De virtuele verhalenverteller:

Agent-gebaseerde generatie van interessante plots

Door S.H.J. Rensen

Astudeer Commissie:

- dr. M. Theune
- prof. dr. ir. A. Nijholt
- dr. ir. H.J.A. op den Akker
- dr. D.K.J. Heylen

Maart 2004, Universiteit Twente, Enschede

Samenvatting

De Virtuele Verhalenverteller is een agent gebaseerde architectuur voor de generatie van verhalen. De architectuur bestaat uit vier soorten agents: Regisseur, Acteur, Verteller en Presentator. De Regisseur agent is verantwoordelijk voor de verhaalstructuur. De Acteurs vertegenwoordigen de karakters die in een virtuele wereld het verhaal beleven. De verteller krijgt informatie van de regisseur over de belevingen van de Acteurs in de virtuele wereld en is verantwoordelijk voor het omzetten van deze informatie naar natuurlijke tekst. De Presentator zet de tekst om in spraak en is tevens zichtbaar op het scherm als een geanimeerd figuurtje.

Dit onderzoek gaat verder met het werk dat gedaan is door S. Faas [Fa02]. De nadruk in dit onderzoek wordt gelegd op genereren van interessante plots. Er wordt een architectuur gepresenteerd, waarin geloofwaardige karakters met emoties de basis vormen voor een goed verhaal. De geloofwaardige karakters worden vertegenwoordigd door Acteurs, die leven in een virtuele wereld. Doordat de Regisseur de karakters binnen een van tevoren gedefinieerde structuur, bestaande uit episodes, laat blijven, ontstaat er verhalen met een interessante plot.

De huidige versie van de Virtuele Verhalenverteller is in staat om korte verhalen te generen met simpele plots. De Virtuele Verhalenverteller biedt de gebruiker de mogelijkheid om een structuur voor een verhaal te definiëren en karakters aan te passen naar eigen inzicht.

De Virtuele Verhalenverteller kan nog lang niet als af worden beschouwd. Er zijn nog vele mogelijkheden om de Virtuele Verhalenverteller te verbeteren, maar we kunnen deze versie beschouwen als een grote vooruitgang.

Abstract

The Virtual Storyteller is an agent based system for computational story telling. Its architecture contains four types of agents: Director, Actor, Narrator and Presenter. The Director agent is responsible for the story structure. The Actors represent the characters, which live in a virtual world. A Narrator agent receives information from the Director about the adventures of the Actors in order to generate a story in natural language text. The Presentation agent transforms the story text into speech and shows up on the screen as an animated character.

This thesis continues on the work of S.Faas [Fa02]. This thesis focuses on the generation of interesting plots. An architecture which takes care of a good story, with believable and emotional charachters is presented. The believable characters are represented by Actor agents who live in a virtual world. Because the Director keeps the Actors to a predefined structure, which consists of episodes, the Virtual Storyteller is capable of generating stories with an interesting plot.

The current version of the Virtual Storyteller can tell short stories with simple plots. The Virtual Storyteller offers the user the opportunity to define a structure for stories and to alter characters to his own opinion.

The Virtual Storyteller isn't finished yet. There are still many opportunities to improve the Virtual Storyteller, but this version can be seen as a great improvement.

Voorwoord

Na vier en half jaar de opleiding Informatica te hebben doorlopen was ik toe aan mijn afstudeeropdracht. Als allereerst moest er een onderwerp worden gekozen. Na met vele mensen gesproken te hebben attendeerde Anton Nijholt mij op het project “The Virtual Storyteller”. Sander Faas had al een architectuur ontwikkeld voor een Virtuele Verhalenverteller, maar er was nog ruimte genoeg voor verder onderzoek. Indien ik interesse had, moest ik voor informatie naar “Mariët”. Nou, interesse had ik zeker. Zelf ben ik een verwoede lezer. Soms lees ik drie dikke pillen in een week en ik vroeg mij af of een computerprogramma in staat zou zijn om een mooi verhaal te maken.

“Mariët” bleek Mariët Theune te zijn. Zij had Sander Faas begeleid en wou ook mij wel begeleiden. Haar enthousiasme sloeg op mij over en samen met mijn al aanwezige interesse maakte dat ik de keuze snel gemaakt had. Voor mijn afstudeeropdracht zou ik mij bezig houden met het ontwikkelen van een programma dat verhalen genereert.

Het ontwikkelen van een programma dat verhalen genereert bleek al met al nog niet zo gemakkelijk te zijn. Er waren al wel programma’s gemaakt die ongeveer hetzelfde deden als wat ik wilde, maar er kwamen vele facetten bij kijken. Uiteindelijk besloot ik me toe te leggen op het genereren van interessante plots met behulp van autonome agents.

Nu, aan het einde van mijn afstudeeropdracht, ligt er een verslag en niet minder belangrijk een werkend programma dat in staat is om korte verhalen met simpele plots te genereren. Het gaf een fijn gevoel om te beginnen aan een opdracht die je interesseert, maar het geeft toch minstens een zo’n fijn gevoel als je de opdracht af kunt sluiten met het idee dat je echt iets gepresteerd hebt.

Dit presteren heb ik natuurlijk niet alleen gedaan. Als eerste gaat mijn dank uit naar Mariët Theune. Zij heeft mij gedurende het hele project intensief begeleid en was altijd bereid om een gesprek met mij aan te gaan om aanwijzingen te geven. Ook toonde ze veel begrip als ik een weekje iets minder had gedaan. Vervolgens wil ik de rest van mijn afstudeercommissie bedanken. Rieks voor de algemene aanwijzingen bij zowel het opzetten van de architectuur als het daadwerkelijk implementeren van het programma. Dirk en Anton wil ik bedanken voor de vele relevante vragen, de opbouwende kritiek en de aanwijzingen. Ook wil ik Hendri Hondorp bedanken voor enkele technische ondersteuningen.

Als laatste en niet in het minst gaat mijn dank uit naar de personen die een belangrijke rol in mijn leven spelen. Mijn vrienden, mijn familie en natuurlijk mijn vriendin. Zij bleven vragen hoe het met de vorderingen betreffende mijn afstuderen stond en ze ondersteunden mij daar waar nodig was.

Enschede/Luttenberg, Maart 2004

Sander Rensen

Inhoudsopgave

1. Inleiding

"The Great Automatic Grammatizator" is een verhaal van Roald Dahl, waarin wordt verteld over een machine die in staat is om aan de hand van een paar instellingen complete boeken te schrijven. De boeken worden door vele mensen mooi gevonden en verkopen dan ook goed. De machine publiceert zelfs onder verschillende pseudoniemen om niet op te laten vallen, dat qua hoeveelheid bedachte verhalen het niet om een menselijke auteur zou kunnen gaan.

Zou het daadwerkelijk mogelijk zijn om een computer een verhaal te laten bedenken waarmee een compleet boek te vullen is?

Dit verslag, zal geen antwoord kunnen geven op deze vraag. Wel wordt er een eerste opzet gegeven in de ontwikkeling van een computerprogramma, dat in staat is om verhalen te bedenken waarmee een compleet boek te vullen is.

Dit verslag beschrijft de verdere ontwikkeling van de architectuur die gepresenteerd is door S. Faas [Faa02]. In dit verslag wordt de nadruk gelegd op het genereren van interessante plots met behulp van geloofwaardige en emotionele agents.

Dit verslag is opgebouwd in drie delen:

Het eerste deel van dit verslag beschrijft de literatuur. Er zal gekeken worden welke onderzoeken er al verricht zijn op het gebied van Virtuele Verhalenvertellers. Aan de hand van deze eerder verrichte onderzoeken zullen de eisen worden vastgesteld waaraan de nieuwe architectuur voor de Virtuele Verhalenverteller moet voldoen.
Aan de hand van deze eisen zal er vervolgens in hoofdstuk 3 en 4 gekeken worden naar het onderzoek dat gedaan is op het gebied van de structuur van verhalen en geloofwaardige karakters.

In het tweede deel geven we antwoord op de vraag hoe volgens ons, autonome agents zich binnen een voorgeschreven verhaalstructuur zouden kunnen houden. Ook beantwoorden we de vraag hoe volgens ons hoe autonome agents geloofwaardige karakters kunnen spelen in een verhaal. Als laatste presenteren we een architectuur voor de Virtuele Verhalenverteller. Binnen deze architectuur.

In het derde deel beschrijven we het prototype dat we hebben gemaakt aan de hand van de in deel twee gepresenteerde architectuur. We beschrijven welke keuzes we hebben moeten maken bij het implementeren van het prototype en waar we voor de implementatie afwijken van de architectuur.

In het vierde en laatste deel is maken we plaats voor discussie en bespreken we de sterke en de zwakke punten van zowel onze architectuur als onze implementatie.

Onderdeel 1: Literatuuronderzoek

2. Eerdere werken

Er zijn in de tijd, al verscheidene pogingen gedaan om een programma te schrijven dat in staat is om verhalen te maken en te vertellen. In dit hoofdstuk nemen we enkele van die pogingen eens nader onder de loep. We richtten ons hierbij vooral op systemen die tekst als uitvoer produceren.

2.1 Tale-Spin (1976)

James Meehan [Mee81] maakt in zijn Tale-Spin gebruik van een simulatie van een kleine virtuele wereld met karakters (zie figuur 2.1). De karakters hebben doelen en maken plannen en acties om de doelen te behalen. Meehan maakt hierbij gebruik van de Conceptual Dependency Theory [Sch77].

Belangrijk is, dat karakters in Tale-Spin in dertien verschillende emotionele staten kunnen verkeren. Afhankelijk van de emotionele status waarin ze verkeren, voeren de karakters verschillende acties uit.

Voorbeeld:
Als John Jim aardig vindt, zal hij hem niet aanvallen, maar als John Jim niet aardig vindt, dan zal hij dit wel doen.

Een ander belangrijk punt van Tale-Spin is de mogelijkheid voor karakters om te liegen. Karakters kunnen andere karakters laten geloven dat feiten die eigenlijk niet waar zijn, waar zijn. Op deze manier is het mogelijk voor karakters om elkaar voor te liegen. Dit kan leiden tot interessante ontwikkelingen.

