
21 JAAR EN OUDER
De ontwikkeling van een agent-team in de UT-GOAL omgeving

De ontwikkeling van een agent-team in de UT-GOAL omgeving

"21 jaar en ouder"

Groep 21

Jeroen Offerijns	4221524
Alexander Felix	1326236
Robbert van Staveren	1527118
Robert Carosi	4242130
Yorick van Pelt	4230264

12 juni 2013

Faculteit Elektrotechniek, Wiskunde en Informatica
Technische Universiteit Delft

Voorwoord

Dit verslag is het eindverslag van projectgroep 21 voor het project Multi Agent Systemen. De opdracht van het project was om een team van 4 bots te ontwerpen voor het computerspel Unreal Tournament 2004, dat in staat is om een vijandig team van bots met moeilijkheidsgraad 3 te verslaan, in het spel "Capture the Flag".

Dit verslag is bestemd voor mensen met enig verstand van programmeren in PROLOG en GOAL. Daarnaast is het aan te raden om tenminste de projecthandleiding te hebben gelezen.

Lezers die geïnteresseerd zijn in de analyse van de UT-omgeving en beschrijving van de scenario's kunnen terecht in hoofdstuk 3. De beschrijving van de agentrollen is te vinden in hoofdstuk 4 en de beschrijving van de code is te vinden in hoofdstuk 5. Het is aan te raden om tijdens het lezen gebruik te maken van de begrippenlijst die beschreven is in Bijlage C.

Als projectgroep willen wij een aantal mensen bedanken. Birna van Riemsdijk voor het geven van de MAS colleges, Koen Hindriks voor het opzetten van het project, onze studententenassistent Alex Kolpa voor het begeleiden van ons project, Özer Ulusoy en alle individuele groepsleden voor verschaffen van de versnaperingen tijdens elke projectbijeenkomst .

Delft, 12 juni 2013

Robert Carosi
Alexander Felix
Jeroen Offerijns
Yorick van Pelt
Robbert van Staveren

Samenvatting

Voor het project multi agent systemen, kregen wij de opdracht om een team van 4 bots te ontwikkelen dat instaat was om in het computerspel Unreal Tournament 2004, de strijd aan te gaan met een ander team van bots in het spel “Capture the Flag”. Het doel van het spel is om de vlag van de tegenstander te veroveren en deze vervolgens terug te bregen naar de thuisbasis.

Om dit te realiseren hebben wij ervoor gezorgd dat het team van boven wordt aangestuurd door één losstaande agent, de manager. Deze manager zorgt ervoor dat elk lid van het team een apart rol krijgt die afhankelijk is van zijn positie. De twee belangrijkste rollen zijn:
- Flag capturer, deze gaat opzoek naar de vlag
- Flag defender, deze verdedigd de thuisbasis

Naast deze twee rollen zijn er ook 3 ondersteunende rollen:
- Attacker, deze valt een toegewezen doelwit aan
- Defender, deze verdedigd een lid van het team
- Assassin, deze verzameld vooral wapens en valt elke vijandige bot aan die hij tegen komt.

Door telkens de rollen af te wisselen, kan elk te bedenken scenario worden opgelost. Uiteindelijk hebben wij 10 specifieke scenario’s beschreven en één default scenario. Al deze scenario’s zijn vervolgens zorgvuldig getest en uit die tests is gebleken dat wij door het gebruiken van een manager die de rollen steeds afwisseld, de juiste keuze hebben gemaakt. Aangezien wij in staat waren om een team van drie vijandige bots, met moeilijkheidsgraad 3, te verslaan.

Inhoudsopgave

1 Inleiding

Aan ons, een team van 5 informatici, is gevraagd een team van vier bots te ontwikkelen, die zo effectief mogelijk capture the flag speelt, in Unreal Tournament 2004. Unreal Tournament is een zogeheten first-person shooter. In de modus “capture the flag” is het de bedoeling de vlag van de tegenstander te pakken en deze naar de thuisbasis te brengen. Aangezien beide teams dit proberen zullen er tegenstanders uitgeschakeld moeten worden.
Voor het ontwikkelen van een team van agents, om zo effectief mogelijk capture the flag te spelen, zijn vele strategieën mogelijk. In het verslag wordt beschreven welke strategieën zijn overwogen en welke strategie uiteindelijk is gekozen. Door middel van testresultaten en literatuuronderzoek wordt deze keuze toegelicht.

De agents individueel te laten functioneren is een voorbeeld van een strategie. Een alternatief hierop is om naast het team van vier bots, een manager agent aan te maken. Deze verkrijgt informatie van over de omgeving, van de agents. Op basis hiervan kan de manager rollen toekennen aan verschillende agents en er zo voor zorgen dat er vlotte samenwerking plaatsvindt.

De opbouw van dit rapport is als volgt. In hoofdstuk twee staat informatie over de omgeving waarin de bots opereren. Verder staat hier een lijst van eisen waarin duidelijk wordt waaraan het team zal voldoen en niet aan zal voldoen. In hoofdstuk drie worden de verschillende scenario’s beschreven waarin een agent zich kan bevinden en wat een agent moet doen als deze zich in deze situatie bevindt. Verder zal in hoofdstuk vier aan bod komen welke rollen we hebben gekozen. Een agent krijgt een bepaalde rol toegekend als deze zich in een specifieke situatie bevindt. Vervolgens is in 5 de implementatie te vinden. Hierin worden beslissingen m.b.t. de code toegelicht en stukjes van het programma uitgelegd. In hoofdstuk 6 zijn de testresultaten te bekijken, hierin staat de uitwerking van de verschillende tests die zijn uitgevoerd.
Het verslag eindigt met een conclusie waarin terug wordt gekeken op het ontwikkelingsproces en de keuze van de uiteindelijke strategie toegelicht.

2 Programma van eisen en analyse

Bij het oriënteren voor de eerste fase, waarbij een ontwerp uitgedacht wordt, is eerst een lijst met eisen en prioriteiten opgesteld, op basis van de MoSCoW-methode. Hierbij wordt iedere eis geplaatst in één van de vier categorieën:

1. Must-haves: functionaliteiten die absoluut vereist zijn in een juiste implementatie
2. Should-haves: eigenschappen die van belang zijn, en indien mogelijk geïmplementeerd moeten worden
3. Could-haves: mogelijkheden die nuttig zouden kunnen zijn, maar niet per se van belang
4. Won't-haves: zaken die in een vervolgproject aan bod zouden kunnen komen, maar in dit project niet geïmplementeerd zullen worden.

