Software Requirements Specification

Created by Auto Mation, last modified by Roy Gerrits on Jun 11, 2021

Onderwerp	Software Requirement Specification
Status	DONE
Versie	2.0
Datum	10 Jun 2021
Auteurs	ASD 2020-2021-s2 Klas ITN-ASD-A-f
Docenten	@Eveline Bouwman @Michel Koolwaaij @Rody Middelkoop
Projectgroep	ASD 2020-2021-s2 Projectgroep 1
Eindverantwoordelijke	Planners

Inhoud

1. Introduction

1.1. Overall Description

Het doel van het project is om een nieuwe variant van een dungeon text-adventure game te maken. Meer informatie over deze opdracht is terug te vinden in hoofdstuk 4 van het Plan van Aanpak. De beschrijving van het spel zelf is terug te vinden in het Game Design Document. In dit document worden de requirements opgesteld en deze worden uitgewerkt in de vorm van usecases. Ook worden het domeinmodel, system sequence diagrammen en user interfaces schetsen uitgewerkt

1.2. User Classes and Characteristics

In de context van het document worden de mensen die gebruik gaan maken van de software uitgedrukt in actoren. Per actor is een beschrijving te vinden in tabel 1.

Tabel 1: Beschrijving actoren

Actor	Beschrijving
Speler	Een persoon die het spel speelt.
Host	De speler die het spel start.

1.3. Operating Environment

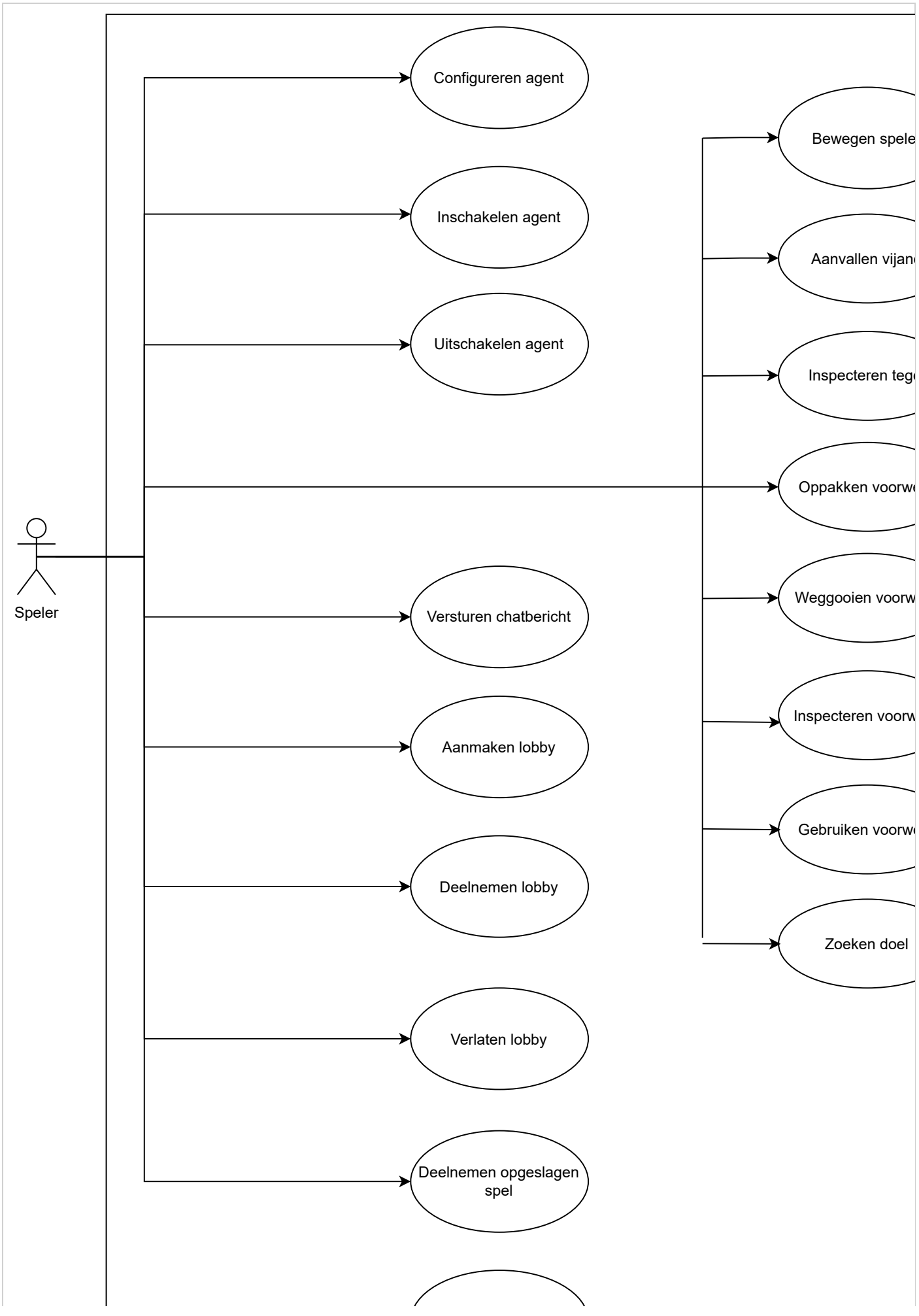
Spelers kunnen het spel downloaden via een centrale server. Tijdens het spel zal een speler aangewezen worden als host die communiceert met andere spelers via een softwarematige netwerkswitch. Een diagram van deze omgeving is terug te vinden in het hoofdstuk 3 van het SAD. Daarnaast is er in het deployment diagram meer informatie te vinden over de omgeving waarin het spel wordt uitgevoerd.