Als laatste is de interactie met de gebruiker een belangrijk punt om te noemen (zie figuur 2.1). De gebruiker moet tijdens het verhaal bepaalde beslissingen nemen, waardoor het verhaal verschillende uitkomsten heeft. De interactie geeft het verhaal een extra dimensie. De gebruiker gaat zich afvragen wat er zal gebeuren als hij een beslissing neemt.

Tale-Spin is een zeer behoorlijk programma met leuke karakters, maar er zijn toch wel enkele tekortkomingen:

Zo zegt Turner[tur92] dat het probleem van Tale-Spin is, dat de verhalen niet interessant zijn. Ze zijn “pointless”. Tale-Spin houdt volgens Turner op geen enkele manier binnen een verhaal rekening met structuur boven het niveau van karakter planning. Bij karakter planning voeren de karakters domweg acties uit, zonder dat er gekeken wordt naar een overkoepelende verhaalstructuur.

Mark Lee [Lee94] claimt dat karakters in Tale-Spin altijd het eerste plan gebruiken dat ze tegenkomen. Ook heeft het programma er geen idee van wat een verhaal leuk maakt om te vertellen. Als laatste zegt Lee, dat de structuur van de verhalen van Tale-Spin slecht is. Er is geen goed einde. Het verhaal eindigt simpelweg, wanneer de hoofdpersoon zijn doel bereikt. Dit kan eindeloos doorgaan, maar het kan ook gebeuren dat het verhaal eindigt terwijl een subplot nog volledig in ontwikkeling is.

Callaway [Cal00] zegt dat Tale-spin geen hoogwaardige en meerdere pagina's bestrijkende verhalen kan maken. Volgens Callaway mist Tale-Spin de karakter-karakter dialogen, is het niet mogelijk om te veranderen van oogpunt en zijn de zinnen van beperkte complexiteit. Verder zegt hij het volgende:
"Talespin has no sophisticated linguistic or emotional knowledge about stories and what is needed to successfully create and narrate stories."

Sander Faas [Fa02] zegt dat vooral de beperkte zinkwaliteit en de slechte structuur van de verhalen het grote probleem vormen van Tale-spin.

```
******** WELCOME TO TALE-SPIN ********
CHOOSE ANY OF THE FOLLOWING CHARACTERS FOR THE STORY:
1: BEAR
2: BEE
3: BOY
4: GIRL
5: FOX
6: CROW
7: ANT
8: CANARY
* 1 2

ONCE UPON A TIME SAM BEAR LIVED IN A CAVE. SAM KNEW
THAT SAM WAS IN HIS CAVE. THERE WAS A BEEHIVE IN AN
APPLE TREE. BETTY BEE KNEW THAT THE BEEHIVE WAS IN
THE APPLE TREE. BETTY WAS IN HER BEEHIVE. BETTY KNEW
THAT BETTY WAS IN HER BEEHIVE. THERE WAS SOME HONEY
IN BETTY'S BEEHIVE. BETTY KNEW THAT THE HONEY WAS IN
BETTY'S BEEHIVE. BETTY HAD THE HONEY. BETTY KNEW THAT
BETTY HAD THE HONEY.

- DECIDE: DOES BETTY BEE KNOW WHERE SAM BEAR IS?    *
NO
- DECIDE: DOES SAM BEAR KNOW WHERE BETTY BEE IS?    *
YES
```

Figuur 2.1 Tale-Spin

Zelf zijn we van mening, dat Tale-Spin vrij aardige verhaaltjes maakt. We vinden de interactie met de gebruiker het sterkste punt, maar zijn het eens met Turner dat het gebrek aan een goede verhaalstructuur het probleem is van Tale-Spin.

2.2 Universe (1985)

Michael Lebowitz heeft met Universe een programma gecreëerd, dat plots genereert voor een oneindige soap. Universe maakt gebruik van zogenaamde persoonsframes om karakters te specificeren. Elk karakter krijgt hierbij een nummer van 0 tot 10 voor eigenschappen als aardigheid, gezondheid en intelligentie.
In Universe is er een soort van auteur aanwezig die het verhaal plant. Er wordt hierbij gebruik gemaakt van zogenaamde plotfragmenten. Een voorbeeld van zo'n plotfragment is te zien in het figuur 2.2.

Volgens Turner[tur92] is het gebruik van doelen op auteurniveau zeer bruikbaar. Waarmee hij met doelen op auteurniveau bedoelt, dat er gekeken wordt naar de complete structuur van het verhaal zoals een auteur dat zou doen. Volgens Turner zal, doordat de planner gebruik maakt van plot fragmenten, er echter nooit erg veel diepte in het verhaal zijn.

```
Plot fragment – forced-marriage
Characters – ?him, ?her ?husband ?parent
Constraints – (has-husband ?her) (the husband character)
      (has-parent ?husband) (the parent character)
      (less-than (trait-value ?parent niceness) 5)
      (female-adult ?her)
      (male-adult ?him)
Doelen – (churn ?him ?her) (prevent them from being happy)
Subdoelen – (do-threaten ?parent ?her 'forget-it'') (threaten
      ?her)
      (dump-lover ?her ?him) (have ?her dump ?him)
      (worry-about ?him) (get ?him involved with somebody
      else)
      (eliminate ?parent) (get rid of ?parent (breaking
      threat))
      (do-divorce ?husband ?her) (end the unhappy marriage)
      (or (churn ?him ?her) (either keep churning or)
      (together ?her ?him) (try and get ?her and ?him back
      together)
```

Figuur 2.2 Een plotfragment uit Universe

Lebowitz zelf [Lee94] zegt dat de plot fragmenten inderdaad de prestaties van Universe erg beïnvloeden. Er moeten genoeg plot fragmenten aanwezig zijn voor een goed verhaal. Uiteindelijk moet Universe volgens Lebowitz zelf plot fragmenten kunnen creëren om niet te veel in herhaling te vallen.

Sander Faas[Fa02] zegt over Universe dat er geen echte structuur in de verhalen valt aan te wijzen. Dit komt doordat het verhaal oneindig is en er dus geen echt einde hoeft te worden gemaakt.

Universe is inderdaad, zoals ook Callaway [Cal00] beschrijft, geen echte verhalenverteller. Universe is alleen in staat om plot fragmenten te maken en beschikt dus niet over informatie over het maken van goede natuurlijke taal.

2.3 Minstrel (1994)

In 1994 ontwikkelde Scott Turner Minstrel. Minstrel [tur92] vertelt verhalen over Koning Arthur en de ronde tafel. In Minstrel hebben de acteurs doelen die ze proberen te halen. Het grote verschil met Tale-Spin is hierbij dat de auteur ook doelen heeft.

In het voorbeeld (figuur 2.3) was het doel van de auteur om een verhaal te maken met het volgende thema (Pat genoemd): Good-Deeds-Rewarded story.

```
THE KNIGHT AND THE HERMIT

ONCE UPON A TIME, THERE WAS A HERMIT NAMED BEBE
AND A KNIGHT NAMED CEDRIC. ONE DAY, CEDRIC WAS
WOUNDED WHEN HE KILLED A DRAGON IN ORDER TO
IMPRESS THE KING. BEBE, WHO WAS IN THE WOODS
PICKING BERRIES, HEALED CEDRIC. CEDRIC WAS
GRATEFUL AND VOWED TO RETURN THE FAVOR.

LATER, BEBE BELIEVED HE WOULD DIE BECAUSE HE SAW
A DRAGON MOVING TOWARDS HIM AND BELIEVED THAT IT
WOULD EAT HIM. BEBE TRIED TO RUN AWAY BUT
FAILED. CEDREC, WHO WAS IN THE WOODS, KILLED A
DRAGON AND SAVED BEBE.

MORAL: "A FAVOR EARNED IS A FAVOR RETURNED"
```

Figuur 2.3 Minstrel

Minstrel maakt gebruik van een model, waarbij de auteur als probleemoplosser wordt gezien. De auteur lost een probleem op door te kijken naar eenzelfde problemen uit het verleden. De oplossing van het verleden wordt vervolgens gebruikt voor de oplossing van nu. Turner noemt het gebruik van oplossingen uit het verleden: “gebruik maken van Episodic Memory”.

Minstrel gebruikt zogenaamde Transform Recall Adept methods (TRAMs) om de oplossingen uit het verleden toe te passen. In het onderstaande voorbeeld wordt de werking van zon’n TRAM nog eens verduidelijkt.

Voorbeeld:
Oude situatie: Een hond rent weg. De man zoekt zijn hond en tijdens het zoeken vindt hij een meid. De man vindt dus de meid doordat de hond wegliep.

Huidige situatie: Een ridder moet een prinses vinden
Oplossing: De auteur maakt een paard aan dat wegloopt met de ridder in een bos. In het bos ontmoeten ze een prinses. De ridder ontmoet dus de prinses doordat het paard wegliep.

Lee [Lee94] zegt over Minstrel dat de verhalen een punt hebben. In tegenstelling tot Universe en Tale-Spin wordt geprobeerd om in elk verhaal een boodschap of een moraal te verwerken. Ook hebben de verhalen een goede structuur. Volgens Lee is het gevaarlijk dat elk thema of Pat identiek is in structuur. Mochten er dus meerdere Pat's gebruikt worden in een verhaal, dan is de kans groot dat het verhaal vrij simpel en vol herhalingen wordt. Een ander nadeel, aldus Lee, is dat het erg moeilijk is om op deze manier veel duidelijk verschillende verhalen te creëren. Als laatste nadeel noemt Lee, dat de taalgeneratie gebaseerd is op zinsdelen en niet op woorden. Op deze manier worden zinnen vaak op dezelfde manier verteld en kan het verhaal snel saai worden.

Sander Faas[Fa02] noemt als nadeel dat de taal die gegenereerd wordt behoorlijk mechanisch is. Als grootste nadeel noemt Faas echter het gebrek aan variatie. Door het gebruik van TRAMs zullen alle nieuwe gebeurtenissen dicht bij eerdere (al bekende) gebeurtenissen liggen.