In dit hoofdstuk zijn een aantal van deze MoSCoW-eisen op een rijtje gezet, te beginnen met de Must-haves.

2.1 Must haves

Dit zijn enkele zaken die in een juiste implementatie vereist zijn.

- Communicatie tussen bots - dit is absoluut van belang, omdat het anders vrijwel onmogelijk is een effectief team te maken dat beter is dan teams die wel communiceren. Onder andere om elkaar te verdedigen, samen aan te vallen, dubbel werk te voorkomen etc.
- Modules voor:
    - Defend the flag - als het andere team de vlag probeert te verkrijgen, dan moet deze verdedigd worden.
    - Capture the flag - de vlag van het andere team moet verkregen worden en naar de eigen basis worden gebracht.
    - Return the flag - als de eigen vlag in bezit is van het andere team, dan moet degene die de vlag draagt gedood worden, en de vlag teruggebracht worden naar de basis.
    - Attack (help flag capturer)- als het eigen team de vlag heeft, dan moet degene die hem draagt verdedigd worden tot hij bij de basis is.
- Bots verzamelen wapens en health - als bots niets verzamelen, houden zij alleen de twee beginwapens, en gaan ze dood als ze geen health meer hebben, waardoor opgepakte items en positie steeds weer verloren gaan.
- Bots verzamelen items (ammo, adrenaline, armor) - als bots items verzamelen kunnen ze hiervan de voordelen gebruiken om zo te winnen van tegenstanders. Zo zullen ze minder snel dood gaan als ze armor hebben, door kunnen schieten als ze ammo hebben en combo's uit kunnen voeren als ze adrenaline hebben.
- Bots voeren combo's uit als dit mogelijk is - als bots 100 adrenaline hebben kunnen ze een combo uitvoeren, er zit geen nadeel aan om dit niet te doen, en een klein voordeel om dit wel te doen. De uitdaging zit erin om dit te timen zodat de combo het meest van pas zal komen.

2.2 Should haves

Deze functionaliteiten zijn niet vereist, maar indien mogelijk moeten ze wel geïmplementeerd worden.

- Manager - centralisatie van besluiten van bots is vrij nuttig, zodat er geen onenigheid kan zijn, en niet elke bot zijn eigen besluiten hoeft te nemen, of hoeft te communiceren met de rest van het team. Dit scheelt code in de bots, en hoewel het uiteindelijk meer werk is, zorgt het wel voor een duidelijkere implementatie.
- Chase opponent - als er eenmaal op een tegenstander geschoten is, heeft deze minder health, en waarschijnlijk ook minder ammo. Als een bot de overhand heeft, is het dus nuttig om een tegenstander achterna te gaan om hem helemaal te doden.

2.3 Could haves

Deze opties zouden nuttig kunnen zijn en als er tijd voor is zouden ze geïmplementeerd kunnen worden.

- Change strategy according to score - als de score veel hoger is dan die van het andere team, zouden er meer risico's genomen kunnen worden, of zou een bot 'respawn' kunnen gebruiken om snel health terug te krijgen en op een veilige plek te komen, maar het is goed mogelijk dat dit niet veel voordeel biedt en daarom is dit een Could-have.
- Follow teammate - het kan handig zijn een teamgenoot te volgen, om elkaar te verdedigen in het geval van een aanval of samen andere bots aan te vallen. Het is echter onduidelijk of dit voordeel biedt ten opzichte van het assisteren van bots die toevallig in de buurt zijn. Een nadeel hieraan is dat sommige items verdwijnen bij het oppakken. Zo krijgt de bot die achter loopt minder items.

2.4 Won’t haves

De volgende zaken zullen in dit project niet aan bod komen, maar zouden in een vervolgproject onderzocht kunnen worden:

- Drop items - het is mogelijk om items op te pakken, en deze aan teamgenoten te geven, maar in de praktijk gebeurt dit nooit. Ook zijn de mogelijke voordelen gering ten opzichte van de hoeveelheid werk dat dit kost om te implementeren.
- Cheats - er zal geen gebruik worden gemaakt van cheats tijdens het runnen van het programma.
- Hide with flag - als een bot de vlag heeft verkregen, dan kan hij zich proberen te verstoppen totdat hij gemakkelijk terug naar de basis kan, maar het is mogelijk dat deze strategie niet goed werkt, omdat bots makkelijker andere bots kunnen vinden dan spelers dat kunnen.

3 Analyse UT-omgeving en beschrijving scenario's

In dit hoofdstuk zal een analyse worden gegeven van de UT-omgeving en aan de hand daarvan zullen er tien verschillende scenario's worden opgesteld die kunnen voorkomen tijdens het spelen van capture the flag in Unreal Tournament.

3.1 Analyse UT-omgeving

In deze paragraaf zal eerst een omschrijving worden gegeven van de UT-omgeving waar de bots in rondlopen. Vervolgens hoe de informatie en mogelijke acties uit deze omgeving in een tussenlaag worden omgezet naar percepts en acties in de UT-GOAL omgeving. Deze UT-GOAL omgeving is waar onze eigen agents in uitgevoerd worden. De UT-GOAL omgeving zal tot slot worden vergeleken met de BW4T-omgeving.

Het programma wordt uitgevoerd in de UT-GOAL omgeving. Het programma zorgt ervoor dat het bot-team wordt aangestuurd door agents. Beslissingen van de bots gebeuren in realtime. De beslissingen van de agents kosten echter wat tijd. Hierdoor zijn de acties vaak wat vertraagd maar blijft het een realtime omgeving.

De BW4T-omgeving was ook in realtime, alleen verschilt deze in een paar aspecten van de UT-GOAL omgeving. Zo werd er in de BW4T omgeving telkens een nieuw navigatiepunt als doel opgegeven aan de bot. In de UT-GOAL omgeving is dit niet het geval, omdat de bot anders elke keer een korte tijd stil zal moeten staan om een nieuw doel door te krijgen, wat niet erg praktisch is.

Een ander verschil tussen beide omgevingen is dat in de BW4T-omgeving het bot-team één hoofddoel had die werd opgebroken in een aantal subdoelen. Aan elke bot werd een subdoel als taak toegewezen en vervolgens werd er samen naar het hoofddoel gewerkt. Als de taken van twee bots overlapten werd er aan één van hen een andere taak toegewezen.