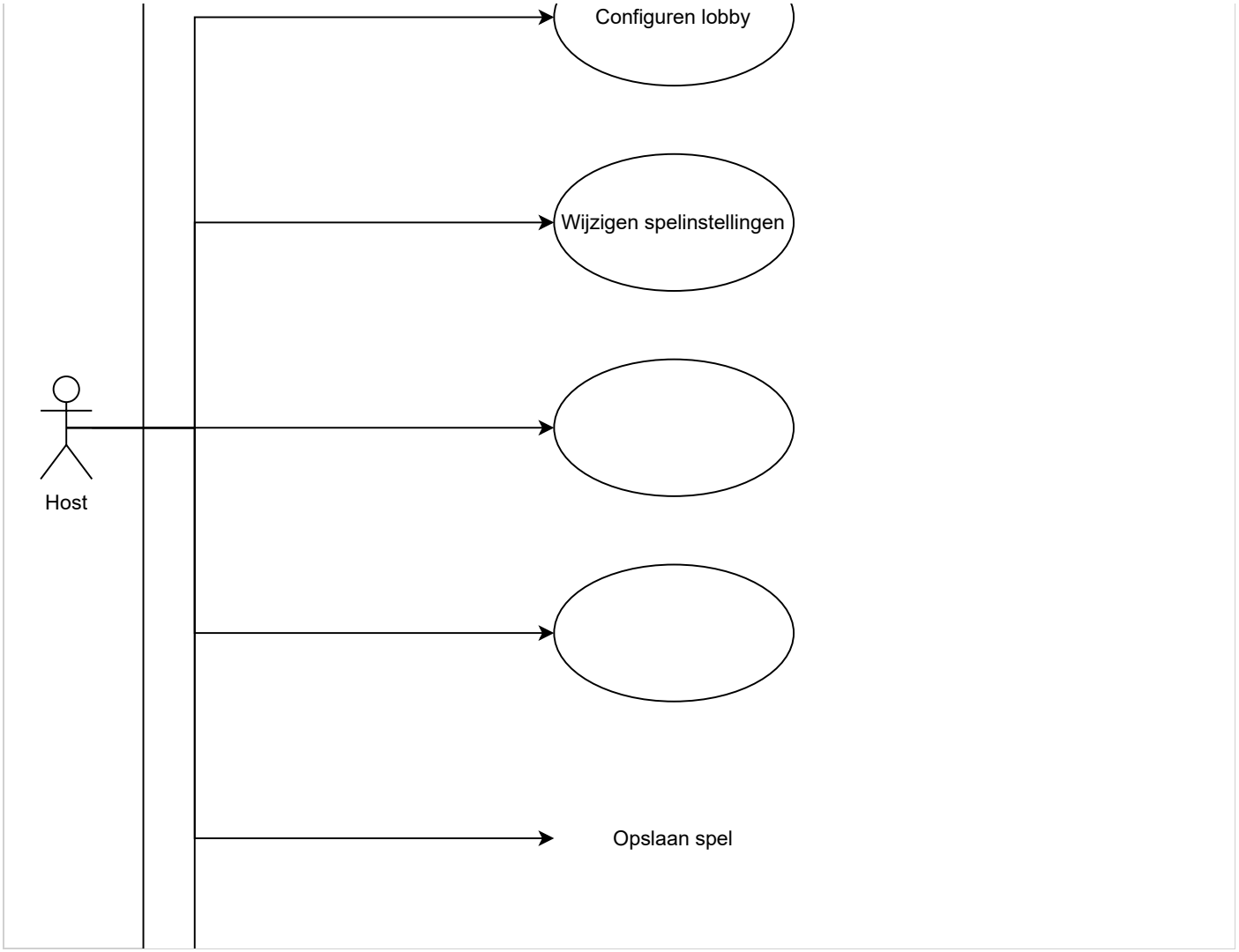
1.4. Design and Implementation Constraints

De technische constraints zijn terug te vinden in hoofdstuk 5.3 van het SAD.

1.5. Product Functions

Hieronder staat een usecasediagram weergegeven. Alle acties die een speler uitvoert, moet een host ook kunnen. Wij hebben dit alleen apart gemodelleerd omdat een host een type speler is.


Configureren agent
Inschakelen agent
Uitschakelen agent
Versturen chatbericht
Aanmaken lobby
Deelnemen lobby
Verlaten lobby
Deelnemen opgeslagen spel
Speler
Bewegen spele
Aanvallen vijan
Inspecteren teg
Oppakken voorw
Weggooien voorw
Inspecteren voorw
Gebruiken voorw
Zoeken doel



Figuur 1: Usecasediagram

1.5.1. Usecase overview

Tabel 2: High level usecase overview

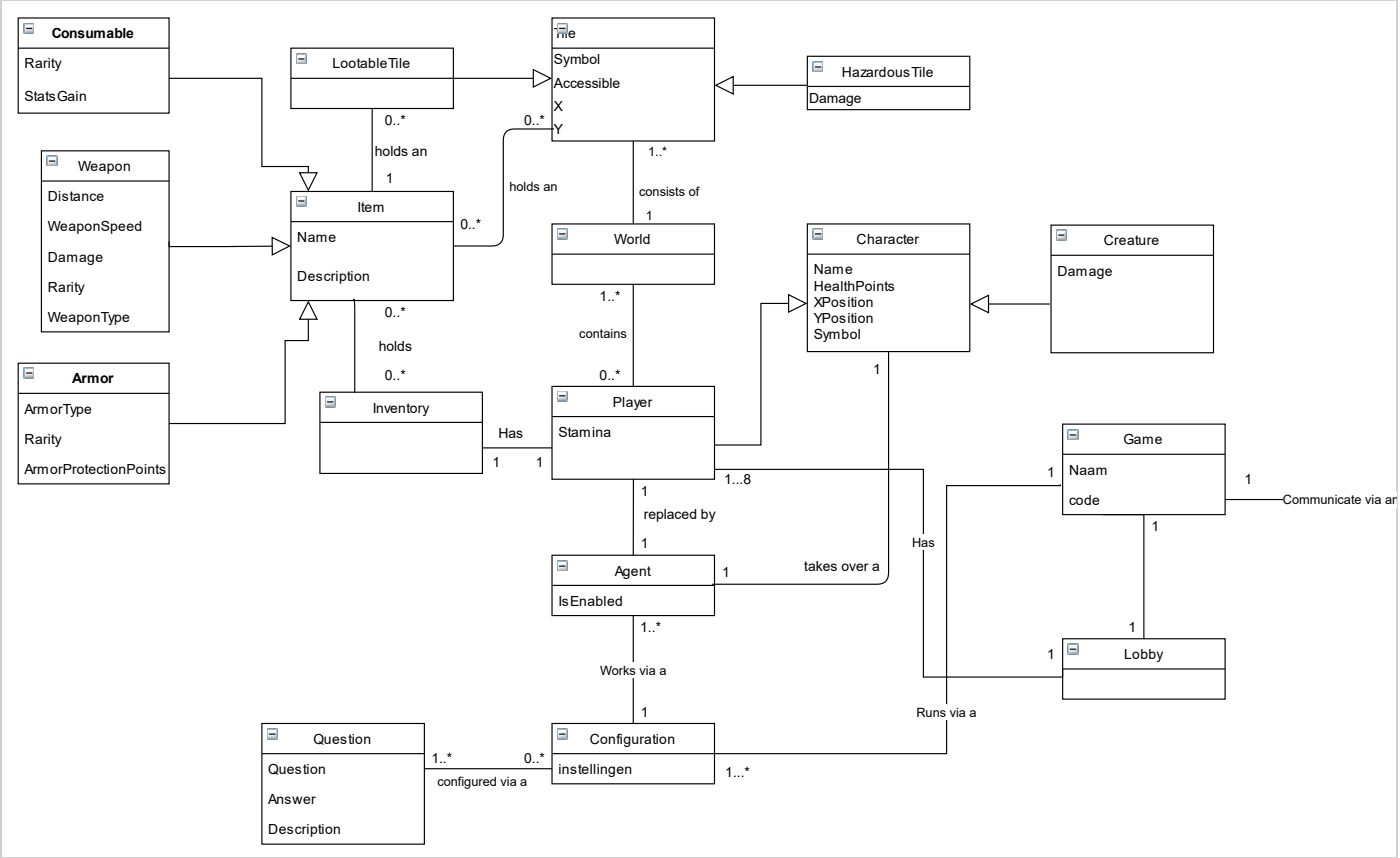
Usecase	Beschrijving
Configureren agent	De speler configureert het gedrag van de agent, zoals het gedrag bij vechten en verkennen door middel van vragen.
Inschakelen agent	De speler schakelt tijdens het spelen de agent in om de besturing van het karakter over te nemen.
Uitschakelen agent	De speler schakelt tijdens het spelen de agent uit en neemt de besturing van het karakter weer over.