We zijn het eens met Lee dat de verhalen een goede structuur hebben en over een thema gaan en delen ook de mening van Faas dat het gebruik van TRAMs gevaar oplevert voor de variatie. Oplossingen zullen dicht bij oplossingen uit het verleden liggen en daardoor zullen de verhalen veel op elkaar gaan lijken.

2.4 Joseph (1997)

Joseph is een programma gemaakt door Raymond Lang [Lan94]. In Joseph hebben karakters doelen en emoties. Volgens Lang is Joseph het eerste systeem dat gebruik maakt van een expliciet model voor verhalen. De grammatica van Rumelhart [Rum75] diende bij Lang als basis voor de grammtica die hij gebruikt in Joseph.

```
story( story(Setting, Ep(list) ) --->
      setting(Setting, S_int),
      episodes(Ep(list, E_int),
      {meets(S_int, E_int)}.
```

Figuur 2.4 Eerste regel uit de grammatica, gebruikt in Joseph

In figuur 2.4 is een regel te zien uit de grammatica die Lang gebruikte. In zijn grammatica maakt Lang gebruik van tijdsintervallen (E_int en S_int). Lang beweert, dat om goede verhalen te maken er veranderingen in de omgeving moeten worden gemoduleerd. Hij is dan ook degene geweest, die als eerste de temporele logica introduceerde in programma's voor verhaalgeneratie.

Joseph produceert met behulp van de grammatica een simpel plot dat vervolgens weer wordt omgezet in natuurlijke taal (zie figuur 2.5).

```
one day it happened that peasant quarreled with the
wife. when this happened, peasant felt distress. in
response, peasant took a walk in the woods. peasant
found a pit when he looked under the bush. when this
happened, peasant desired to punish wife. in
response, peasant made it his doel that wife would be
in the pit. peasant tricked wife. wife was in the
pit. peasant lived alone.
```

Figuur 2.5 Een fragment gemaakt door Joseph

Callaway [Cal00] zegt over Joseph, dat het gebruik maakt van simpele templates om daadwerkelijke zinnen te construeren. Dit zorgt niet voor mooie vlotte zinnen. Callaway merkt hierbij wel op dat de focus van het project niet lag op het genereren van de taal. Faas [Fa02] vindt de introductie van temporele logica en een grammatica een zeer belangrijke ontwikkeling.

2.5 The OZ project (1997)

Het OZ project [Bat92-1] is een project met tot doelstelling een programma te maken die het auteurs mogelijk maakt om interactieve toneelstukken te maken en te presenteren. Het is het doel om gebruikers van de interactieve drama's te laten ervaren hoe het is om in een kleine wereld te leven met daarin emotionele agents [Bat92-2].

Interactieve toneelstukken zijn toneelstukken waarin de gebruiker zelf een rol speelt. Oz kent feitelijk twee verschillende versies die een eerste aanzet vormen tot het maken van interactieve toneelstukken. Eén realtime animatie versie en een tekstuele versie. In onderstaand figuur een voorbeeld van het OZ project in tekstversie. Zoals te zien valt is de gebruiker de hoofdpersoon. Dit is bij OZ altijd het geval (in beide versies). OZ lijkt dan ook wel meer op een spel dan op een verhaal, maar interactieve verhalen liggen dan ook erg dicht bij een spel.

```
You are in the dining room.
To the south, you see the sunroom.
To the east, you see the kitchen.
The end table and the small chair are in the dining room.
The jar is on the end table.
The nine black sardines are in the jar.
PLAYER> take the jar
You take the jar.
PLAYER> go south
You are in the sunroom.
Lyotard goes to the sunroom.
PLAYER> give a sardine to lyotard
You offer the black sardine to Lyotard.
Lyotard runs to the dining room.
PLAYER> follow lyotard
You run to the dining room.
Lyotard looks around nervously.
PLAYER> pet lyotard
You pet Lyotard.
Lyotard bites you.
```

Figuur 2.6 The OZ project

In OZ wordt er gebruik gemaakt van een dramamanager. Deze dramamanager bekijkt de wereld en de acties (inclusief de acties van de gebruiker). Zodra de dramamanager constateert dat er een punt (plot point genoemd) is aangebroken in het verhaal waarbij het verhaal een richting uitgestuurd kan worden zal ze proberen het verhaal in de goede richting te sturen. Deze richting is afhankelijk van de globale doelen die de dramamanager vooraf heeft meegekregen.

De vooruitgang die in de verhalen van het OZ project geboekt wordt is toch wel de geloofwaardigheid van de aanwezige karakters. In Oz wordt er gebruik gemaakt van autonome karakters met emoties. De gebruiker kan zich voorstellen dat de karakters uit de verhalen echt bestaan.

2.6 NIMIS: Teatrix (1999)

Teatrix [Pai01] is een project van de NIMIS groep en is gemaakt voor kinderen. Met behulp van het programma kunnen kinderen hun eigen virtuele verhalen maken. De kinderen kunnen de setting en de karakters kiezen. Bovendien is het mogelijk dat ze een of meerdere agents besturen in een multi-user omgeving.

In Teatrix zijn er drie verschillende soorten agents [Nim01]:

- Agents gecontroleerd door kinderen. Deze agents zijn de karakters die voorkomen in elk verhaal. De karakters hebben een bepaalde rol in het verhaal waar hun gedrag en doelen op aansluiten. Hoewel ze dus een bepaalde rol hebben in het verhaal, worden ze wel primair bestuurd door de kinderen.
- Agents gecontroleerd door het systeem. Deze agents zijn hetzelfde als de agents die gecontroleerd worden door de kinderen, alleen zijn deze agents autonoom. Deze agent valt onder de controle komen van de Director agent.

- Director agent. Deze agent speelt alleen een rol op de achtergrond. De agent probeert het verhaal te controleren en ervoor te zorgen dat het vloeiend verloopt.

In Teatrix wordt er gebruik gemaakt van vijf autonome karakters die hun eigen emoties en beliefs hebben. De karakters en zijn acties zijn gebaseerd op Propp[Pro68]. Hun emoties op het OCC model[Ort88]. Door het gebruik van emoties wordt evenals in OZ geprobeerd de karakters geloofwaardig te maken.

Teatrix produceert geen tekst. Er wordt alleen een virtueel verhaal met figuren getoond op het scherm. De architectuur is echter vrij duidelijk en zou gebruikt kunnen worden voor het maken van tekstverhalen.

Een probleem van Teatrix is dat het eind van het verhaal niet een duidelijk punt of climax heeft. Het programma is vooral bedoeld om kinderen te vermaken en het gebruik van de computer te leren. Het is niet de bedoeling om een echt goed verhaal met een goede structuur te maken. Er is een Director, maar zijn invloed is niet erg duidelijk. Hij kan nieuwe objecten en locaties introduceren, maar het lijkt er niet op dat hij daadwerkelijk probeert om een verhaal te maken met een duidelijke climax.

2.7 Storybook (2000)

Storybook [Cal00] is programma gemaakt door Charles Callaway als promotieonderzoek aan de Carolina State University. Het programma vertelt verhalen over roodkapje (zie figuur 2.7).

Callaway[Cal00] zegt zelf dat Storybook het eerste systeem is waar de focus bij de ontwikkeling lag op het maken van verhalende taalgeneratie. Storybook maakt gebruik van het zogenaamde AUTHOR systeem. Het AUTHOR systeem heeft kennis over fabula en suzjet [bal97][seg88] en stilistische factoren. Fabula is de totale kennis over de te vertellen wereld en suzjet is de rangschikking en specificaties van wat de auteur presenteert en op welke positie het thuishoort binnen het vertelde. Ze worden ook wel gezien als synoniemen voor het plot en het daadwerkelijke verhaal [Kau99]. AUTHOR gebruikt deze kennis om paragrafen te maken met goede stijl en dialoog.

```
Once upon a time, there was a woodman and his wife who lived in a
pretty cottage on the borders of a great forest. They had one little
daughter, a sweet child, who was a favourite with everyone. She was
the joy of her mother's heart, and to please her, the good woman made
her a little scarlet cloak and hood. She looked so pretty in it that
everyone called her Little Red Riding Hood. One day her mother told
Little Red Riding Hood she meant to send her to her grandmother to
take her some fresh butter and new-laid eggs and a nice cake. Her
grandmother was a very old lady, who lived in the heart of a
neighbouring wood. She was delighted at being sent on this errand, for
she liked to do kind things. It was a very long time since she had
seen her grandmother, and so she had almost forgotten what the old
lady looked like.
```

