/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Memento + Adaptor).
- Collaboratiediagram undo gemaakt.
- Collaboratiediagram redo aangepast.
- Collaboratiediagram plaatsPion, neemPionTerug en plaatsTegel gemaakt
- Andere collaboratiediagrammen waar nodig aangepast.

15/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Factory Method + AbstractFactory).
- Collaboratiediagram startApplicatie gemaakt.

- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

19/11/2008 (6u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Facade Controller).
- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

20/11/2008 (10u gewerkt)
- Verder uitwerken van het klassendiagram.
- Collaboratiediagram Laden en Opslaan gemaakt.
- Collaboratiediagram SpelerVerwijderen gemaakt.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

21/11/2008 (8u gewerkt)
- Verder uitwerken van het klassendiagram.
- Collaboratiediagram VeranderOpties en Help gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

26/11/2008 (3u gewerkt)
- Begonnen aan de Unit Tests in Junit.
- Klassendiagram bijgewerkt na enkele opmerkingen.

27/11/2008 (4u gewerkt)
- Unit Tests afwerken (Bestand, AI-Speler)

28/11/2008 (4u gewerkt)
- Unit Test afwerken (Bestand, AI-Speler)
- Klassendiagram bijwerken.
- Collaboratiediagrammen waar nodig aangepast.
- Design Patterns in een document uitgelegd.

9/12/2008 (3u gewerkt)
- Implementatie (Gspeelveld, QTSpeelveld, Bestand)

10/12/2008 (4u gewerkt)
- Implementatie (Gspeelveld, QTSpeelveld, Camera)

11/12/2008 (4u gewerkt)
- Implementatie (Gspeelveld, QTSpeelveld, QTWindow)

12/12/2008 (5u gewerkt)
- Implementatie (Gspeelveld, QTSpeelveld, debuggen)

14/12/2008 (4u gewerkt)
- Implementatie (debuggen)

18/12/2008 (6u gewerkt)
- Implementatie (klassen gerefactored)
- Presentatietje ineengestoken

19/12/2008 (3u gewerkt)
- Implementatie (klassen gerefactored)

31/12/2008 (3u gewerkt)
- Implementatie (debuggen, mogelijkheden groen kleuren hermaken)

01/01/2009 (7u gewerkt)
- Implementatie (debuggen, pionplaatsing debuggen)

03/01/2009 (4u gewerkt)
- Eindverslag bijgewerkt.
- Implementatie (QTSpeelveld, debuggen)

04/01/2009 (6u gewerkt)
- Eindverslag bijgewerkt.

06/01/2009 (5u gewerkt)
- Eindverslag bijgewerkt.
- Debuggen + refactoring

07/01/2009 (3u gewerkt)
- Eindverslag afgewerkt.

Sam Van Gucht (gewerkte uren zijn vooral facultatief)

03/10/2008 (8u gewerkt)
- Opstellen van de probleemstelling.
- Indificatie van de systeemfuncties en de systeemattributen.
- Eerste Use Case afgewerkt.

14/10/2008 (7u30 gewerkt)
- Alle Use Cases afgewerkt.

17/10/2008 (10u gewerkt)
- Alle SSDs gemaakt. Contracten opgesteld.
- Alles nog eens goed nagekeken voor de deadline.

18/10/2008
- jpeg en document gemaakt om op te sturen.

24/10/2008 (6u gewerkt)
- collaboratie diagrammen voor volgendeSpeler en creatieVanEenSpeler gemaakt.

30/10/2008 (10u gewerkt)
- Alle collaboratiediagrammen afgewerkt.
- Domein model gestart. Design patterns besproken/voorgesteld.

07/11/2008 (7u45 gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.

09/11/2008 (9u gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.
- Begin klassendiagram met design patterns.