In de UT-omgeving hebben wij voor een andere aanpak gekozen. De taken van een bepaalde agent worden toegewezen door een manager agent. Deze zal op basis van score, tijd en de positie van iedere bot een geschikte taak vinden en deze toekennen aan de bot. Voorbeelden van taken die een agent kan hebben zijn: aanvaller, verdediger en vlagdrager. Om deze taken uit te voeren heeft de UT-omgeving verschillende opties gegeven in de vorm van vooraf geprogrammeerde acties en percepts. Zo zorgt de actie "Navigate" er bijvoorbeeld voor dat de agent zich kan verplaatsen in de omgeving en houdt een percept als "status" de statistieken bij zoals health, armor, adrenaline. Aan de hand van de verschillende taken zijn hieronder tien scenario's opgesteld, die daarna kort worden beschreven en vervolgens kort worden samengevat.

3.2 Scenario's

Scenario 1 - Default

Actions: Navigate

Bij het begin van het spel zal er een balans worden gezocht tussen aanvallen en verdedigen. De vlag van de tegenstander zal worden opgehaald door middel van de Navigate actie en deze agent zal door een ander worden verdedigd
De eigen vlag zal verdedigd worden door in de buurt van de vlag rond te gaan lopen. Daarnaast zal er één agent rond gaan lopen door de hele map. Tegelijkertijd zal deze zoveel mogelijk items verzamelen en als doel hebben om tegenstanders uit te schakelen.

Scenario 2 - Flag Taken

Percepts: Flag Status, See Flag, See Other Bot
Actions: Look, Shoot, Navigate

Op het moment dat een tegenstander onze vlag heeft gepakt, wordt door de Flag Status percept aangegeven dat onze vlag gepakt is. De agents kunnen twee dingen doen om te verhinderen dat de tegenstander onze vlag naar zijn basis terug brengt en zo punten scoort.
Ten eerste kunnen de agents de tegenstander uitschakelen door eerst met de look actie de flag-carrier te vinden en vervolgens na het ontvangen van see other bot percept de shoot actie uit te voeren. Dan kan de vlag worden teruggebracht.
Ten tweede kunnen we de vlag van de tegenstanders pakken door middel van de navigate actie. Door dit te doen kan onze vlag niet naar onze basis gebracht worden, om zo punten te scoren.

- Als onze vlag gepakt is, moet de tegenstander die de vlag heeft, worden opgespoord.
- De agents die onze vlag beschermden, gaan op zoek naar de vijandige vlagdrager.
- Als de agents de tegenstander hebben opgespoord, wordt er op hem geschoten totdat de vlag valt.

Scenario 3 - Enemy Flag Taken

Percepts: Flag Status, See Flag
Actions: Navigate

Zodra de vlag van de tegenstander gepakt is door één van onze agents, is het de bedoeling dat deze wordt gebracht naar de thuis basis. Als eerste wordt de navigate actie gebruikt om naar de vlag te gaan. Door de See Flag percept wordt de vlag gezien. Nadat de vlag gezien is zal de bot er naar toe lopen en deze oppakken. Uiteindelijk zal de vlag terug gebracht worden naar de thuisbasis

- Als de vlag van de tegenstander is gepakt, moet deze naar de thuisbasis worden gebracht.
- De overige agents blijven in de buurt van eigen vlag en verdedigen deze.
- De teamgenoot met de vlag moet ook beschermd worden als dit mogelijk is.

Scenario 4 - Health Low

Percepts: Status,See
Actions: Navigate

Wanneer de health van een bot onder de 50% is, zullen health pillen steeds belangrijker worden. Hierdoor zal hij “liever” health oppakken dan bijvoorbeeld wapens en ammo. Om health op te pakken zal de bot de navigate actie gebruiken.

- Bot gaat op zoek naar health totdat hij 100% health heeft.
- De dichtstbijzijnde bot neemt de taak over van de bot die op zoek is naar health.
- Als de eerste bot genoeg health heeft, keert hij terug en zal de tweede bot zijn oude taak hervatten.

Scenario 5 - Flag Dropped

Percepts: Flag Status, See Flag
Actions: Navigate

De vlag valt in een paar gevallen: de tegenstander wordt uitgeschakeld door één of meer agents of de tegenstander schakelt zichzelf uit. Op het moment dat de tegenstander wordt uitgeschakeld, moet de dichtstbijzijnde agent de vlag terugpakken. In het geval dat de tegenstander zichzelf uitschakelt, moet onze vlag zo spoedig mogelijk worden opgespoord en worden teruggebracht naar onze vlagbasis.

- Als de vlag is gevallen, moet deze worden opgespoord.
- De agent die de vlag heeft gevonden, pakt deze.
- De vlag wordt teruggebracht naar de vlagbasis.

Scenario 6 - Enemy Flag Dropped

Percepts: Flag Status, See Flag
Actions: Navigate

Wanneer de vlagdrager de vijandige vlag laat vallen zal het dichtstbijzijnde teamlid deze zo snel mogelijk proberen op te pakken en de rol van vlagdrager overnemen. Vervolgens moet een tweede teamlid de nieuwe vlagdrager beschermen. Met de Flag status weten de bots dat één van de bots de vlag heeft laten vallen, vervolgens moet het dichtstbijzijnde teamlid met de Navigate actie naar de gevallen vlag lopen. Met de See Flag percept ziet hij de vlag en loopt er doorheen om het op te pakken. Hij krijgt nu de opdracht om naar de basis te lopen met de Navigate actie.

- De vlagdrager laat de vlag van de tegenstander vallen.
- Dichtstbijzijnde teamlid zorgt ervoor dat de vlag wordt opgepakt en neemt de rol van vlagdrager over.
- Een tweede teamlid zal worden opgeroepen om de vlagdrager te beschermen.
- De vlagdrager brengt de vlag van de tegenstander naar de thuisbasis.

Scenario 7 - Stuck
Percepts: Navigation
Actions: Respawn

Het kan wel eens gebeuren dat een agent vast komt te zitten ergens in de map. Als dit gebeurt, moet de agent zo snel mogelijk weggaan. Dit kan makkelijk gedaan worden als de agent respawnt. De vastgelopen agent zal dan zelfmoord plegen en vervolgens is het de bedoeling dat hij ergens anders in de map terecht komt en de agent weer verder gaat met zijn taak.