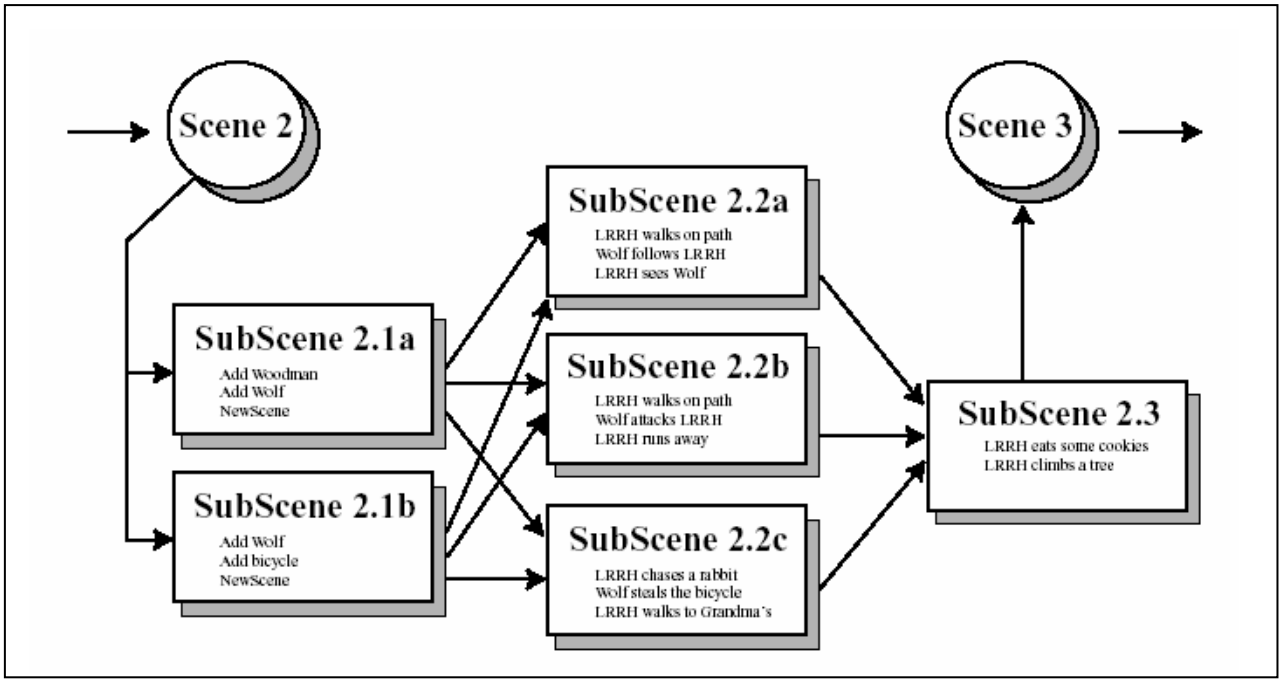
Figuur 2.7 Storybook

Zoals te zien is in onderstaand figuur (Fig 2.8), is Storybook erg goed in staat om een dialoog tussen verschillende karakters weer te geven. Ook kan Storybook wisselen van oogpunt. Dit maakt de verhalen van Storybook fijn om te lezen.

```
So the wolf followed Little Red Riding Hood a little way and came up to
her very gently. He said: “Good day, Little Red Riding Hood, where are
you going?"
“To see my grandmother," said the child. “To take her a present from
mother of eggs and butter and cake."
“Where does your grandmother live?" asked the wolf.
“Quite in the middle of the wood," Little Red Riding Hood replied.
“Oh! I think I know the house. Good-bye, Little Red Riding Hood." And
he ran off as fast as he could go.
```

Figuur 2.8 Een dialoog gecreëerd door Storybook

Een grote tekortkoming van Storybook is echter de generatie van plots. Storybook kan alleen overweg met plotfragmenten van het sprookje roodkapje. Er wordt geen gebruik gemaakt van autonome agents. Er wordt voor de plotgeneratie gebruik gemaakt van scenes met de verschillende sub-scenes (zie figuur 2.9).



Figuur 2.9 plotgeneratie in Storybook

Het gevaar is aanwezig dat de verhalen van Storybook erg in herhaling zullen vallen. Verder zullen de plots nooit erg creatief zijn. De plots zijn namelijk allen voorgeprogrammeerd.

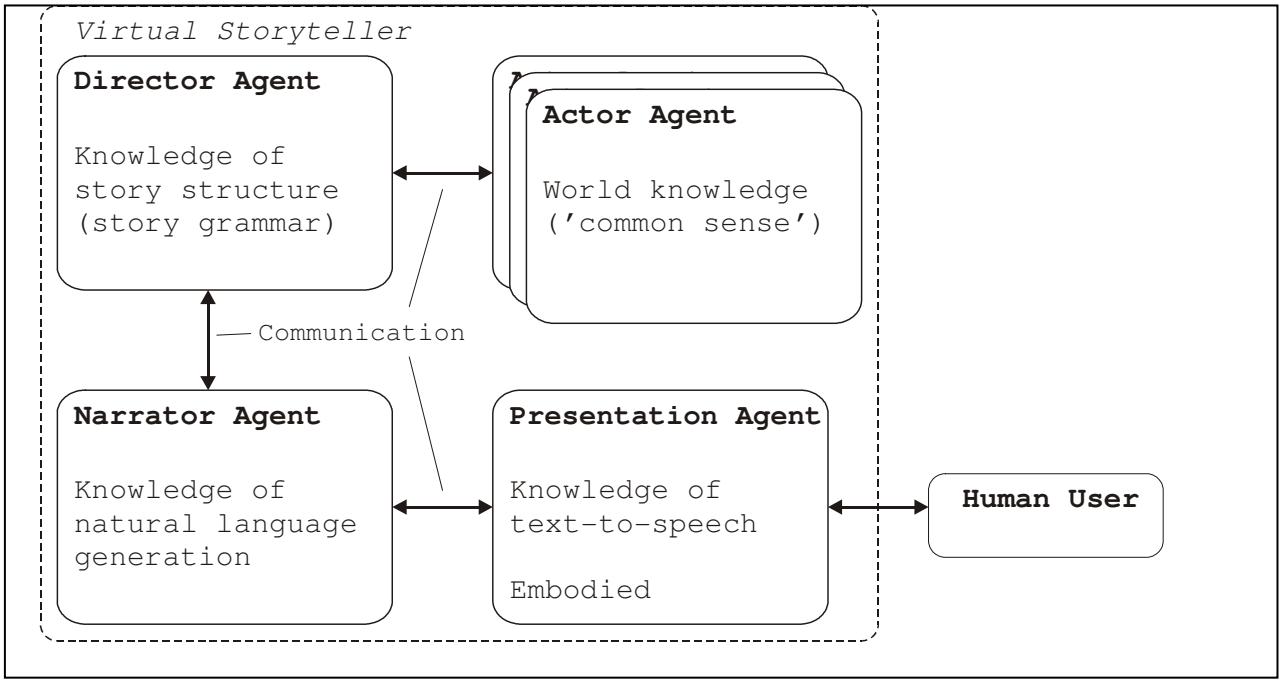
2.8 The Virtual Storyteller (2002)

De virtuele storyteller is een architectuur gemaakt door Sander Faas [Faa02] voor zijn afstudeerproject aan de Universiteit Twente. Zijn doel was om een interactieve verhalenverteller te maken. Dit bleek een te grote opgave en uiteindelijk is er een architectuur voor een verhalenverteller ontstaan.

De architectuur die Sander Faas ontwikkeld heeft is gebaseerd op een structuur, die overgenomen is uit het improvisatietheater, genaamd Typewriter [Joh81]. Bij de Typewriter structuur is er een verteller aanwezig, die niet alleen maar het verhaal vertelt, maar ook de regisseur is. Hij vertelt over de omstandigheden waarin het verhaal zich afspeelt, introduceert acteurs als het nodig is en houdt tevens de structuur van het verhaal in de gaten. Er zijn bij de Typewriter structuur dus drie rollen aan te wijzen: de regisseur (director), de verteller (narrator) en de acteur (actor).

Faas vindt de gedeelde verantwoordelijkheid binnen de structuur het belangrijkste. De verteller/regisseur houdt de structuur van het verhaal in de gaten, terwijl de acteurs in het verhaal hun eigen fantasie gebruiken om het plot gestalte te geven.

De architectuur van de Vritual Storyteller bestaat uit één of meerdere “Actor agents”, een “Director Agent”, een “Narrator Agent en een “Presentation Agent (zie figuur 2.10). De “Director Agent” is verantwoordelijk voor de verhaalstructuur. De “Actor agents” vertegenwoordigen de karakters die in een virtuele wereld avonturen beleven. De “Narrator Agent” krijgt informatie van de “Director Agent” over de belevingen van de “Actor agents” in de virtuele wereld en is verantwoordelijk voor het omzetten van deze informatie naar natuurlijke tekst. De “Presentation Agent” zet de tekst om in spraak en is tevens zichtbaar op het scherm als een geanimeerd figuurtje.



Figuur 2.10 Architectuur Virtual Storyteller

Zoals te zien valt in de onderstaande figuur (figuur 2.11) kan de Virtual Storyteller zeer kleine simpele verhalen maken. Ook is de gebruikte tekst erg mechanisch. De structuur van de verhalen is ook niet erg goed. Net zoals bij Tale-Spin eindigt het verhaal wanneer de hoofdpersoon zijn doel heeft bereikt.

```
Er was een s een dwerg. Hij heette Plop.
Plop was in het bos. Een huis was in het
bos. Een appel was in het huis. Plop was
hongerig. Plop liep naar het huis. Hij ging
het huis binnen. Hij pakte de appel op. Hij
at de appel op. Plop leefde nog lang en
gelukkig
```

Figuur 2.11 Verhaal gemaakt door Virtual Storyteller

Sander Faas heeft een kleine grammatica ontwikkeld (begin-midden-eind) voor de verhalen van de storyteller. De Director-agent moet deze grammatica gebruiken.Op dit moment doet de Director echter niet veel met de grammatica. Hij introduceert de setting en de acteurs als begin en als einde produceert hij een zin dat iedereen nog lang en gelukkig leeft. In het midden van het verhaal onderneemt de Director geen enkele actie. Deze mogelijkheid is echter wel open gelaten in de architectuur.

Een ander probleem is, dat de Director, die bepaalt wat er verteld wordt, geen idee heeft welk deel van de tekst saai is en welk deel interessant is.

Het laatste probleem zijn de karakters. Er is eigenlijk maar een karakter ontwikkeld en deze is zeer beperkt. Zo heeft hij geen emoties en probeert slechts 1 doel te verwezenlijken. Het programma heeft bovendien problemen bij het gebruik van meerdere karakters.

2.9 Conclusies

Na het analyseren van vroegere werken kunnen we vier dingen concluderen, die betrekking hebben tot het maken van de nieuwe Virtuele Verhalenverteller.

- De verhalen die de verhalenverteller zal gaan maken moeten een structuur bevatten. Binnen de structuur moet echter wel plaats zijn voor creativiteit. Met creativiteit bedoelen we dat er binnen één en dezelfde structuur veel verschillende en gevarieerde verhalen gemaakt kunnen worden. Bovendien zorgt een goede structuur voor een plot.
- De verhalen hebben goede karakters nodig. Karakters moeten geloofwaardig zijn, zodat mensen zich kunnen inleven in de personages. Er zullen autonome karakters zoals gebruikt bij het Oz project, Teatrix en in mindere mate Tale-Spin gebruikt moeten worden.
- De verhalen moeten interessant zijn en binnen een verhaal moet één centraal thema gevolgd worden. Minstrel kan hier als voorbeeld dienen.
- De verhalen moeten goed leesbaar zijn. Storybook is hiervan een goed voorbeeld. De gebruikte taal is niet mechanisch.