Usecase	Beschrijving
Uitvoeren actie	De speler voert tijdens het spel een actie uit zodat de speler het spel kan spelen.
Bewegen speler	De speler beweegt tijdens het spel op een vakje naar rechts, links, boven of onder.
Aanvallen vijand	Een speler kan een vijand aanvallen.
Inspecteren tegel	Een speler wilt kijken welke voorwerpen op de huidige tegel liggen.
Oppakken voorwerp	Een speler wilt een voorwerp oppakken.
Weggooien voorwerp	Een speler wilt een voorwerp weg doen.
Inspecteren voorwerp	Een speler wilt informatie over een voorwerp in zijn inventaris.
Gebruiken voorwerp	Een speler wilt een voorwerp gebruiken.
Zoeken doel	Een speler wilt een doel zoeken.
Versturen chatbericht	Een speler of host kan een chatbericht versturen tijdens een spelsessie om op deze manier te communiceren met andere spelers.
Aanmaken lobby	Een speler kan een nieuwe lobby maken vanuit het spel en krijgt de optie om deze te configureren. Zodra de lobby succesvol is aangemaakt wordt deze speler de host van de lobby.
Deelnemen lobby	Een speler kan aan een bestaande spel lobby deelnemen.
Verlaten lobby	Een speler wilt een spel verlaten.
Wijzigen spelinstellingen	Als host wil ik tijdens het spelen instellingen zoals een monster moeilijkheidsgraad kunnen aanpassen.
Starten spel	Een host wilt het spel starten.
Beëindigen spel	Als host van een lobby kun je het gehele spel stoppen, zodat het spel opgeslagen wordt en iedereen de lobby verlaat.
Opslaan spel	Als host wil ik een spel kunnen opslaan.

Usecase	Beschrijving
Laden opgeslagen spel	Als host wil ik een opgeslagen spel kunnen laden.

2. Domain Model

In dit hoofdstuk worden alle domeinen beschreven inclusief relaties, eigenschappen en multipliciteit. Het doel van dit hoofdstuk is om alle domeinen die relevant zijn voor dit project in een volledig diagram te plaatsen zodat hiernaar gerefereerd kan worden tijdens het opstellen van het SDD.



Figuur 2: Domain Model

2.1. Glossary

Tabel 3: Glossary van het Domain Model met een beschrijving voor iedere conceptuele klasse.

Concept	Beschrijving
Character	Representeert een entiteit die interacteert met het spel. Kan bestaan uit een player, creature of agent.
Player	Representeert een speler die zich in het spel bevindt. Een speler is een vorm van een character en kan worden geïdentificeerd aan de hand van zijn naam. De speler bevat ook alle items van de inventaris.
Item	Representeert een voorwerp dat zich in de wereld bevindt. De speler en de wereld bevatten samen alle items.
Agent	Representeert de agent die het karakter van de speler aanstuurt als deze is ingeschakeld.
World	Representeert de wereld waarin het spel zich afspeelt.

Tile	Representeert een tegel in de wereld waar characters overheen kunnen lopen. Er zijn verschillende tegels, deze worden onderscheiden met behulp van een symbool.
HazardousTile	Representeert een tegel in de wereld waarop characters damage krijgen.
LootableTile	Representeert een tegel waarop een voorwerp ligt.
Creature	Representeert verschillende soorten creatures zoals een zombie of een player.
Inventory	Representeert een verzameling van verschillende voorwerpen.
Weapon	Representeert het wapen die de speler draagt.
Armor	Representeert het schild die de speler beschermt.
Consumable	Representeert de spullen die de speler heeft zoals medicijnen of eten.
Configuration	Representeert de instellingen van een agent. Dit is een serie zinnen met daarin het verschillend gedrag.
Question	Representeert de vragen waarmee de configuratie kan worden opgebouwd.
Game	Representeert het spel dat gespeeld wordt door spelers
Chat	Representeert de berichten die tijdens het spel worden verstuurd
Lobby	Representeert de plek waar spelers terecht komen voor dat een spel wordt gestart

3. Usecasebeschrijvingen

In dit hoofdstuk staan de usecases gedetailleerd uitgewerkt. Elke usecase bevat een basic flow en een of meerdere alternative flows als deze nodig is. Verder heeft elke usecase een omschrijving, primary actor, stakeholders, interests en pre- en postcondities.

3.1. Configureren agent

Hieronder staat de usecase die gaat over het configureren van een agent.

3.1.1. Fully-dressed usecase description

Tabel 4: Fully dressed usecase description: configureren agent

Usecase: Configureren agent	
Description:	De speler configureert het gedrag van de agent, zoals het gedrag bij vechten en verkennen door middel van vragen.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Het spel moet gestart zijn.
Postconditions:	• Er is een configuratiebestand aangemaakt. • De configuratie bevat geen opbouw fouten.
Basic Flow (Main Success Scenario):	

Actor action	System responsibility
1. Speler selecteert agent editor.	
	2. Systeem begint vraag loop.
	3. Systeem stelt eerstvolgende vraag aan speler.
4. Speler geeft antwoord.	
	5. Systeem controleert of het antwoord een toegestaan antwoord is.
	6. Systeem controleert of er een volgende vraag is. → Ga naar 3, anders 7.
	7. Systeem controleert of de speler heeft aangegeven dat die extra regels wilt configureren. Ga naar 8, anders 10.
8. Speler geeft extra regels op.	
	9. Systeem controleert of de extra regels voldoen aan de syntaxis en semantiek.
	10. Systeem parset de agentregels.
	11. Systeem heeft een configuratiebestand opgeslagen.
	12. Systeem gaat naar het menu scherm.
Alternative flow A	
Actor action	**System responsibility**
4A1. Speler geeft verkeerde input.	
	4A2. Systeem detecteert foute input.
	4A3. Systeem geeft juiste opties aan.
	4A4. Systeem print vraag opnieuw.
	4A5. Gaat verder naar 6.
Alternative flow B	
Actor action	**System responsibility**
4B1. Speler geeft als input 'Help'.	
	4B2. Systeem print informatie één voor één het gedrag van de opties.
	4B3. Gaat verder naar 6.
Alternative flow C	
Actor action	**System responsibility**
4C1. Speler geeft als input 'Stop'.	
	4C2. Systeem stop met vragen stellen.
	4C3. Usecase stopt en het systeem gaat naar het menu scherm.