- De agent komt vast te zitten en kan zijn taak niet meer volbrengen.
- De agent respawnt en verschijnt nu op een plek in de map waar hij niet vast zit.
- De agent hervat zijn taak.

Scenario 8 - (Re)Spawn
Percepts: Item, Status
Actions: Respawn, Navigate

In het begin is het de bedoeling dat agents zich gaan voorbereiden op hun taak. Om zichzelf goed te beschermen, moeten deze agents sterke wapens, armor e.d. verzamelen . Zo kunnen zij lang op het slagveld zitten en aanvallen van de vijand beter overleven. Dit is ook het geval als je zelf respawnt, want bij respawnen verlies je alles wat je tot nu toe had verzameld.

- De agent (re)spawnt en geeft zijn status door.
- De agent gaat op zoek naar wapens, ammo.
- De agent krijgt een taak die hij moet gaan doen.

Scenario 9 - Enemy Spotted
Percepts: See Other Bot
Actions: Navigate, Shoot

Wanneer de agent een vijandige bot tegenkomt, is het de bedoeling dat hij deze uitschakelt. Nadat de tegenstander is uitgeschakeld, zal de agent zijn taak hervatten. Als de agent de tegenstander spot, zal hij ook de locatie doorgeven aan de andere bots.

- De agent ziet de vijandige bot.
- De agent gaat naar hem toe totdat de afstand kort genoeg is om op hem te schieten.
- De agent schiet op de tegenstander.

Scenario 10 - Armor Low

Percepts: Status, See
Actions: Navigate

Wanneer de armor van een bot onder de 50% is zal de agent wanneer hij armor of super armor ziet, deze oppakken. Als zijn armor boven de 50% is, dan negeert hij armor behalve super armor. Deze zal hij dan nog steeds oppakken.

- Bot gaat op zoek naar armor totdat hij boven de 50% armor heeft.
- Bot pakt armor op en heeft nu genoeg armor.
- Bot negeert alle armor behalve super armor.

Scenario 11 - Adrenaline Full

Percepts: Status
Actions: Combo action

Via de Status percept kan een agent zien hoeveel adrenaline hij heeft. Wanneer deze vol is, kan de agent een combo uitvoeren met de Combo action. Aan de hand van welke taak de agent heeft, kan hij kiezen om één van de volgende combo's uit te voeren: speed, booster, invisibility en berserk. Booster is handig om je te beschermen bijvoorbeeld voor FlagDefender. Speed kan je gebruiken als je heel snel ergens moet zijn, voornamelijk handig voor degene die de vlag heeft. Invisibility maakt een agent onzichtbaar. Met Berserk kan je 2x zo snel kogels schieten.

- Bot heeft adrenaline full.
- Bot kiest aan de hand van zijn situatie welke combo hij gaat uitvoeren.

Bot voert combo uit.

4 Ontwerp

In dit hoofdstuk wordt beschreven welke rollen er zijn binnen het team, hoe deze worden verdeeld en wat de rollen precies inhouden.
Er zullen een aantal rollen zijn, geïmplementeerd in losse modules:

1. Manager
2. FlagCapturer
3. FlagDefender
4. Attacker
5. Defender
6. Assassin

Er is altijd één agent die de rol Manager heeft. Hij zal tijdens het spel bepalen welke rollen er nodig zijn en deze rollen aanwijzen aan de andere agents. Dit doen we zodat er geen onduidelijkheid ontstaat over de rolverdeling tussen de agents en om de communicatie te versimpelen - de meeste berichten hoeven alleen naar de Manager te worden gestuurd en door de Manager te worden opgevangen.

4.1 Manager

Er zal een agent zijn die niet in het spel rondloopt, maar er als het ware boven staat. Deze Manager zal berichten krijgen van de andere agents over de status of het verloop van het spel en hierop reageren, door bijvoorbeeld de agents andere rollen te geven om op een juiste manier te reageren op een situatie die zich voordoet.

Aanvankelijk zal dit de verdeling zijn:

- 1 FlagCapturer
- 1 Defender (target de FlagCapturer)
- 1 FlagDefender
- 1 Assassin

Dit zorgt voor een goede balans tussen aanvallen en verdedigen.

Als de vlag van de tegenstander wordt gepakt zal de verdeling als volgt zijn:

- 1 FlagCapturer
- 2 Defenders (target de FlagCapturer)
- 1 FlagDefender

Op deze manier wordt de opgepakte vlag goed verdedigd en tegelijk is de eigen vlag ook veilig.

Als de eigen vlag gepakt wordt, zal de verdeling als volgt zijn:

- 1 FlagCapturer
- 3 Attackers (target de vlagdrager van de tegenstander)

Op deze manier zal de vlagdrager van de tegenstander onder druk komen te staan en waarschijnlijk worden gestopt en zal het spel direct doorgaan doordat de vlag van de tegenstander ook gepakt wordt.

4.2 FlagCapturer

Als een agent de rol FlagCapturer heeft, zal hij als voornaamste doel hebben om de vlag van het andere team te veroveren, onafhankelijk of hij op de basis van de tegenstander is of ergens op de grond ligt, en deze terug te brengen naar de eigen basis. Tijdens het bewegen richting de vlag zal hij health en armor oppakken, als verdediging zodra hij de vlag eenmaal heeft, maar geen nieuwe wapens of adrenaline. Zodra hij de vlag heeft, zal hij direct richting de eigen basis gaan, weer door alleen health en armor op te pakken (als het dichtbij genoeg is).

4.3 FlagDefender

Als een agent de rol FlagDefender heeft, zal hij als voornaamste doel hebben om de eigen vlag te beschermen. Indien de eigen vlag nog op de basis is, zal hij in de buurt van die vlag items gaan verzamelen en rond blijven lopen, zodat hij enige tegenstanders makkelijk kan doden.
Is de vlag echter al opgepakt door een tegenstander, dan zal hij gaan navigeren naar de tegenstander en proberen deze te doden en achtereenvolgens de vlag op te pakken.

4.4 Attacker

Als een agent de rol Attacker heeft, zal hij als doel hebben om een door de Manager aangewezen tegenstander - vaak degene met de eigen vlag - te doden. Hij zal alleen items verzamelen die dichtbij de meest directe route naar de tegenstander liggen, om tijd te besparen en er zo snel mogelijk te zijn. De vlag zal hij volledig negeren, behalve als hij er toevallig langskomt (en dus zo dichtbij staat dat het geen extra tijd kost).