In deze thesis zal vooral gekeken worden naar de eerste drie punten. Aan de leesbaarheid van de verhalen zal nauwelijks aandacht worden besteed. Dit punt zal niet worden meegenomen, om het onderzoek niet te uitgebreid te maken.

3. De structuur van verhalen

Zoals geconcludeerd is uit de bestudering van vorige werken (zie paragraaf 2.9), moeten de verhalen een goede structuur bevatten. Er is veel werk verricht om de structuur van een verhaal te beschrijven. We maken hier een grof onderscheid in het vele werk dat beschrijft hoe een verhaal structuur moet krijgen:

- grammatica's die het hele verhaal beschrijven
- climax structuren, die beschrijven hoe het verhaal naar een climax toewerkt.

3.1 Grammatica's

In de loop der jaren zijn er vele pogingen ondernomen om verhalen te beschrijven in de vorm van een grammatica. In deze paragraaf zullen de bekendste grammatica's eens nader worden bekeken.

3.1.1 Aristoteles

Aristoteles is de eerste persoon waar bekend van is, dat hij iets heeft opgeschreven over de structuur van een verhaal en zijn plot. Met Poetica[Ari88] schreef hij omstreeks 330 voor Christus al een systematische beschouwing over literatuur. Hij beweert dat een goed toneelstuk dan wel verhaal bestaat uit de delen: begin- midden- eind. Elk verhaal had volgens Aristoteles een begin, waarin de personen en de situatie werd uitgelegd. Het midden was het gedeelte waar de acties gebeurden en het eind maakte het verhaal af.

3.1.2 Propp

Alexander Propp analyseerde in zijn "Morphology of the Folktale" [Pro68] honderdvijftien Russische sprookjes. Propp concludeerde drie dingen: Ten eerste, verhalen worden gekarakteriseerd door functies die worden uitgevoerd door de verschillende karakters uit het verhaal. Ten tweede dat het aantal functies eindig was en ten derde was er een regulier patroon van functies aan te wijzen in de verhalen.
Propp herkende in de honderdvijftien sprookjes éénendertig functies. Deze functies (zie voor een impressie figuur 2.1 en voor alle functies bijlage 1) hebben voor menig programmeur als basis gediend voor het maken van een programma.

Naast de éénendertig functies herkende Propp ook zeven aanwezige karakters:

- De schurk
- De donor
- Held
- Valse held
- De afgevaardigde
- De helper
- De prinses

Elk karakter heeft zijn eigen doelen. Ook heeft elk karakter zijn eigen specifieke eigenschappen. Zo kan bijvoorbeeld alleen de donor de held de magische drank bezorgen.

```
1. A member of a family leaves home
   (the hero is introduced);
2. An interdiction is addressed to the
   hero
   ('don't go there', 'go to this
   place');
3. The interdiction is violated
   (villain enters the tale);
.. ...
.. ...
30. Villain is punished;
31. Hero marries and ascends the throne
   (is rewarded/promoted).
```

Figuur 3.1 Impressie van Propps functies

Toolan [Too88] claimt dat de grammatica van Propp niet erg goed is, omdat de sprookjes terug gebracht moeten worden naar simpele verhaaltjes om in de grammatica te passen. Hij vindt dat een fout van de grammatica. Een ander probleem volgens Toolan is dat de functies en karakters gebaseerd zijn op honderdenvijftien verhalen. Voor dit corpus zijn éénendertig functies genoeg, maar voor andere verhalen zijn misschien wel tweeëndertig of veertig functies nodig. De grammatica is dus alleen representatief voor die Russische sprookjes.

Volgens Turner [Tur92] heeft Propps grammatica een groot probleem. De grammatica weet wel de structuur maar niet de inhoud van het verhaal weer te geven. Propps grammatica kan volgens hem verhalen produceren die goede structuur hebben maar een absurde inhoud.

Mark Lee [Lee94] vertelt over Propp dat de grammatica misschien wel gebruikt zou kunnen worden, maar dat er dan veel aanpassingen gedaan moeten worden. De grammatica is te complex volgens hem.

3.1.3 Greimas herformulering van Propps functies

Greimas [Gre66] erkende evenals Lee [Lee94] dat de functies van Propp te complex en te specifiek zijn. Hij herformuleerde Propps éénendertig functies dan ook terug naar vijf meer algemene functies:

- disruption of a state of equilibrium
- arrival and mission of hero
- trial of the hero
- task of the hero accomplished
- return to original state

Deze functies zijn veel algemener en ook makkelijker toe te passen.

3.1.4 Colby

Colby [Col73] ondervond dat, als hij Propps functies zou gebruiken voor zijn domein (eskimo sprookjes uit Noord Alaska), er teveel uitzonderingen op deze functies gemaakt moesten worden. Ook zou er geen duidelijk patroon in de functies aan te wijzen zijn. Colby heeft daarom een meer systematische grammatica voor verhalen gemaakt.

Colby introduceerde het concept van een "eidon". Eidons, zoals Colby ze noemt zijn "de verhalende gebeurtenissen en omstandigheden" die de bouwstenen vormen voor een speciaal genre of sprookje.

```
Rule 1 Move → M, Resp *
Rule 2 Resp → E * R
Rule 3 M → VM | IM
Rule 4 E → (PA) + (MA)
Rule 5 R → (IR) + (VR)
Rule 6 VM → Fl | Sl | Ml
Rule 7 IM → Vl | Bt | Sp
Rule 8 PA → (En) + (Hs) + (Ch) + (Cn)
Rule 9 MA → (Ak) + (Fh) + (Ru) + (Ps);
           (Tr) + (Me) + (Ma) + (El);
           (Sf) + (Ds) + (Dc)
Rule 10 IR → (Vc) + (Ri) + (Po) + (Rs);
           (Es) + (Re) + (Mr)
Rule 11 VR → (Gr) + (Se) + (At)
```

Figuur 3.2 De grammatica van Colby

Colby onderscheidt de primaire "Motivation" (M), "Engagement"(E) en "Resolution" (R) eidons zoals te zien is in regel één en twee van figuur 3.2. Zij vormen de terminale entiteiten van de grammatica. Deze primaire eidons worden in de rest van de grammatica (regel drie t/m elf uit figuur 3.2) uitgewerkt tot normale eidons. Normale eidons zijn zoals gezegd: "de verhalende gebeurtenissen en omstandigheden". Deze eidons lijken sterk op de functies van Propp. Twee voorbeelden van "Engagement" eidons zijn het flirten van de hoofdpersoon met een vrouw, of het overhalen van iemand om iets uit te voeren. Deze eidons worden in de grammatica weergegeven als (En) en (Ps) en staan voor "Engagement" en "Persuasion".

Colby maakt ook gebruik van "secundaire eidons". Deze komen niet echt terug in de regels van de grammatica. Zij zorgen ervoor dat een verhaal vloeiend verloopt. Voorbeelden van secundaire eidons zijn: het overgaan van de ene scène in de ander door middel van een reis en het verkrijgen van bepaalde vaardigheden door de hoofdpersoon.

3.1.5 Virtual Storyteller

In de Virtual Storyteller wordt gebruik gemaakt van een zeer beperkte grammatica. Er is een begin, waarin de setting en het enige karakter wordt voorgesteld. Tevens wordt de reden van de hoofdpersoon om wat te gaan doen weergegeven (het doel). In het midden van het verhaal probeert de hoofdpersoon zijn doel te vervullen. Zodra dit is gelukt, is het verhaal afgelopen. Als laatste wordt een einde gecreëerd, dat bestaat uit de zin "en ze leefden nog lang en gelukkig". De Virtual Storyteller maakt dus eigenlijk slechts gebruik van een grammatica zoals Aristoteles die al aangaf (begin – midden – eind).

In de bestaande structuur wordt de grammatica vooraf opgelegd aan de hoofdpersoon door de Director Agent (zie paragraaf 2.8).

Sander Faas [Faa02] heeft wel een kleine grammatica bedacht die gebruikt zou kunnen worden bij verdere ontwikkeling (zie figuur 2.3). De grammatica is echter nog niet geïmplementeerd.

```
Begin → Setting Protagonist
Setting → Locale+ Path*
Locale → Environment Object*
Begin → Setting Protagonist
<Protagonist.Locale = Setting.Locale_1>
Middle → Attack Rescue
Attack → Event* AttackEvent
Rescue → Event* RescueEvent
```

Figuur 3.3 Grammatica Virtual Storyteller

Deze grammatica moet ervoor zorgen dat er een plot ontstaat. In het middengedeelte vinden er acties plaats gevolgd door een aanval. Na de aanval kunnen er weer enkele acties plaatsvinden, maar vervolgens moet er een redding ontstaan door het uitvoeren van een reddingsactie. Bij deze grammatica moet wel aangetekend worden, dat deze slechts als mogelijke grammatica wordt gepresenteerd door Faas. De grammatica zou natuurlijk nog uitgebreid kunnen worden.

Hoe de grammatica geïmplementeerd zou moeten worden, wordt door Faas opengelaten.

Binnen deze grammatica is er niet veel ruimte voor veel creativiteit. Er kan slechts één aanval en één redding plaatsvinden. Natuurlijk is het mogelijk voor de karakters om eerst een rondje te lopen (ptrans) of om wat op te pakken(wapen), maar erg veel verschillende verhalen met verrassende gebeurtenissen zullen niet plaatsvinden.

Wat ook opvalt, is dat Faas niets verteld over het gebruik van meerdere karakters. Het ligt voor de hand, dat er bij verdere ontwikkeling van de architectuur, meerdere karakters zullen worden geïmplementeerd.

Als laatste is het de vraag of de grammatica gemakkelijk geïmplementeerd kan worden. Vele programmeurs hebben al geprobeerd om een complexe grammatica te implementeren en zijn er op stukgelopen (o.a. [Lee94] en Meehan [Mee81]).