3.2. Inschakelen agent

Hieronder staat de usecase die gaat over het inschakelen van een agent.

3.2.1. Fully-dressed usecase description

Tabel 5: Fully dressed use case description: inschakelen agent

Usecase: Inschakelen agent	
Description:	De speler schakelt tijdens het spelen de agent in om de besturing van het karakter over te nemen.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel. • Zijn agent is nog niet ingeschakeld. • Er is een configuratiebestand voor de agent aangemaakt.
Postconditions:	• De agent is ingeschakeld en de besturing van het karakter overgenomen.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert het commando *"replace"* in.	
	2. Het systeem haalt het configuratiebestand van de agent op.
	3. Het systeem schakelt de agent in om de besturing van het karakter over te nemen.
	4. Het systeem voert acties uit aan de hand van het configuratiebestand.
Alternative flow A	
Actor action	**System responsibility**
	2A1. Het systeem geeft een melding aan de speler.
	2A2. Het systeem laadt een standaardconfiguratie in.
	2A3 Het systeem gaat verder met stap 3.

3.3. Uitschakelen agent

Hieronder staat de usecase die gaat over het uitschakelen van een agent.

3.3.1. Fully-dressed usecase description

Tabel 6: Fully dressed usecase description: uitschakelen agent

Usecase: Uitschakelen agent	
Description:	De speler schakelt tijdens het spelen de agent uit en neemt de besturing van het karakter weer over.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel. • Zijn agent is ingeschakeld.
Postconditions:	• De agent is uitgeschakeld en de speler heeft de besturing van het karakter overgenomen.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert het commando *"replace"* in.	
	2. Het systeem schakelt de agent uit en stopt de automatische uitvoering van acties.
	3. Het systeem geeft de besturing van het karakter terug aan de speler.

3.4. Bewegen speler

Hieronder staat de usecase die gaat over het bewegen van een speler.

3.4.1. Fully-dressed usecase description

Tabel 7: Fully dressed usecase description: bewegen speler

Usecase: Bewegen speler	
Description:	De speler beweegt tijdens het spel op een vakje naar rechts, links, boven of onder.
Primary actor:	Speler

Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel.
Postconditions:	• De speler heeft een nieuwe positie en is succesvol bewogen.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert het commando *move <richting> <hoeveelheid stappen>* in.	
	2. Het systeem bekijkt en ontleed het commando.
	3. Het systeem controleert of de speler nog genoeg stamina heeft.
	4. Het systeem controleert voor elke stap of er een obstakel in de weg staat.
	5. Het systeem geeft de verandering door aan alle spelers.
	6. Het systeem veranderd de positie en stamina van de speler en laad het scherm opnieuw.
Alternative flow A	
Actor action	**System responsibility**
	4A1. Systeem verplaatst de speler tot de laatst mogelijke stap.
	4A2. Het systeem geeft een melding aan de speler dat de volledige verplaatsing niet voltooid kan worden omdat er een obstakel in de weg staat.
	4A3. Systeem gaat door naar stap 5.
Alternative flow B	
	4B1. Het systeem geeft een melding aan de speler dat de verplaatsing niet voltooid kan worden,

	omdat hij te veel stappen in gaf.
Einde usecase.	
Alternative flow C	
	4C1. Het systeem geeft een melding aan de speler dat de verplaatsing niet voltooid kan worden, omdat hij te veel stappen in gaf.
Einde usecase.	
Alternative flow D	
	4A1. Systeem verplaatst de speler tot de laatst mogelijke stap.
	4A2. Het systeem geeft een melding aan de speler dat de volledige verplaatsing niet voltooid kan worden omdat hij te weinig stamina had.
	4A3. Systeem gaat door naar stap 5.

3.5. Aanvallen vijand

Hieronder staat de usecase die gaat over het aanvallen van een vijand.

3.5.1. Fully-dressed usecase description

Tabel 8: Usecase: Aanvallen vijand

Usecase: Aanvallen vijand	
Description:	Een speler kan een vijand aanvallen.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler → moet acties kunnen uitvoeren met en tegen andere spelers
Preconditions:	• Speler bevindt zich in een lopend spel. • Speler is in het bezit van een wapen dat bij de aanvalssoort past.
Postconditions:	• Een speler heeft een vijand aangevallen en er is stamina afgehaald, de vijand heeft damage gekregen. • Er is armor of health

	afgegaan van de vijand.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface een attack-commando[1] in. [1] Attack-commando: attack <direction> [1] Attack-commando: slash <direction> [1] Attack-commando: shoot <direction>	
	2. Systeem controleert of de speler die aanvalt nog genoeg stamina heeft.
	3. Systeem controleert of er een vijand staat in de opgegeven richting.
	4. Systeem verwerkt schade aan vijand.
	5. Systeem geeft een melding met de schade die aangebracht is.
Alternative flow A **Er is niet genoeg stamina om een aanval uit te voeren (begin: basic flow, stap 2)**	
Actor action	**System responsibility**
	A1. Systeem vraagt stamina van speler die aanvalt op.
	A2. Systeem ziet dat speler die aanvalt niet genoeg stamina heeft.
	A3. Systeem geeft de melding dat de speler niet genoeg stamina heeft.
Einde usecase.	
Alternative flow B **Er is geen vijand op het aangegeven vak (begin: basic flow, stap 3)**	
Actor action	**System responsibility**

	A1. Systeem ziet geen speler in de opgegeven richting.
	A2. Systeem geeft de melding dat er geen vijand op de tegel staat.
Einde usecase.	

3.6. Inspecteren tegel

Hieronder staat de usecase die gaat over het inspecteren van een tegel.

3.6.1. Fully-dressed usecase description

Tabel 9: Usecase: Inspecteren tegel

Usecase: Inspecteren tegel	
Description:	Een speler wil kijken welke voorwerpen op de huidige tegel liggen.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel.
Postconditions:	• De speler weet welke voorwerpen er op de huidige tegel liggen.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando in om te zoeken naar voorwerpen.	
	2. Systeem geeft een lijst met alle voorwerpen die op de huidige tegel liggen.