4.5 Defender

Als een agent de rol Defender heeft, zal hij als doel hebben om een door de Manager aangewezen teamlid - vaak degene met de vlag van de tegenstanders - te beschermen. Hij zal alle items verzamelen die binnen een redelijke afstand liggen van zijn route en zal de vlag niet oppakken van de basis. In het geval dat het teamlid dat hij moet beschermen toch gedood wordt, zal hij eerst zorgen dat enige tegenstanders binnen zicht gedood worden en daarna de vlag oppakken als deze op de grond is gelegd door het teamlid.

4.6 Assassin

Als een agent de rol Assassin heeft, zal hij als doel hebben om zoveel mogelijk tegenstanders te doden. Dit zal hij doen door eerst zoveel mogelijk items te verzamelen, de hele tijd rond te lopen en zich naar een tegenstander te bewegen als er een gespot wordt door een teamlid dat in de buurt is.

5 Implementatie

In dit hoofdstuk gaan we vertellen hoe de in hoofdstuk 3 beschreven strategie is geimplementeerd in de GOAL programmeertaal. Er zal uitgelegd worden welke keuzes we hebben gemaakt wat betreft taal-constructies en we zullen deze keuzes motiveren.

5.1 Rollenverdeling

Bij het ontwerpen van het systeem besloten we om verschillende rollen te hebben, die dynamisch worden aangewezen, en dit hebben we uiteindelijk geimplementeerd door losse modules te maken.
We hebben voor elke rol een eigen module aangemaakt, inclusief één voor de Manager, en hierin de rol-specifieke code geplaatst. De code die generiek is en gebruikt kon worden bij elke rol, is blijven staan in de main module.
In het begin van elke rol hebben we een constructie toegevoegd waarmee er wordt gekeken of de agent nog wel de huidige rol heeft, of dat hij al een andere rol heeft toegewezen gekregen van de Manager. Als dit het geval is, zal hij de module verlaten en verder gaan met de nieuwe aangewezen module.

Daarnaast is er aan de meeste modules code toegevoegd, zodat het programma meteen de module seeFlag in gaat, als de agent een vlag ziet liggen,. Binnen deze module gaat hij navigeren naar vlaggen die op de grond liggen, onafhankelijk van het team waarvan de vlag is, of naar de vlag van de tegenstander die nog op de basis is.
Verder is er aan elke module code toegevoegd, zodat er een combo zal worden geactiveerd, als de agent 100 adrenaline heeft. De te activeren combo wordt bepaald door te kijken naar de huidige omstandigheden, zoals de rol van de agent, de health van de agent en of hij een vlag ziet.

Als laatste is er aan elke module functionaliteit toegevoegd, die ervoor zorgt, dat de agent zal respawnen in zijn eigen basis, als hij vast komt te zitten. Het programma kijkt of de agent daadwerkelijk stuck is, door te kijken of hij 5 programma cycles achter elkaar stuck is. Eerst keken we namelijk alleen of hij één keer stuck is, maar soms komt hij vanzelf weer vrij en dan zou hij in dit geval toch al respawnen.

Manager

De manager heeft voor verschillende scenario's die kunnen voorkomen een lijst met benodigde rollen. Als de rollen die de bots hebben niet overeen komen met de rollen in de lijst, dan kijkt de manager welke agent een rol heeft die niet nodig is. Naar deze agent wordt een bericht gestuurd met een nieuwe rol.

Voor de rol flag_capturer is een aparte regel toegevoegd voor het geval de vlag wordt vastgehouden. Dan moet de bot die de vlag vast houdt, de rol flag_capturer op zich nemen.

FlagCapturer

De module flag_capturer stuurt een bot aan om de vlag van de tegenstander te halen en terug te brengen naar de eigen basis. Deze module is ontworpen om rekening te houden met drie verschillende situaties waar de vlag zich in kan bevinden. De vlag is home, held of dropped. Voor iedere status is een module gemaakt. Voordat er naar de status van de vlag wordt gekeken, wordt er eerst gekeken of de vlag wordt gezien, er een combo kan worden uitgevoerd en of er nuttige items opgepakt kunnen worden.

Zodra de bot de vlag ziet, zal hij de module seeFlag aanroepen, zoals hierboven is omschreven. Verder zal de bot kijken naar items die hij nodig heeft. Hiervoor gebruikt de module neededItems uit de knowledge, deze is selectiever dan usefulItem die de andere modules gebruiken. Hij zal hier pas naartoe navigeren als de afstand naar dat item klein genoeg is.

Indien de vlag home is, dus bij de basis van de tegenstander, navigeert de bot naar die basis om de vlag te pakken. Als de vlag de status held heeft, een bot heeft de vlag, dan zijn er twee mogelijkheden. Als de bot zelf de vlag heeft, dan navigeert de bot naar de eigen basis. Indien een teamgenoot de vlag heeft, gaat de bot naar de basis van de tegenstander. Hier kan de bot de tegenstander aanvallen, of wachten tot de vlag daar gereset wordt.
Dropped, de laatste status van de vlag, komt voor als de bot die de vlag had, is neergeschoten. Hier is een aparte module van gemaakt om scenario 6, enemy flag dropped, uit te kunnen voeren. Deze module hebben we echter niet meer kunnen uitwerken en nu roept hij de module voor vlag status home aan. Bij de basis van de tegenstander zal hij wachten tot de vlag gereset is.

FlagDefender

De flagDefender maakt gebruik van usefulItem en manhattanDistance om te kijken of er een nuttig item is met een afstand kleiner dan 1500 units tot de eigen basis. In dat geval neemt dan een goal om dit item te pakken. Indien de bot verder dan 1500 units van de eigen basis is, wordt hij door de agent terug naar de eigen basis gestuurd door de agent.

Attacker

De functionaliteit van de attacker is vergelijkbaar met die van de defender. Het enige verschil is dat hij bij de basis van de tegenstander rondloopt in plaats van de eigen basis.

Defender

Bij de module van de rol Defender gaat hij de standaard functionaliteit volgen, met als enige uitzondering dat hij altijd binnen 1500 units van de eigen basis blijft. Dit zorgt ervoor dat hij in de meeste gevallen de basis binnen zicht heeft en hierdoor de eigen vlag goed kan verdedigen.
Echter is deze afstand wel groot genoeg dat hij de mogelijkheid heeft om items binnen handbereik van de vlag te verzamelen.