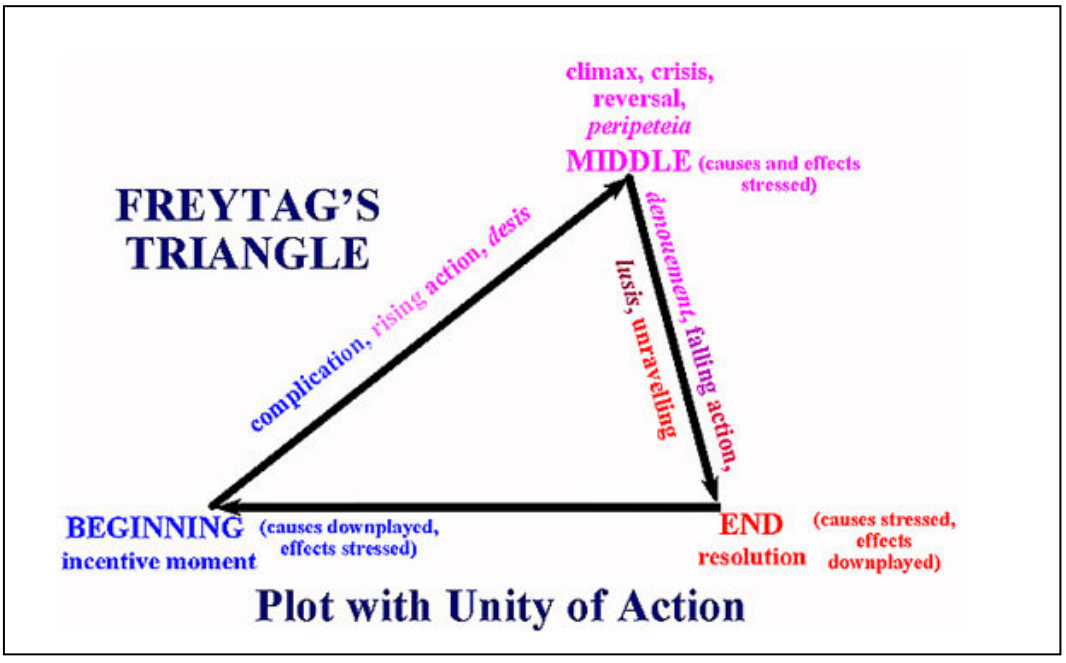
3.2 Climax structuren

Naast de grammatica's zijn er ook nog climax structuren aan te geven die een verhaal kunnen beschrijven. We geven kort enkele climax structuren weer.

3.2.1 De driehoek van Freytag

Freytag [Fre63] beschrijft een climax structuur die gebaseerd is op de grammatica van Aristoteles [Ari88]. Volgens Freytag bestaat een verhaal uit verschillende acts. Elke act begint normaal en daarna wordt er toegewerkt naar een climax. Op het moment van de climax zal er een soort van keerpunt zijn.

Dit keerpunt kan van alles zijn zoals een onthulling van informatie, een fysieke actie of het plotseling begrijpen van iets. Na het keerpunt zal teruggekeerd worden naar het begin, waarna de spanning opnieuw opgebouwd wordt. Freytags driehoek is een zeer bekende structuur die ook op vele verhalen van toepassing is.

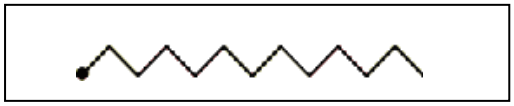


Figuur 3.4 De driehoek van Freytag

3.2.2 Climax structuren van Kimberly Appelcline

Appelcline [App00] onderscheidt 4 verschillende structuren in de verhalen. Deze structuren maken allen gebruik van de grammatica van Aristoteles

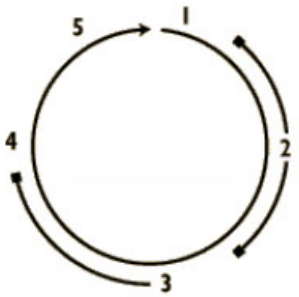
- "The episodic plot" is een continu doorgaande begin-midden-eind structuur van climaxen.



Figuur 3.5 "The episodic plot"

- "The hero's journey" is volgens haar, de structuur die het meest gebruikt wordt in sprookjes. Ze onderscheidt hier 5 onderdelen in het verhaal.

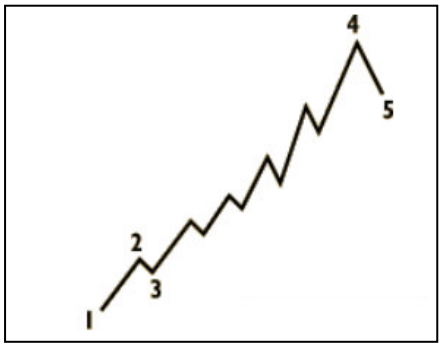
1. call to adventure
2. journey through unfamiliar world; hero is tested
3. supreme ordeal
4. reward
5. return and reintegration into society



Figuur 3.6 "The hero's journey"

Deze climaxstructuur lijkt zeer veel op de herziene grammatica van Propp, gemaakt door Greimas (zie paragraaf 3.1.3).

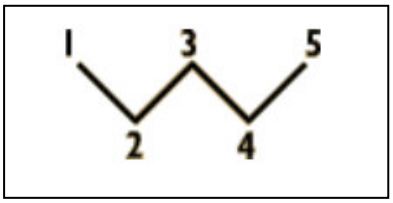
- "The Mountain Plot" maakt gebruik van meerdere miniclimaxen. Zodra het verhaal begint wordt de spanning opgebouwd. De hoofdpersoon komt het eerste probleem tegen en de spanning wordt minder. De hoofdpersoon vindt een manier om het probleem te overwinnen.



Figuur 3.7 "The mountain plot"

De spanning wordt weer opgebouwd tot het volgende probleem (stap 2 en 3 worden meerdere malen herhaald) Het verhaal bereikt zijn climax. De spanning komt op het hoogst. De hoofdpersoon zal nu slagen of falen voor zijn doel.Een kleine afsluiting wordt gemaakt om de lezer goed het verhaal te laten uitlezen.

- Bij “The W plot” komt de hoofdpersoon als het verhaal begint direct een probleem tegen. De hoofdpersoon overwint het probleem en de spanning wordt weer opgebouwd. Net op het moment dat de hoofdpersoon zijn probleem lijkt te overwinnen, valt hij terug. De spanning is dan weer weg. De hoofdpersoon overwint vervolgens ook het tweede probleem en de spanning wordt weer opgebouwd. Uiteindelijk is de hoofdpersoon terug bij zijn normale leven.



Figuur 3.8 “The W plot”

Appelcline [App00] laat aantekenen dat elke structuur weer sub-structuren kan hebben.

De climax structuren van Appelcline worden gebruikt in de wereld van role-playing-games voor het uitwerken van de verhalen. Zoals we reeds bij OZ hebben gezien (paragraaf 2.5) kunnen spellen en verhalen dicht bij elkaar liggen.

3.3 Conclusies over grammatica’s en structuren

Er zijn enkele grammatica’s die gebruikt zouden kunnen worden voor de Virtuele Verhalenverteller. Vooral de functies van Propp zouden een goede start kunnen geven. Hoewel er vele verhalen mee te maken zijn, is toch sprake van een relatief duidelijk en simpele structuur. Grammatica’s hebben echter zo hun problemen.

Het probleem van de grammatica’s, met uitzondering die van Faas, is dat ze de verhalen proberen te beschrijven. Het zijn geen verhalen genererende grammatica’s. De grammatica’s zijn gemaakt aan de hand van de verhalen en niet andersom. Gegenereerde verhalen kunnen eraan voldoen, maar doen het lang niet altijd.
Ook de variatie binnen de verhalen kan sterk in het geding komen als er voldaan moet worden aan een complexe grammatica, hoewel er vele uitbreidingen mogelijk zijn.
Ten derde zijn goede grammatica’s erg complex. Menig programmeur is zich stuk gelopen op een complexe grammatica (o.a. Mark Lee [Lee94] en Meehan [Mee81]).

Het maken van een goede grammatica waarbinnen toch wel veel variatie aanwezig is, is dus erg moeilijk en in het verleden dan ook niet echt goed tot uiting gekomen.

Climax structuren zijn vaak gemakkelijk aan te wijzen in de verhalen. Het creëren van zo’n structuur is minder gemakkelijk.

Voor de nieuwe Virtuele Verhalenverteller is het, het beste om niet een al te complexe grammaticastructuur te maken. Karakters zullen vrijheid moeten krijgen binnen een beperkte grammaticastructuur. Op die manier blijft er wel structuur binnen het verhaal, maar blijft de variatie van verhalen gewaarborgd. De vijf functies waartoe Greimas Propps functie heeft herschreven lijken meer dan voldoende. Deze vijf functies zorgen voor een goede structuur waarbinnen vele verhalen kunnen worden gemaakt. Tevens zorgt de grammatica ervoor dat er een climax structuur zal ontstaan. Immers de held zal een missie moeten ondernemen en dat werkt spanning verhogend. De held zal allerlei problemen op zijn missie tegen kunnen komen, waardoor de driehoek van Freytag vaak aan te wijzen zal zijn.

4. Geloofwaardige karakters

Uit literatuuronderzoek is gebleken dat er autonome, geloofwaardige karakters aanwezig moeten zijn in de architectuur van de Virtuele Verhalenverteller (zie paragraaf 2.9). Mensen moeten zich kunnen verplaatsen in de karakters.

4.1 Wat maakt een karakter geloofwaardig?

Waarom zijn de Disney films toch zo succesvol? Wat maakt deze tekenfilms tot een succes? Het komt niet alleen door de goede marketing. Nee, het komt vooral door de emoties die mensen voelen als ze naar de films kijken. Iedereen leeft mee met de kleine Leeuwenkoning. Mensen houden echt van de karakters en kunnen zich in hen inleven. Het maakt niet uit dat de acties die ze maken niet in de echte wereld kunnen plaatsvinden (denk aan een vliegend tapijt of een pratende leeuw).