3.7. Oppakken voorwerp

Hieronder staat de usecase die gaat over het oppakken van een voorwerp.

3.7.1. Fully-dressed usecase description

Tabel 10: Usecase: Oppakken voorwerp

Usecase: Oppakken voorwerp

Description:	Een speler wil een voorwerp oppakken.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel.
Postconditions:	• Een speler heeft een voorwerp opgepakt.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando in om een voorwerp op te pakken.	
	2. Systeem bevestigt dat er een voorwerp ligt.
	3. Systeem verwijdert voorwerp uit de spelwereld.
	4. Systeem voegt voorwerp toe aan inventaris van de speler.
Alternative flow A **Er is geen voorwerp op het aangegeven vak (begin: basic flow, stap 2)**	
Actor action	**System responsibility**
	A1. Systeem ziet geen voorwerp op de tegel.
	A2. Systeem geeft de melding dat er geen voorwerp ligt.
Einde usecase.	
Alternative flow B **Er is geen voorwerp op de tegel met de opgegeven naam (begin: basic flow, stap 2)**	
Actor action	**System responsibility**
	A1. Systeem herkent het opgegeven voorwerp niet.
	A2. Systeem geeft de melding dat er geen voorwerp ligt.
Einde usecase.	

3.8. Weggooien voorwerp

Hieronder staat de usecase die gaat over het weggooien van een voorwerp.

3.8.1. Fully-dressed usecase description

Tabel 11: Usecase: Weggooien voorwerp

Usecase: Weggooien voorwerp	
Description:	Een speler wil een voorwerp weggooien.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Een speler heeft een voorwerp in de inventaris.
Postconditions:	• Een speler heeft het voorwerp niet meer in de inventaris.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando in om een voorwerp weg te gooien.	
	2. Systeem bevestigt dat het voorwerp in de inventaris van de speler zit.
	3. Systeem verwijdert voorwerp uit de inventaris.
Alternative flow A **Het aangegeven voorwerp zit niet in de inventaris (begin: basic flow, stap 2)**	
Actor action	**System responsibility**
	A1. Systeem ziet aangegeven voorwerp niet in inventaris.
	A2. Systeem geeft de melding dat aangegeven voorwerp niet in de inventaris zit.
Einde usecase.	
Alternative flow B **Er is geen voorwerp op in de inventaris met de opgegeven naam (begin: basic flow, stap 2)**	

Actor action	System responsibility
	A1. Systeem herkent de opgegeven naam van het voorwerp niet.
	A2. Systeem geeft de melding dat er geen voorwerp gevonden is met de opgegeven naam.
Einde usecase.	

3.9. Inspecteren voorwerp

Hieronder staat de usecase die gaat over het inspecteren van een voorwerp.

3.9.1. Fully-dressed usecase description

Tabel 12: Usecase: Inspecteren voorwerp

Usecase: Inspecteren voorwerp	
Description:	Een speler wil informatie over een voorwerp in zijn inventaris.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel.
Postconditions:	• Een speler heeft informatie ontvangen over een voorwerp in een specifiek inventarisslot.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando om een voorwerp te inspecteren in.	
	2. Systeem checkt of het opgevraagde inventarisslot een voorwerp bevat.
	3. Systeem geeft informatie over ingevoerde object.
Alternative flow A **Er zit geen voorwerp in de opgevraagde inventarisslot (begin: basic flow, stap 3)**	

Actor action	System responsibility
	A3. Systeem geeft de melding dat aangegeven slot van de inventaris leeg is.
Einde usecase.	

3.10. Gebruiken voorwerp

Hieronder staat de usecase die gaat over het gebruiken van een voorwerp.

3.10.1. Fully-dressed usecase description

Tabel 13: Usecase: Weggooien voorwerp

Usecase: Weggooien voorwerp	
Description:	Een speler wil een voorwerp gebruiken.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel.
Postconditions:	• Een speler heeft een bruikbaar voorwerp in zijn inventaris.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando om een voorwerp te gebruiken in.	
	2. Systeem bevestigt dat het ingevoerde voorwerp in de inventaris van de speler zit op de opgegeven positie.
	3. Systeem verwijdert voorwerp uit de inventaris op de opgegeven positie.
	4. Systeem verhoogt de statistieken van de speler die het voorwerp verhoogt.
Alternative flow A **Het aangegeven voorwerp zit niet in de inventaris (begin: basic flow, stap 2)**	

Actor action	System responsibility
	A1. Systeem ziet aangegeven voorwerp niet in inventaris.
	A2. Systeem geeft de melding dat aangegeven voorwerp niet in de inventaris zit.
Einde usecase.	
Alternative flow B **Er is geen voorwerp op in de inventaris met de opgegeven naam(begin: basic flow, stap 2)**	
Actor action	**System responsibility**
	A1. Systeem herkent de opgegeven naam van het voorwerp niet.
	A2. Systeem geeft de melding dat er geen voorwerp gevonden is met de opgegeven naam.
Einde usecase.	

3.11. Zoeken doel

Hieronder staat de usecase die gaat over het zoeken van een doel.

3.11.1. Fully-dressed usecase description

Tabel 14: Usecase: Zoeken doel

Usecase: Zoeken doel	
Description:	Een speler wil een doel zoeken
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel.
Postconditions:	• Een speler weet de directie van het dichtstbijzijnde doel
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando om rond te kijken in.	

	2. Systeem zoekt het dichtstbijzijnde doel.
	3. Systeem toont directie dichtstbijzijnde doel.

3.12. Versturen chatbericht

Hieronder staat de usecase die gaat over het versturen van een chatbericht.

3.12.1. Fully-dressed usecase description

Tabel 15: Usecase: Versturen chatbericht

Usecase: Versturen chatbericht	
Description:	Een speler of host kan een chatbericht versturen tijdens een spelsessie om op deze manier te communiceren met andere spelers.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Een spel is gestart.
Postconditions:	• Een bericht is aangekomen bij de andere spelers.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando om chat berichten te versturen in.	
	2. Systeem verstuurt het bericht naar de host.
	3. Host systeem ontvangt bericht van de netwerk switch.
	4. Host systeem voert het chatbericht commando uit.
	5. Host systeem toont chatbericht in de interface.
	6. Host verstuurd chatbericht naar alle clients.
	7. Client ontvangt bericht van de netwerk switch.
	8. Client voert het chatbericht commando uit.