14/11/2008 (9u gewerkt)
- Samenvoegen van de individueel gemaakte contracten.
- Verder uitwerken van het klassendiagram.
- Design pattern ingevoegd. (Strategy)
- Collaboratiediagram volgendeSpeler en TafelOverzichtWijzigen aangemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

14/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Memento + Adaptor).
- Collaboratiediagram undo gemaakt.
- Collaboratiediagram redo aangepast.
- Collaboratiediagram plaatsPion, neemPionTerug en plaatsTegel gemaakt
- Andere collaboratiediagrammen waar nodig aangepast.

15/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Factory Method + AbstractFactory).
- Collaboratiediagram startApplicatie gemaakt.
- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

19/11/2008 (6u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Facade Controller).
- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

20/11/2008 (10u gewerkt)
- Verder uitwerken van het klassendiagram.
- Collaboratiediagram Laden en Opslaan gemaakt.
- Collaboratiediagram SpelerVerwijderen gemaakt.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

21/11/2008 (8u gewerkt)

- Verder uitwerken van het klassendiagram.
- Collaboratiediagram VeranderOpties en Help gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

26/11/2008 (3u gewerkt)
- Begonnen aan de Unit Tests in Junit.
- Klassendiagram bijgewerkt na enkele opmerkingen.

27/11/2008 (4u gewerkt)
- Unit Tests afwerken (Tegel,Tafel,Pion)

28/11/2008 (4u gewerkt)
- Unit Test afwerken (Tegel,Tafel,Pion)
- Klassendiagram bijwerken.
- Collaboratiediagrammen waar nodig aangepast.

5/12/2008 (5u gewerkt)
- Begonnen aan implementatie.

8/12/2008 (4u gewerkt)
- Implementatie (Ghelp,QTHelp)

9/12/2008 (4u gewerkt)
- Implementatie (Ghelp,QTHelp,QTInfo,QTSpelerInfo)

10/12/2008 (4u gewerkt)
- Implementatie (Tegel,TegelVerwerker)

11/12/2008 (4u gewerkt)
- Implementatie (QTHelp,QTTegel)

12/12/2008 (5u gewerkt)
- Implementatie (QTTegel, Tegel)

14/12/2008 (4u gewerkt)
- Implementatie (QTInfo,QTSpelerInfo)

18/12/2008 (6u gewerkt)
- Implementatie (Debugging)
- Presentatietje ineengestoken

20/12/2008 (3u gewerkt)
- Implementatie (QTInfo)

01/01/2009 (3u gewerkt)
- Implementatie (QTInfo,QTInitSpeler)

04/01/2009 (2u gewerkt)
- Implementatie(Debugging)

05/01/2009 (5u gewerkt)
- Implementatie (debugging + refactoring)

06/01/2009 (5u gewerkt)
- Eindverslag wat bijgewerkt.
- Debugging + refactoring

07/01/2009 (3u gewerkt)
- Eindverslag afgewerkt.

<u>Sibrand Staessens (gewerkte uren zijn vooral facultatief)</u>

03/10/2008 (8u gewerkt)
- Opstellen van de probleemstelling.
- Indificatie van de systeemfuncties en de systeemattributen.
- Eerste Use Case afgewerkt.

14/10/2008 (7u30 gewerkt)
- Alle Use Cases afgewerkt.

17/10/2008 (10u gewerkt)
- Alle SSDs gemaakt. Contracten opgesteld.
- Alles nog eens goed nagekeken voor de deadline.

18/10/2008
- jpeg en document gemaakt om op te sturen.

24/10/2008 (6u gewerkt)

- collaboratie diagrammen voor volgendeSpeler en creatieVanEenSpeler gemaakt.

30/10/2008 (10u gewerkt)
- Alle collaboratiediagrammen afgewerkt.
- Domein model gestart. Design patterns besproken/voorgesteld.

07/11/2008 (7u45 gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.

09/11/2008 (9u gewerkt)
- Aanpassing gemaakt aan de Use Cases, SSDs en contracten a.d.h.v. de feedback.
- Begin klassendiagram met design patterns.

11/11/2008 (1u30 gewerkt)
- Individueel cotracten gemaakt.