Thomas en Johnston [Tho81] hebben veel onderzoek gedaan naar de geloofwaardige karakters in de Disney films. Zij concluderen dat het overbrengen van emoties datgene is, wat de karakters de illusie geeft dat ze leven. Ook Bartneck [Bar02] en [Ell92] zijn van mening, dat emoties een essentieel onderdeel zijn, om karakters geloofwaardig te maken.

In de nieuwe versie van de Virtuele Verhalenverteller zullen er geen geanimeerde karakters zijn, maar de emoties zouden per tekst overgebracht kunnen worden:

Voorbeeld:
De held was zeer kwaad, omdat de draak de prinses gevangen hield.

Behalve Thomas en Johnston heeft ook Bates [Bat94] veel onderzoek gedaan naar geloofwaaridge agents. Ook hij claimt dat emoties nodig zijn om een agent geloofwaardig te maken.

“Emotion is one of the primary means to achieve this believability, this illusion of life, because it helps us know that characters really care about what happens in the world, that they truly have desires”.

Het hebben van emoties is een zeer belangrijke factor om karakters geloofwaardig te maken.

4.2 Een karakter en emoties

Emoties zijn een belangrijke factor om karakters geloofwaardig te maken. Hoe kunnen we op een gebruiker overbrengen dat een karakter emoties ervaart.

Bij stereotype karakters is het mogelijk emoties over te brengen via standaard reacties en doelen (een karakter reageert namelijk altijd op dezelfde manier). We geven hiervan een voorbeeld:

Voorbeeld:
Een prinses loopt door het bos en komt een schurk tegen. Op dat moment zal de prinses wegvluchten. Een prinses is in sprookjes namelijk bijna altijd bang voor een schurk en zal vervolgens wegvluchten. Zodra een prinses een schurk ziet, zal haar doel dus altijd het proberen weg te vluchten worden.

Bij minder stereotype karakters zal dit moeilijk worden. Het éne karakter zal wel wegvluchten bij het zien van een schurk en een ander karakter niet. We geven weer een voorbeeld:

Voorbeeld:
Een oud vrouwtje loopt door het bos en komt een schurk tegen. Ze heeft in haar leven echter al vaker schurken gezien en loopt door zonder de schurk ook maar aan te kijken

Zoals te zien is in het voorbeeld reageert het oude vrouwtje niet stereotiep. Een oud vrouwtje zal normaal gesproken toch ook proberen om weg te komen.

In het geval van het gebruik van stereotype karakters kunnen emoties worden weergegeven door vaststaande acties en doelen. Indien er echter geen stereotype karakters gebruikt worden, zal er een emotioneel model aanwezig moeten zijn. Dit emotionele model moet de emoties van een karakter weergeven.

Naar verwachting zal de nieuwe Virtuele Verhalenverteller gebruik maken van vastomlijnde karakters in een soort sprookje, toch maken we de keuze om gebruik te maken van een emotioneel model. We voeren hiervoor de volgende redenen aan:

- Mocht er in de toekomst voor gekozen worden om de karakters te gaan belichamen in het virtuele muziek centrum dan is het uiterst handig als bekend is welke emoties de karakters hebben. Deze emoties kunnen gebruikt worden om de karakters bepaalde uitdrukkingen te laten aannemen.
- Door emoties toe te voegen aan de acties is het mogelijk om een verteller nuances in verhalen te laten aanbrengen. Voorbeeld: als een karakter ergens rondloopt en hij is boos. Dan kan de verteller als hij weet dat, dat karakter boos is vertellen dat het karakter stampvoetend rondloopt i.p.v. dat de agent gewoon rondloopt.
- Mocht er overgestapt worden naar meer normale verhalen dan sprookjes, dan zullen er minder karakters zijn die aan de verwachtingen voldoen (m.a.w er is niet een duidelijke held, een prinses en een schurk). Het gebruik van een emotioneel model bevordert dan de geloofwaardigheid van de karakters en is in dat geval noodzakelijk.
- Als er een emotioneel model wordt toegevoegd is het mogelijk om te gaan. experimenteren met de emotionele waarden. Het is bijvoorbeeld interessant om te kijken hoe een karakter die qua emoties een schurk is, om gaat met de opdracht om de prinses te redden uit de handen van een andere schurk.

4.3 Emotionele modellen

In de vorige paragraaf is de keuze gemaakt om een emotioneel model te gaan gebruiken voor de karakters. Nu zullen we gaan kijken naar verschillende emotionele modellen.

Eerst beschrijven we twee zogenaamde event-appraisal modellen. Dit zijn modellen die gebaseerd zijn op het waarderen van gebeurtenissen. Als tweede beschrijven we een emotioneel model uit de AI hoek: TABASCO.

4.3.1 Rosemans model

Roseman heeft een model gemaakt waarin 5 cognitieve dimensies bepalen of een emotie opkomt en welke emotie het is. [Ros90].

De eerste dimensie betreft de consistentie met de doelen van een persoon. Als een gebeurtenis de waarschijnlijkheid dat een doel wordt behaald verhoogd, dan worden er positieve emoties ervaren en indien een gebeurtenis de waarschijnlijkheid dat een doel wordt behaald verlaagd, dan worden er negatieve emoties ervaren. Deze dimensie kan ook worden gezien als de beschrijving of de huidige situatie overeenkomt met de motivaties van de persoon of niet. Deze dimensie kent de staten “positive”, “negative”, “motive consistent” en “motive inconsistent”. “Motive consistent” correspondeert altijd met een het ervaren van positieve emoties en “motive inconsistent” correspondeert altijd met het ervaren van negatieve emoties

De tweede dimensie beschrijft de motivatie voor een gebeurtenis. Een motivatie voor een gebeurtenis kan het verkrijgen van een beloning (Appetitive) of het voorkomen van straf zijn (Aversive).

De derde dimensie beschrijft of een gebeurtenis wordt gezien als zeker of als een mogelijkheid. Deze dimensie kent de staten “certain” en “uncertain”. De staat “uncertain” (onzeker) is opgenomen in het model om ervoor te zorgen dat de emotie verrassing in het model te brengen [Ros90].

De vierde dimensie beschrijft hoe een persoon over zichzelf denkt. Een persoon kan zichzelf als sterk of zwak zien in het omgaan met een bepaalde gebeurtenis.

De vijfde dimensie beschrijft hoe de gebeurtenis ontstaat. Deze dimensie kent de staten: door omstandigheden, door anderen of door zichzelf ("the circumstances", "other-caused" or "self-caused").

Met behulp van deze 5 dimensies is het mogelijk om de volgende tabel uit figuur 4.1te maken.

Om te laten zien hoe de tabel werkt geven we een voorbeeld:

Voorbeeld:
Als een persoon verwacht dat een motive consistente gebeurtenis plaats zal vinden met een zekerheid van 70 % en hij tevens denkt dat hij sterk is in zulke situaties, dan zal hij hoop ervaren.

		Positive Emotions ***Motive-Consistent***		**Negative emotions** *Motive-Inconsistent*		
		Appetive	**Aversive**	**Appetive**	**Aversive**	
Circumstance-Caused	*Unknown*	Surprise				
	Uncertain	Hope		Fear		**Weak**
	Certain	Joy	Relief	Sadness	Distress, Disgust	
	Uncertain	Hope		Frustration		**Strong**
	Certain	Joy	Relief			
Other-Caused	*Uncertain*	Liking		Dislike		**Weak**
	Certain					
	Uncertain			Anger		**Strong**
	Certain					
Self-Caused	*Uncertain*	Pride		Shame, Guilt		**Weak**
	Certain					
	Uncertain			Regret		**Strong**
	Certain					

Figuur 4.1 Rosemans model

Dit model heeft enkele problemen met het omgaan met situaties, waarin een persoon twee verschillende benaderingen heeft.

Voorbeeld:
Een persoon die afstudeert en weet dat de leraar hem niet mag en hij nooit zal slagen, maar hij weet ook dat zijn werk nog niet goed genoeg is om af te studeren,