	9. Client toont chatbericht in de interface.
Alternative flow A **Speler voert een verkeerd commando in (begin: basic flow, stap 1).**	
Actor action	**System responsibility**
	A1. Systeem geeft foutmelding dat een verkeerd commando is ingevoerd.
Usecase gaat verder naar stap 1 van de basic flow.	
Alternative flow B **Systeem is niet verbonden met het internet (begin: basic flow, stap 2).**	
	B1. Systeem toont foutmelding in de interface.
Einde usecase.	

3.13. Aanmaken lobby

Hieronder staat de usecase die gaat over het aanmaken van een lobby.

3.13.1. Fully-dressed usecase description

Tabel 16: Usecase: Aanmaken lobby

Usecase: Aanmaken lobby	
Description:	Een speler kan een nieuwe lobby maken vanuit het spel en krijgt de optie om spel- en lobby-instellingen te configureren. Zodra de lobby succesvol is aangemaakt wordt deze speler de host van de lobby.
Primary actor:	Speler
Stakeholders and interests:	• SH-2 Software developer → moet de configuratie van agent mogelijk maken • SH-1 Speler → Moet in staat zijn de editor te gebruiken. • SH-1 Speler → Krijgt de titel host
Preconditions:	• Het programma moet actief zijn. • De speler heeft nog geen spel gestart.

Postconditions:	• Lobby is aangemaakt. • Instellingen zijn ingesteld naar wens. • Andere spelers kunnen aan de lobby deelnemen.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler kiest om een spel te hosten in het spelmenu.	
	2. Systeem geeft de optie om het spel te configureren.
3. Speler kiest om het spel niet te configureren.	
	4. Systeem vraagt om een gebruikersnaam op te geven.
5. Speler geeft een gebruikersnaam.	
	6. Systeem vraagt om een sessienaam op te geven.
7. Speler geeft een sessienaam.	
	8. Systeem maakt een spel sessie aan met de gegeven instellingen.
	9 Systeem geeft de lobby weer.
Einde usecase.	
Alternative flow A **Speler kiest om het spel te configureren (begin: basic flow, stap 3).**	
Actor action	**System responsibility**
A1. Speler kiest om het spel wel te configureren.	
	A2. Systeem geeft de optie om de NPC-moeilijkheidsgraad in te stellen.
A3. Speler kiest een moeilijkheidsgraad.	
	A4. Systeem geeft de optie om in te stellen hoe vaak voorwerpen voorkomen.

A5. De speler kiest hoe vaak voorwerpen voorkomen.	
Usecase gaat verder naar stap 4 van de basic flow.	
Alternative flow B **Speler geeft geen gebruikersnaam op (begin: basic flow, stap 5).**	
Actor action	**System responsibility**
B1. Speler geeft geen gebruikersnaam op.	
	B2. Systeem gebruikt een standaardnaam.
Usecase gaat verder naar stap 6 van de basic flow.	
Alternative flow C **Speler geeft geen sessie naam op (begin: basic flow, stap 7).**	
Actor action	**System responsibility**
C1. Speler geeft geen sessienaam op.	
	C2. Systeem gebruikt een standaardnaam.
Usecase gaat verder naar stap 8 van de basic flow.	

3.14. Deelnemen lobby

Hieronder staat de usecase die gaat over het deelnemen aan een lobby.

3.14.1. Fully-dressed usecase description

Tabel 17: Usecase: Deelnemen lobby

Usecase: Deelnemen lobby	
Description:	Een speler kan aan een bestaande spel lobby deelnemen.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Host (Een type speler) • SH-1 Speler

Preconditions:	• Er is door de host een spelsessie aangemaakt. • Speler heeft verbinding met de netwerk switch.
Postconditions:	• De nieuwe speler is toegevoegd aan de spelsessie. • Alle spelers in de spelsessie zijn op de hoogte gebracht van de nieuwe speler.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler gaat naar het scherm voor spelsessies	
	2. Het systeem toont beschikbare spelsessies.
3. Speler kiest een sessie en geeft aan deze te willen joinen. De speler geeft ook aan welke gebruikersnaam moet worden gebruikt.	
	4. Systeem bevestigt en voegt speler aan spelsessie toe.
	5. Systeem brengt overige spelers uit spelsessie op de hoogte van nieuwe speler.
Alternative flow A **Spelsessie zit vol (begin: basic flow, stap 4).**	
Actor action	**System responsibility**
	A1. Systeem geeft foutmelding dat spelsessie vol zit.
Alternative flow B **Er zijn geen beschikbare spelsessies (begin: basic flow, stap 2).**	
Actor action	**System responsibility**
	B1. Systeem laat geen spelsessies zien.
Usecase gaat verder naar stap 1 van de basic flow.	

Alternative flow C **Speler heeft niet eerder aan het opgeslagen spel deelgenomen (begin: basic flow, stap 4).**	
Actor action	**System responsibility**
	C1. Systeem geeft de foutmelding dat speler het spel niet kan joinen omdat hij niet eerder aan het opgeslagen spel heeft deelgenomen.
Einde usecase.	

3.15. Verlaten lobby

Hieronder staat de usecase die gaat over het verlaten van een spel.

3.15.1. Fully-dressed usecase description

Tabel 18: Usecase: Verlaten spel

Usecase: Verlaten lobby	
Description:	Een speler wil een spel verlaten.
Primary actor:	Speler
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel of lobby.
Postconditions:	• Speler bevindt zich niet langer in een spel of lobby.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Speler voert in de interface het commando in om het spel af te sluiten.	
	2. Het systeem verwijderd de speler uit het spel.
Einde usecase.	

3.16. Wijzigen spelconfiguratie in-game

Hieronder staat de usecase die gaat over het wijzigen van de spelinstellingen tijdens het spelen van een spel.

3.16.1. Fully-dressed usecase description

Tabel 19: In-game configuratie aanpassen

Usecase: In-game configuratie aanpassen	
Description:	Als host wil ik tijdens het spelen instellingen zoals een monstermoeilijkheidsgraad kunnen aanpassen.
Primary actor:	Host
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• De usecase starten spel is succesvol uitgevoerd
Postconditions:	• De configuratie is opgeslagen bij de host en de back-up-host
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Host voert in de interface het commando in om de monstermoeilijkheidsgraad aan te passen.	
	2. Systeem wijzigt de moeilijkheidsgraad bij de host en de back-up-host.
Alternative flow A **(begin: basic flow, stap 1)**	
Actor action	**System responsibility**
A1. Host voert in de interface het commando in om de spawn-rate van de voorwerpen aan te passen.	
	A2. Systeem wijzigt de spawn-rate bij de host en de back-up-host.
Alternative flow B **(begin: basic flow, stap 1)**	
Actor action	**System responsibility**
B1. Speler die niet de host is voert in de interface het commando in om de monstermoeilijkheidsgraad aan te passen.	