Assassin

Bij de module van de rol Assassin is het doel van de agent om zoveel mogelijk items te verzamelen en tegenstanders te doden. Dit doet hij door een hoge prioriteit te geven aan het verzamelen van items die in de buurt zijn. Als er geen item in de buurt is zal hij verderweg gaan zoeken naar items, op volgorde van belangrijkheid van het item.

5.2 GOAL-constructies

Bij het schrijven van het hele programma hebben we rekening gehouden met een efficiente opzet en het juiste gebruik van GOAL-specifieke constructies.

Voor het afhandelen van taken hebben we weinig gebruik gemaakt van goals. In plaats daarvan hebben we gekozen voor het toevoegen van beliefs als een bepaald scenario zich voordoet en actions aan te

roepen als er bepaalde beliefs in de belief base zijn. Dit vonden wij de meer straight-forward oplossing, aangezien het de code in ons optiek overzichtelijker maakt en er meer consistentie bij is.

Bij de modules hebben we verschillende action-rules gebruikt. Zo hebben we bij sommige rollen gekozen voor exit=nogoals als we in de code van die rol goals hebben gebruikt om handelingen te voltooien. Hierbij zal hij eerst alle rol-specifieke goals ondernemen en daarna pas de module verlaten en doorgaan naar de main module.
Bij andere rollen hebben we actions i.p.v. goals gebruikt en hier hebben we dan ook exit=noaction toegepast. Dit zorgt ervoor dat hij de module verlaat zodra alle rol-specifieke acties zijn voltooid.

Bij de assassin module is gebruik gemaakt van focus = new, dit zorgt ervoor dat de globale goals niet worden overgenomen, maar er gebruikt gemaakt wordt van een nieuwe set goals. De reden hiervoor is dat de assassin alleen als hoofddoel had om items te verzamelen en niet actief op zoek gaat naar de vlag.

5.3 Code uitleg

In het programma zitten een aantal essentiele regels code die wel wat nadere toelichting kunnen gebruiken; dit zullen we dan ook nu gaan behandelen.

Nuttige wapens

```
goodWeapon(flak_cannon).
goodWeapon(rocket_launcher).
```

Door het spel te testen hebben we gevonden dat de Flak Cannon en de Rocket Launcher de meest effectieve wapens zijn.

Afstand berekenen

```
manhattanDistance(A, B, Distance) :-
        location(X1, Y1, Z1) = A,
        location(X2, Y2, Z2) = B,
        Dx is abs(X1 - X2), Dy = abs(Y1 - Y2), Dz = abs(Z1 - Z2),
        Distance is Dx + Dy + Dz.
```

De afstand tussen twee punten, bijvoorbeeld de huidige agent en de vlag van de tegenstander, wordt berekend met de Manhattan Distance. Hier hebben we voor gekozen, omdat het snel te berekenen is, aangezien de Manager snel moet kunnen bepalen wie er het dicht bij is. Daarnaast hebben we alleen de X en Y coordinaten gebruikt, niet het Z coordinaat, omdat dit het versimpeld en geen grote invloed heeft.

Nuttige combo's

```
usefullCombo(booster) :- status(Health, _, _, _), Health < 40.
usefullCombo(speed) :- role(flag_capturer).
usefullCombo(beserk) :- role(flag_defender).
```

usefullCombo(beserk) :- role(defender).
usefullCombo(beserk) :- role(assassin).
usefullCombo(beserk) :- role(attacker).

Met deze code bepalen we per type combo of het met de huidige rol nuttig is. Een health booster is namelijk alleen nuttig als je low health hebt, dus onder de 40. Daarnaast is speed vooral nuttig voor de FlagCapturer, zodat hij sneller naar de vlag kan en met de vlag terug naar de basis. Berserk is nuttig voor de andere rollen, aangezien ze hiermee beter tegenstanders kunnen vermoorden.

6 Testen van MAS

In dit hoofdstuk zal worden ingegaan op verschillende tests. Eerst zullen de algemene afspraken over het testen worden toegelicht en daarna worden de scenario-tests besproken behorende bij de scenario's beschreven in hoofdstuk 3. Tot slot geven we de resultaten van deze tests.

6.1 Algemene afspraken

Nadat een functionaliteit is geschreven, zal deze direct handmatig gecontroleerd worden door degene die de code heeft geïmplementeerd. Dit zal indien mogelijk gebeuren door static testing (systematisch doorlopen van de code) en daarnaast door functionele tests (programma uitvoeren en gedrag bekijken). Vervolgens laat hij de functionaliteit door tenminste één ander persoon testen.

Indien we vermoeden dat een nieuwe functionaliteit invloed kan hebben op een andere functionaliteit, zullen beide getest worden na het schrijven van de nieuwe functionaliteit, zowel door de auteur als één andere persoon.
Ten slotte zullen alle functionaliteiten iedere fase eenmaal getest worden om te zorgen dat deze ondanks wijzigingen in de code correct blijven werken.

6.2 Scenario-tests

Natuurlijk zullen alle scenario's uitgebreid worden getest, met als doel achter alle fouten in de implementatie en ongeïmplementeerde zaken te komen. De manier waarop dit gebeurt zal hieronder worden beschreven.

Scenario 1 - Flag Taken

Om dit scenario te testen zal één van de ontwikkelaars zich aansluiten bij het vijandige team. Deze persoon zal de vlag proberen te pakken en zich dan ergens op de map verstoppen. Vervolgens zal hij wachten totdat hij door het agent team wordt uitgeschakeld en de laat vallen.

Scenario 2 - Enemy Flag Taken

Bij het testen van dit scenario zal er op een drie punten worden gelet. Ten eerste of de agent met de vlag direct de eigen basis als nieuwe bestemming heeft. Ten tweede of de verdedigers hun taken behouden en onze vlag in de vlagbasis beschermen. Ten slotte of de overige agents de agent met de vlag proberen te beschermen, door bijvoorbeeld de tegenstanders die op hem jagen, te blokkeren.

Scenario 3 - Health Low

Dit scenario kan getest worden door ervoor te zorgen dat de health onder de grens van 50% valt. Dit zou bijvoorbeeld gedaan kunnen worden door de health van te voren in te stellen op een getal onder de 50%. Vervolgens wordt gekeken of de agent ook daadwerkelijk op zoek gaat naar health. Tegelijkertijd zal in de gaten worden gehouden of een andere agent de taak van de vorige overneemt. Ten slotte zal gecontroleerd worden of de agent stopt met zoeken naar health als hij 100% heeft en hij zijn oude taak zal hervatten.