13/11/2008 (2u gewerkt)
- Individueel SSD gemaakt

14/11/2008 (9u gewerkt)
- Samenvoegen van de individueel gemaakte contracten.
- Verder uitwerken van het klassendiagram.
- Design pattern ingevoegd. (Strategy)
- CollaboratiediagramvolgendeSpeler en TafelOverzichtWijzigen aangemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

14/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Memento + Adaptor).
- Collaboratiediagram undo gemaakt.
- Collaboratiediagram redo aangepast.
- Collaboratiediagram plaatsPion, neemPionTerug en plaatsTegel gemaakt
- Andere collaboratiediagrammen waar nodig aangepast.

15/11/2008 (4u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Factory Method + AbstractFactory).
- Collaboratiediagram startApplicatie gemaakt.
- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

19/11/2008 (6u gewerkt)
- Verder uitwerken van het klassendiagram.
- Design patterns ingevoegd (Facade Controller).
- Collaboratiediagram creatieVanEenSpeler aangepast.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

20/11/2008 (10u gewerkt)
- Verder uitwerken van het klassendiagram.
- Collaboratiediagram Laden en Opslaan gemaakt.
- Collaboratiediagram SpelerVerwijderen gemaakt.
- Collaboratiediagram startApplicatie gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

21/11/2008 (8u gewerkt)
- Verder uitwerken van het klassendiagram.
- Collaboratiediagram VeranderOpties en Help gemaakt.
- Andere collaboratiediagrammen waar nodig aangepast.

26/11/2008 (3u gewerkt)
- Begonnen aan de Unit Tests in Junit.
- Klassendiagram bijgewerkt na enkele opmerkingen.

27/11/2008 (4u gewerkt)
- Unit Tests afwerken (Help, HelpVerwerker, Optie, OptieVerwerker, AI, Easy, Hard)

28/11/2008 (4u gewerkt)
- Unit Test afwerken (Help, HelpVerwerker, Optie, OptieVerwerker, AI, Easy, Hard)
- Klassendiagram bijwerken.
- Collaboratiediagrammen waar nodig aangepast.

- Design Patterns in een document uitgelegd.

5/12/2008 (5u gewerkt)
- Begonnen aan implementatie.

8/12/2008 (4u gewerkt)
- Implementatie (Speler, SpelerVerwerker, AI, Easy, Hard, Strategy)

9/12/2008 (4u gewerkt)
- Implementatie (Applicatie, SpelerFactory + updates aan de andere klassen)

10/12/2008 (4u gewerkt)
- Implementatie (Optie + OptieVerwerker + Goptie + QTOptie)

11/12/2008 (4u gewerkt)
- Implementatie (Help + HelpVerwerker + HelpItem + HelpParser)

12/12/2008 (5u gewerkt)
- Implementatie (Ghelp + QTHelp, andere klassen geupdatet)

14/12/2008 (4u gewerkt)
- Implementatie (QTKlassen van anderen bestudeerd)

18/12/2008 (6u gewerkt)
- Implementatie (klassen gerefactored)
- Presentatietje ineengestoken

19/12/2008 (3u gewerkt)
- Implementatie (klassen gerefactored)

22/12/2008 (5u gewerkt)
- Implementatie (Tegelplaatsing + Pionplaatsing debuggen)

23/12/2008 (4u gewerkt)
- Implementatie (Tegelplaatsing + Pionplaatsing debuggen)

28/12/2008 (5u gewerkt)
- Implementatie (QT-klassen gerefactored (vooral QTSpeelVeld en QTInfo en QTTegel)

03/01/2009 (2u gewerkt)
- Eindverslag wat bijgewerkt.

05/01/2009 (5u gewerkt)
- Implementatie (debugging + refactoring)

06/01/2009 (5u gewerkt)
- Eindverslag wat bijgewerkt.
- Debugging + refactoring

07/01/2009 (3u gewerkt)
- Eindverslag afgewerkt.
- Executable gemaakt.