	B2. Systeem ziet dat de speler niet de host is en voert daarna niks uit.
Einde usecase.	
Alternative flow C **(begin: basic flow, stap 1)**	
Actor action	**System responsibility**
C1. Speler die niet de host is voert in de interface het commando in om de spawn-rate van de voorwerpen aan te passen.	
	C2. Systeem ziet dat de speler niet de host is en voert daarna niks uit.
Einde usecase.	

3.17. Starten spel

Hieronder staat de usecase die gaat over het starten van een spel.

3.17.1. Fully-dressed usecase description

Tabel 20: Usecase: Starten spel

Usecase: Starten spel	
Description:	Een host wil het spel starten.
Primary actor:	Host
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• UC-4: aanmaken lobby uitgevoerd.
Postconditions:	• Het spel is gestart.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. De host voert in de interface het commando in om de sessie te starten.	
	2. Hostsysteem maakt de wereld op basis van een gegenereerde seed en instellingen.

	3. Hostsysteem verstuurd een start bericht naar de netwerk switch met de seed en alle instellingen.
	4. Clientsysteem ontvangt bericht van de netwerk switch.
	5. Clientsysteem maakt de wereld op basis van de seed en gegeven instellingen.
Alternative flow A **Host voert verkeerde commando in (begin: basic flow, stap 1).**	
Actor action	**System responsibility**
A1. Host voert verkeerde commando in.	
	A2. Systeem geeft foutmelding dat een verkeerde commando is uitgevoerd.
Usecase gaat verder naar stap 1 van de basic flow.	
Alternative flow B **Speler die niet host voert het startcommando in (begin: basic flow, stap 1).**	
Actor action	**System responsibility**
B1. Speler die niet host is voert het commando in om het spel te starten.	
	B2. Systeem geeft foutmelding dat de speler geen recht heeft om het spel te starten.
Usecase gaat verder naar stap 1 van de basic flow.	
Alternative flow C **Systeem is niet verbonden met het internet (begin: basic flow, stap 1).**	
Actor action	**System responsibility**
	C1. Systeem is niet verbonden met het internet.
	C2. Systeem toont foutmelding in de interface.
Usecase gaat verder naar stap 1 van de basic flow.	

Alternative flow D **Host start een opgeslagen spel op (begin: basic flow, stap 1).**	
Actor action	**System responsibility**
	A2. Hostsysteem maakt de wereld op basis van de opgeslagen seed en instellingen van het geselecteerde opgeslagen spel.
Einde usecase.	

3.18. Beëindigen spel

Hieronder staat de usecase die gaat over het beëindigen van een spel.

3.18.1. Fully-dressed usecase description

Tabel 21: Usecase: Beëindigen spel

Usecase: Beëindigen spel	
Description:	Als host van een lobby kun je het gehele spel stoppen, zodat het spel opgeslagen wordt en iedereen de lobby verlaat.
Primary actor:	Host
Stakeholders and interests:	• SH-1 Speler
Preconditions:	• Speler bevindt zich in een lopend spel.
Postconditions:	• Het spel is afgesloten. • Niemand bevindt zich meer in het spel.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Host voert in de interface het commando in om het spel af sluiten.	
	2. Systeem beëindigd het spel.
Einde usecase.	

3.19. Opslaan spel

Hieronder staat de usecase die gaat over het opslaan van een spel.

3.19.1. Fully-dressed usecase description

Tabel 22:

Usecase: Hervatten spel	
Description:	Als host wil ik een spel kunnen opslaan zodat er later verder gespeeld kan worden.
Primary actor:	Host
Stakeholders and interests:	SH-1 Speler → moet pauze kunnen houden in het online spel
Preconditions:	Spel is actief
Postconditions:	Spel is opgeslagen en gestopt
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**
1. Host en clients verbreken de verbinding	
	2. Systeem slaat het spel op.

3.20. Laden opgeslagen spel

Hieronder staat de usecase die gaat over het laden van een opgeslagen spel.

3.20.1. Fully-dressed usecase description

Tabel 23:

Usecase: Hervatten spel	
Description:	Als host wil ik een opgeslagen spel laden.
Primary actor:	Host
Stakeholders and interests:	• SH-1 Host (Een type speler) • SH-1 Speler → moet pauze kunnen beëindigen in het online spel
Preconditions:	Gedurende het spelen van het spel is het spel opgeslagen.
Postconditions:	Opgeslagen spel is geladen en actief.
Basic Flow (Main Success Scenario):	
Actor action	**System responsibility**

1. Host selecteert de optie om een sessie te laden.	
	2. Systeem toont de beschikbare sessies.
3. Host selecteert het opgeslagen spel waar hij mee verder wilt spelen.	
	4. Systeem opent de lobby.
5. Clients kunnen het spel joinen. (zie usecase deelnemen opgeslagen spel)	
6. Host start het spel.	
	7. Het systeem start het spel met de state waarin het opgeslagen was.

4. Functional requirements

In Tabel 24: Functional requirements staan de functionele requirements. De prioritering is gedaan met behulp van MoSCoW-methode.

Tabel 24: Functional requirements

Code	Beschrijving	MoSCoW-prioritering
FR1	Het spel kan gespeeld worden in een terminal van 80x24.	Must
FR2	Het spel is tekst-gebaseerd.	Must
FR3	De wereld is oneindig.	Must
FR4	De wereld bestaat uit begaanbare en onbegaanbare vlakken.	Must
FR5	Alle begaanbare vakken moeten bereikbaar zijn.	Must
FR6	Begaanbare vlakken hebben verschillende kosten om overheen te gaan.	Must
FR7	Het gedrag van een monster kan aangepast worden.	Must
FR8	De attributen van een monster kan aangepast worden.	Must
FR9	Als een speler dood is wordt deze uit het spel gezet.	Must
FR10	Meerdere spelers kunnen deelnemen aan hetzelfde realtime spel zonder beurten.	Must
FR11	Er is een gamemode 'capture the flag'.	Must