Scenario 4 - Flag Dropped

Het scenario waarbij de eigen vlag op de grond is gevallen kan op de volgende manier worden getest.

Als eerste wordt agent-team geladen in de UT-omgeving. Vervolgens zal één van de ontwikkelaars zich weer bij het vijandige team aansluiten. Deze persoon zal opzoek gaan naar de vlag en zal zich vervolgens laten afschieten, zodat hij de vlag laat vallen. Zodra dit gebeurt zal er worden gekeken of de agents hun vlag terug zullen bregen naar de thuisbasis

Scenario 5 - Enemy Flag Dropped

Als de vlag van de tegenstander op de grond is gevallen, moet een bot de vlag oppakken. Vervolgens moet de vlagdrager handmatig worden gerespawnd en zal er worden gecontroleerd of één van de defenders de rol van vlagdrager overneemt en een tweede teamlid de rol van defender overneemt.

Scenario 6 - Stuck

Om dit scenario te testen zal een willekeurige bot gevolgd worden totdat hij ergens vast komt te zitten, zodra dit gebeurd, hoort de bot vanzelf te respawnen. Vervolgens zal worden gekeken of de bot zijn vorige taak hervat.

Scenario 7 - (Re)Spawn

Dit scenario kan worden getest door een willekeurige bot te laten respawnen. Vervolgens zal worden gekeken of de basisstatistieken gereset worden en of de bot op zoek gaat naar wapens en ammunitie.

Scenario 8 - Enemy Spotted

Om de te testen zal één van de ontwikkelaars zich aansluiten bij het vijandige team. Vervolgens zal hij op zoek gaan naar een van de bots, en zal gekeken worden in de console, of de bot de tegenstander ook daadwerkelijk spot en hij vervolgens op hem zal schieten.

Scenario 9 - Armor Low

Dit scenario kan worden getest door een willekeurige bot te volgen en te kijken of deze op zoek gaat naar armor totdat hij het maximale niveau heeft bereikt

Scenario 10 - Adrenaline Full

Dit scenario kan worden getest, door een bot lang genoeg te volgen totdat hij de maximale waarde van adrenaline heeft bereikt. Als dit het geval is hoort de bot een willekeurige combo uit te voeren.

6.3 Testresultaten

Scenario 1 - Flag Taken

Bij het testen van Flag Taken voegde één van de leden van ons team zich bij het vijandige team (het blauwe team in dit geval) en werd de vlag van de bots (het rode team) gepakt, vervolgens werd er geprobeerd om naar de blauwe basis te lopen. Terwijl dit gebeurde werd de gebruiker succesvol gestopt door het rode team van bots en de vlag vervolgens weer terug gebracht naar de rode basis. De test is hiermee succesvol verlopen.

Scenario 2 - Enemy Flag Taken

Bij het testen van Enemy Flag Taken werd het programma gedraaid totdat de vijandige vlag werd gepakt en deze terug werd gebracht naar de thuisbasis. Tijdens de test werd één bot erop uit gestuurd om de vlag te halen en nadat hij de vlag in handen had, bracht hij deze ook terug naar de thuisbasis. Terwijl hij dit deed was het de bedoeling dat de beschermers van de thuisbasis op hun post zouden blijven en de vlaggendrager vergezeld zou worden door een ander teamlid dat als taak had om de vlaggendrager te beschermen. Dit was echter niet het geval, de andere bots namen niet hun beschreven taken aan, in plaats daarvan bleven zij hangen op de basis van de tegenstander en ging de vlaggendrager alleen, zonder ondersteunend teamlid terug naar de thuisbasis.

Scenario 3 - Health Low

Bij het testen van Health Low speelde één van de leden van ons team weer de rol van tegenstander. Vervolgens werd er op een bot geschoten totdat deze minder dan 50% levenspunten had. Hierna werd gekeken of de beschoten bot ook daadwerkelijk op zoek ging naar meer Health packs totdat deze weer het volle aantal levenspunten had.
Dit was ook het geval, alleen werd er tijdens dit proces ook naar wapens gezocht en had het zoeken naar Health packs niet altijd de prioriteit. In de gevallen waar de bot zowel bij een Health pack als een wapen stond, kreeg de Health pack niet altijd de voorkeur en werd er schijnbaar willekeurig een keuze gemaakt tussen beide items. De bot bleef echter wel zoeken naar Health packs totdat hij weer het volle aantal levenspunten had en hervatte vervolgens weer zijn oude taak.

Scenario 4 - Flag Dropped

Bij het testen van het flag-dropped scenario, is de vlag ergens achtergelaten, en is gekeken of de bot de vlag oppakte, dit was inderdaad het geval. Echter, de bot liep als de vlag ergens gevallen was, gewoon naar de basis van de tegenstander om de vlag te zoeken, en zocht deze niet systematisch zelf naar de vlag. Pas toen de vlag in het zicht was liep de bot ernaartoe om hem terug te brengen.

Scenario 5 - Enemy Flag Dropped

Bij het testen van het enemy-flag-dropped scenario, is gekeken of bots die de vlag in het zicht hadden deze oppakten, en of er systematisch door aangewezen bots naar de vlag werd gezocht. Het eerste was wel het geval: bots die de vlag zagen liggen pakten hem op en brachten hem naar hun eigen basis. Het tweede echter niet, de bots sloegen niet op waar de vlag was achtergelaten en zochten niet meer naar deze plek.

Scenario 6 - Stuck

Bij het testen van dit scenario was het de bedoeling dat de bot zou respawnen als hij vast kwam te zitten. Dit is ook daadwerkelijk gebeurd, en er zijn verder geen opmerkingen.

Scenario 7 - (Re)Spawn

Dit scenario werd tegelijkertijd met het vorige scenario getest. De bedoeling was dat alle statistieken behalve adrenaline werden gereset na het respawen en de bot weer verder ging met z’n vorige taak. Deze test was ook succesvol verlopen, zonder verdere aanmerkingen

Scenario 8 - Enemy Spotted

Bij het testen van dit scenario was het de bedoeling dat een bot zou schieten op de tegenstander als hij deze te zien kreeg. Dit is bij het testen ook gelukt, alleen bleef de bot wel zijn eigen traject volgen en ging hij niet achter de tegenstander aan zodra hij hem te zien kreeg en op hem begon te schieten.

Scenario 9 - Armor Low

Alle bots gaan opzoek naar armor als het niveau van armor laag is, dus deze test is ook succesvol afgenomen.

Scenario 10 - Adrenaline Full

Nadat een bot 100 adrenaline heeft bereikt wordt er ook daadwerkelijk een combo uitgevoerd, alleen is er nog niet gespecificeerd welke dat was. Dus wordt er nu een willekeurig combo uitgevoerd.

7 Conclusie

Tijdens het project hebben we ons bezig gehouden met het ontwikkelen van de meest effectieve strategie om met een team van vier agents een capture the flag wedstrijd te spelen. Na velen strategieën en botconfiguraties te hebben getest, hebben we gekozen om de bots door middel van een hiërarchie samen te laten werken. Een manager agent, die zelf niet aanwezig is in de omgeving, stuurt de andere agents aan en houdt het overzicht. Hij richt zich voornamelijk op het maken van berekeningen en krijgt veel informatie binnen van de agents over de omgeving. Op basis hiervan maakt deze manager agent d.m.v. logica verstandige keuzes die ervoor zorgen dat de bots zo effectief mogelijk het spel spelen.

De ontwikkeling van de agents verliep over het algemeen soepeltjes. We zijn echter ook tegen een aantal problemen aan gelopen. Zo kwam het vaak voor dat door foutjes in de GOAL-omgeving, testresultaten er niet uitzagen zoals we hadden gehoopt. Dit zorgde voor veel verwarring, omdat we in eerste instantie dachten dat de fout bij ons lag. Na uitvoerig testen en overleg met de studentassistenten bleek echter dat de fout in de GOAL-omgeving zat. Dit is later echter opgelost in een update van de GOAl-omgeving.

Verder is het zo dat doordat GOAL en de GOAL-omgeving erg traag zijn, we een aantal ideeën niet hebben kunnen implementeren. Zo kost het berekenen van paden in GOAL erg veel tijd. Tegen de tijd dat de paden uiteindelijk zijn berekend, om bijvoorbeeld het kortste pad te zoeken, was deze informatie niet meer relevant. De agent bevond zich dan al op een andere plaats waardoor eigenlijk de berekening opnieuw zou moeten gebeuren.

Een laatste probleem waar wij tegenaan liepen is het oppakken van items. Dit leek erg interessant maar na ook dit getest te hebben, kwamen wij erachter dat de bots niet in staat zijn om in een vloeiende beweging items op te pakken. Dit doordat er een expliciete instructie moet worden gegeven om naar een bepaalde item toe te lopen. Op deze manier moet de agent na het oppakken van een item, beslissen wat zijn volgende actie zal worden. Dit beslissen kost (te) veel tijd, waardoor het is gebleken dat het vaak efficiënter is om minder waarde te hechten aan het oppakken van items.
Al met al is het project geslaagd, er is veel geleerd over het functioneel programmeren en de samenwerking tussen agents. Ons team is er toe in staat om een team van vier epic bots te verslaan en voldoet dus aan de vereisten van de opdracht.

Bijlage B - Urenverantwoording

Alexander Felix
Bedenken en schrijven/ verbeteren van scenario's - 3 uur
Bedenken hoofdpunten programma van eisen - 0.5 uur
Verbeteren spelling/grammatica en opmaak - 6 uur
Analyse UT omgeving, scenario's schrijven /verbeteren - 8 uur
MAS testen schrijven - 1 uur
Module flag defender schrijven - 9 uur
Schrijven voorwoord en samenvatting - 3 uur

Robbert van Staveren
Bedenken hoofdpunten programma van eisen - 0.5 uur
Bedenken en schrijven verbeteren van scenario's - 3 uur
Algemene afspraken testen maken en uitwerken - 2 uur
Scenario tests - 3 uur
Analyse UT omgeving schrijven en verbeteren - 4 uur
Module defender/flag capturer schrijven - 8 uur
Schrijven van hoofdstuk 5 - 3 uur

Yorick van Pelt
Uitwerken en verbeteren van programma van eisen - 2 uur
Verbeteren code - 10 uur
Ontologie definitie verbeteren - 1 uur
Samenwerkingscontract schrijven - 1 uur
MAS testen schrijven - 1 uur
GOAL syntax highlighting - 5 uur
Code main schrijven - 4 uur

Robert Carosi
Verbeteren code - 10 uur
Tickets maken in trac - 0.5 uur
Samenwerkingscontract schrijven - 1 uur
Samenwerkingscontract nakijken - 1 uur
Attacker module maken - 7 uur
Helpen met de manager module - 2 uur
Schrijven van de inleiding en conclusie - 3 uur

Jeroen Offerijns
Verbeteren code - 6 uur
Ontologie definitie - 2 uur
Samenwerkingscontract doorlopen - 1 uur

Ontwerpverslag - 4 uur
Manager module maken - 9 uur
Productverslag verbeteren - 1 uur
Schrijven van hoofdstuk 5 - 3 uur

Bijlage C - Begrippenlijst

Agent	Een klein computerprogramma dat in opdracht van een gebruiker of van een logische redenering bepaalde taken uitvoert.
BW4T	Blocks World For Teams, een omgeving waar door middel van de GOAL programmeertaal bepaalde taken kunnen worden uitgevoerd.
Functionele tests	Testen door middel van uitvoering van het programma observatie van het gedrag.
GOAL	Een logische programmeertaal voor het maken van programma's die inspelen op de omgeving waarin deze worden uitgevoerd en zo beslissingen maken.
Realtime	Realtime betekent zonder wachttijden als gevolg van verwerken van gegevens. In deze context wordt ermee bedoeld dat de agents kunnen inspelen op de omgeving zonder dat hier (significante) tijd tussen zit.
Static testing	Testen door middel van systematisch doorlopen van de code.
UT-GOAL	De omgeving tussen Unreal Tournament en GOAL. Met behulp van deze omgeving kunnen onze GOAL programma's communiceren en inspelen op de UT omgeving.

Literatuurlijst

Bij het schrijven van dit verslag zijn een aantal externe bronnen gebruikt, deze worden hieronder opgesomd.

- Website: http://www.projectsmart.co.uk/moscow-method.html MoSCoW Method, geraadpleegd 12-05-2013.
- Externe Bron: Project Multi-Agent Systemen Projecthandleiding 2012/2013, door Koen Hindriks paragraaf 4.2.2 (Deliverables Fase 2: Implementatie van Basisfunctionaliteit).