Code	Beschrijving	MoSCoW-prioritering
FR12	Er is een gamemode 'last man standing'.	Must
FR13	Tijdens de gamemode 'capture the flag' kunnen spelers met elkaar chatten.	Must
FR14	De agents moeten geprogrammeerd worden in een zelf ontwikkelde taal.	Must
FR15	Het spel wordt aangevuld met software agents die mensen simuleren.	Must
FR16	Het gedrag van een agent lijkt zodanig “intelligent” dat de overige spelers niet in de gaten hebben dat een speler vervangen is door zijn of haar agent.	Must
FR17	Monsters moeten intelligent gedrag vertonen en reageren op de omstandigheden.	Must
FR18	De monsters moeten met behulp van dezelfde taal(mechanismen) als de agents geprogrammeerd kunnen worden.	Must
FR19	Als tijdens het spel een speler uitvalt, dan moet het systeem deze onmiddellijk vervangen door een nieuwe agent–gebaseerde speler zonder dat het spel daar merkbaar door beïnvloed wordt.	Must
FR20	De wezens moeten bestuurd worden met een “text command interface”.	Must
FR21	Het spel moet op normale desktops gespeeld kunnen worden (Linux, Mac en Windows).	Must
FR22	De wereld wordt voor ieder nieuw spel opnieuw gegenereerd.	Must
FR23	Het spel moet opgeslagen kunnen worden.	Must
FR24	De back-up-host neemt de plek over van de host als deze de verbinding verliest.	Must
FR25	Zodra er geen back-up-host is, wordt een speler direct als back-up-host aangewezen.	Must
FR25	Vraag en antwoord editor mag maximaal 5 vragen groot zijn.	Should
FR26	Vraag en antwoord editor geeft hulp aan de speler als hij niet weet wat de commando's inhoud.	Should
FR27	Vraag en antwoord editor controleert de input van de speler en vangt deze op zodat de speler geen fouten maakt.	Should
FR28	Speler kan deelnemen aan een opgeslagen spel zodra hij in het originele spel heeft deelgenomen.	Must

5. Non-functional Requirements

De niet-functionele requirements voor dit project zijn te vinden in het Software Architecture Document: SAD samengevoegd. Deze worden beschreven van hoofdstuk 5.1.3 tot en met 5.1.10. De niet-functionele requirements zijn aan de hand van het ISO 25010 standaard ingedeeld.

6. User interface sketches

De gemaakte schetsen van de gebruikersinterface zijn te vinden in het hoofdstuk user interface van het Game Design Document. In Tabel 25: Schetsen gekoppeld aan high level usecases is te zien welke schets welke usecase betreft.

Tabel 25: Schetsen gekoppeld aan high level usecases

Schetsnaam	Betreft high-level usecases
Begin scherm	UC-4, UC-5, UC-9

Schetsnaam	Betreft high-level usecases
Host game lobby	UC-7
Join game lobby	UC-5
Game gebruiker interface	UC-2, UC-8, UC-10, UC-11, UC-12

7. Commando's

De uitvoerbare commando's zijn terug te vinden in het Game Design Document in het hoofdstuk Commando's.

8. Agentconfiguratie

In het onderzoek programmeertaal agents is er onderzoek gedaan naar een simpele programmeertaal. In H5.3 staat een concept voor de input en output van de zelfgemaakte taal. Daarnaast is er ook gekeken naar de semantiek van de taal. In dit hoofdstuk zal kort deze input, output en de semantiek worden beschreven. Voor verdere informatie kan het onderzoek geraadpleegd worden.

De programmeertaal bevat een eigen code die er als volgt uitziet:

Input code - Exploration

```
explore

    collect
        when inventory contains item then skip otherwise check

    check
        when item strength greater than current then swap
```

In bovenstaand codevoorbeeld staat dat als een NPC een item tegenkomt, en hij er geen heeft of als het item sterker is, dat hij deze dan oppakt. Onze eigen taal maakt gebruikt van een aantal keywords, en als deze niet benoemd worden dan wordt de configuratie, waar de taal gebruikt wordt, afgekeurd. Deze keywords staan beschreven in tabel 4 van hoofdstuk H5.3.

De code die de gebruiker invoert moet ook gecheckt worden op de semantiek. Indien dit niet gebeurd kan een speler onlogische code invoeren waar de agent niks mee kan. Daarom is in H5.3 ook gekeken naar de checks die uitgevoerd moeten worden. Deze checks zijn:

- Settings: "combat", "explore".
- Actions: "walk", "attack", "grab", "drop", "flee", "use", "replace", "engage", "collect".
- Actions die verder te programmeren zijn: "engage", "collect".
- Action "use" moet opgevolgd worden met een Item.
- Items, Stats en Tiles zie Game Design Document.
- In de basis regels kan voor de '=' gebruik gemaakt worden van settings.
- In de basisregels wordt een setting gevolgd door: "random", "circle", "line", "none", "defensive", "offensive".
- Otherwise kan niet dezelfde action bevatten als de then.
- Actions die verder te programmeren zijn kunnen in de conditions niet gebruik maken van hun eigen action.
- Contains/ not contains kan alleen gebruikt worden met links inventory en rechts item.
- Greater_than/ less_than/ equals kan alleen gebruiktmet links een stat en rechts een int value.
- Nearby/ finds kan alleen gebruikt worden met links agent en rechts npc/ player/ tiles/ opponent.

Deze checks worden uitgevoerd in de checker binnen de compiler. Daarna wordt de code door de compiler verwerkt en geeft een configuratie bestand terug. Als voorbeeld wordt de code van hierboven gebruikt als input.

Output configuratie - Exploration

```
explore_default_player_comparable = player
explore_default_player_treshold = item
explore_default_player_comparison = finds
explore_default_player_comparison_true = collect

explore_collect_inventory_comparable = inventory
explore_collect_inventory_treshold = item
explore_collect_inventory_comparison = contains
explore_collect_inventory_comparison_true = skip
explore_collect_inventory_comparison_false = check

explore_check_item_comparable = item
explore_collect_inventory_treshold = current
explore_collect_inventory_comparison = greater than
explore_collect_inventory_comparison_true = swap
```

De compiler pakt één voor één de regels van de input code en deelt deze op. Voor elke regel wordt een comparable, threshold en comparison aangegeven. Daarnaast heeft elke regel een uitkomst, zoals '.._comparison_true', deze staat voor elke regel beschreven. Deze uitkomst wordt gedaan als de conditie waar is. In sommige gevallen kan er ook een uitkomst zijn als de conditie niet waar is, '.._comparison_false', deze is optioneel en wordt dus niet voor elke regel weergegeven. De tags van de configuratie zijn gebaseerd op de tags van de input. In Tabel 26 en Tabel 27 staan beschreven hoe de tags van de input worden gebruikt voor de tags van de configuratie.

Input
Explore collect **when inventory contains item then skip otherwise check**
Output
explore_collect_inventory_comparable = inventory explore_collect_inventory_treshold = item explore_collect_inventory_comparison = contains explore_collect_inventory_comparison_true = skip explore_collect_inventory_comparison_false = check

Tabel 26: Voorbeeld Code naar Config

Tekst kleur	Uitleg
Hemelblauw	Setting
Goud	Action
Zeegroen	First comparable
Amber	Second comparable (Threshold)
Koningsblauw	Condition
Rood	Action

Tekst kleur	Uitleg
Pruim	Alternative action

Tabel 27: Legenda