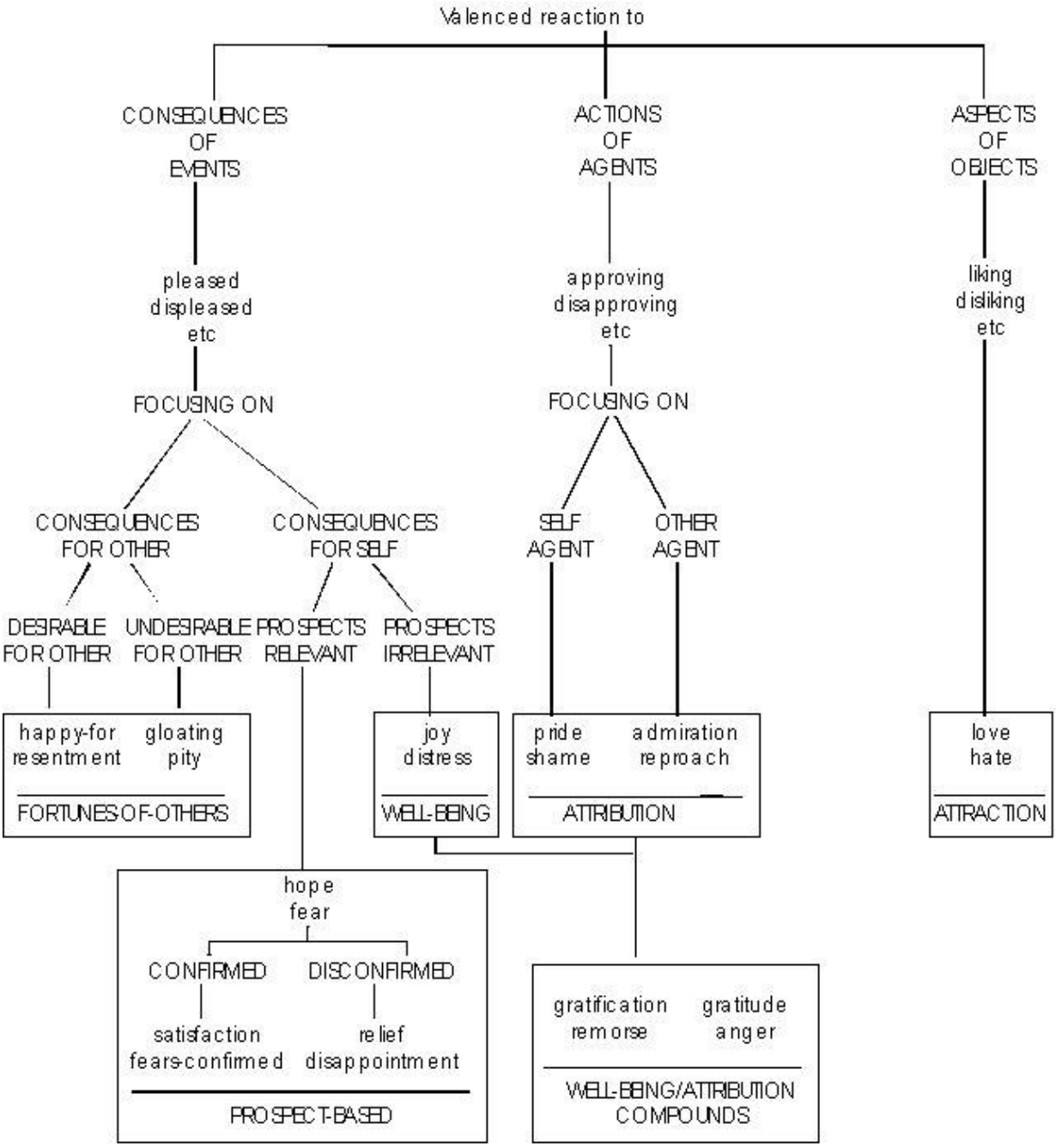
Het model van Roseman kan dan niet de goede staat voorspellen. Dit komt omdat de vijfde dimensie in twee staten verkeert op hetzelfde moment. Deze staten zijn namelijk zowel "Other caused" (schuld van de leraar dat ik niet slaag) als "Self Caused" (het is mijn eigen schuld, moet ik maar beter werk maken).

Zoals A.J. van Kesteren [Kes01] ook laat aantekenen over het Roseman model, is het ook een beperking dat Roseman niets zegt over de intensiteit van de emoties die worden ervaren.

4.3.2 The OCC model

Ortony, Clore en Collins hebben een van het meest gebruikte event appraisal model gemaakt [Ort88]. De theorie van Ortony Clore en Collins gaat ervan uit dat emoties voortkomen uit bepaalde gedachten en interpretaties. De gedachten zijn gesplitst in drie onderdelen: focus op gebeurtenissen, focus op agents en focus op objecten.
Wanneer iemand gefocust is op gebeurtenissen, dan doet hij dat omdat hij geïnteresseerd is in de consequentie van de gebeurtenis. Als iemand gefocust is op agents, dan is hij dat vanwege de acties van de persoon. Als iemand gefocust is op objecten, dan is hij dat vanwege bepaalde eigenschappen van het object.

De drie verschillende soorten gedachten kunnen weer worden opgesplitst. Als iemand bijvoorbeeld gefocust is op gebeurtenissen, dan kan hij gefocust zijn op de consequenties die de actie voor zichzelf heeft of op die voor een ander.



Figuur 4.2 Het OCC-model

We geven een voorbeeld (afkomstig uit Ort88) om de verschillen tussen de drie gedachtes weer te geven. Het voorbeeld gaat over een persoon die merkt dat zijn buurman kinderen slaat:

Voorbeeld:
"If such a person focuses only on the neighbour's role as the agent of child-beating, judging it as blameworthy because of its violation of certain standards, his valenced reaction towards the neighbour could be realized as an Attribution emotion such as reproach. The person could also focus on one or more aspects of a child-beating event. If he focuses only on its undesirability it might cause him distress. He could also focus on the plight of his neighbour's children and experience pity. Finally the person might focus on his neighbours qua (unappealing) object, given rise to an attraction such as hatred".

De combinatie van al deze factoren leidt tot de het plaatje dat getoond wordt in figuur 4.2. Zoals te zien is in de figuur worden er in het OCC model 22 verschillende emoties herkend (happy for, joy, pride e.d.). Er zijn speciale samengestelde emoties (gratification, remorse, gratitude and anger). Deze emoties vinden plaats als er twee andere emoties op hetzelfde moment plaats vinden (gratification ontstaat uit joy en pride).

De intensiteit van de emoties worden bepaald door locale variabelen. De 3 meest belangrijke locale variabelen zijn: Desirability, Praiseworthiness en Appealingnes. Desirability is gekoppeld aan gebeurtenissen en wordt geëvalueerd met het oog op doelen. Praiseworthiness is gekoppeld aan agents en wordt geëvalueerd vanuit de eigen standaarden. Appealingnes is gekoppeld aan objecten en wordt geëvalueerd m.b.v. attitudes.

De locale variabelen worden in hun geheel weergegeven in de volgende tabel:

Events	**Agents**	**Objects**
Desirability	Praiseworthiness	Appealingness
Desirability for other	Strength of cognitive unit	Familiarity
Deservingness	Expectation deviation	
Liking		
Likelihood		
Effort		
Realization		

Figuur 4.3 Locale variabelen OCC Model

Naast de locale variabelen heeft het OCC model ook nog globale variabelen. Deze globale variabelen (Sense of reality, proximity, unexpectedness and arousal) beïnvloeden de intensiteit van alle emoties.

De waarden van de variabelen verschillen van persoon tot persoon.

Bartneck [Ba02]noemt het OCC model een zeer goed startpunt, maar tevens beschrijft hij enkele nadelen die het gebruik van het OCC model geven:

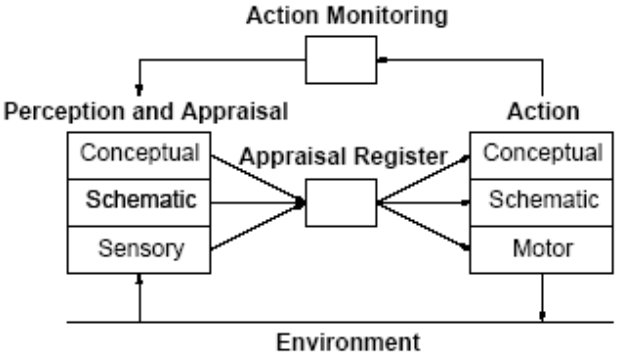
- Het model is erg complex: Er zijn veel emoties en er moet veel bekend zijn om een emotie eventueel uit te rekenen.
- Er is geen geschiedenis functie: Als je 10 bananen achter elkaar krijgt, zul je er de eerste keer blijer mee zijn dan de laatste. In het OCC model ben je even blij met elke banaan. Er zou hiervoor volgens Bartneck een functie moeten komen die de geschiedenis van alle gebeurtenissen bijhoudt en aan de hand daarvan aanpassingen verricht in het waarderen van gebeurtenissen.
- Het OCC model beschrijft niet hoe er met de emoties moet worden omgegaan wanneer ze eenmaal berekend zijn. De emoties zullen op de een of andere manier leiden tot acties of doelen.
- De interactie tussen de verschillende emoties wordt niet beschreven. Een bepaalde gebeurtenis zal een persoon niet altijd blij maken. In plaats daarvan zou het een persoon bijvoorbeeld minder verdrietig kunnen maken. Hoewel dit voor de hand ligt, houdt het OCC hier geen rekening mee.

4.3.3 Tabasco

Tabasco [Pet98] gaat uit van een architectuur met drie lagen. De drie lagen zijn: conceptueel, schematisch en zintuiglijk (“Sensory”) . Het onderliggende psychologische idee bij Tabasco, is dat de scheiding tussen het zintuiglijke, schematische en conceptuele proces niet alleen betekenis geeft wat betreft waardetoekenning aan gebeurtenissen, maar ook aan het genereren van acties.

De hoofdcomponenten zijn dus verantwoordelijk voor waardetoekenning aan gebeurtenissen uit de omgeving en de generatie van acties.

Wat betreft de component voor de waardetoekenning, is de conceptuele laag verantwoordelijk voor abstracte redenering en gevolgtrekkingen. De zintuiglijke laag beschikt over zogenaamde "feature detectors" voor het omgaan met plotselinge, korte en intense stimuli zoals de consequenties van een gebeurtenis. In de schematische en planmatige laag worden de gebeurtenissen vergeleken met de zogenaamde "social and self schemata". Deze schema’s bevatten informaties over de eigen en de sociale standaard.



Figuur 4.4 De TABASCO architectuur

De actie generatie component zorgt voor de lange-baan planning in de conceptuele laag. De generatie van actie bereidheid gebeurt in de schematische laag en actie generatie gebeurt in de motorische laag.

De uitkomst van de actie generatie component, komt via de “Action Monitoring” component weer terug bij het waardetoekenning component. Het “Appraisal Register” zorgt voor de informatiestroom tussen de waardetoekenning component en de component voor de actie generatie. Beide hoofdcomponenten oefenen dus invloed op elkaar uit.

Nadeel van deze architectuur is dat er niets wordt verteld over de te gebruiken emoties en welke intenties er toegekend moeten worden bij het waarderen van gebeurtenissen.

4.3.4 Conclusies over emotionele modellen

In de vorige drie paragrafen zijn respectievelijk het Rosemans model, het OCC model en Tabasco beschreven.

Het OCC model, Rosemans model en TABASCO kunnen allen gebruikt worden in een agentstructuur voor de Virtuele Verhalenverteller.

Het Roseman model is de minst bekende van de drie. Het model vertelt helemaal niets over het toekennen van intensiteiten aan emoties. Er wordt alleen duidelijk welke emotie er wordt ervaren en niet de intensiteit van de emotie.

Tabasco is een bekender model dan Rosemans model. Het is een model waar steeds vaker aan gerefereerd wordt, maar het is nog niet veelvuldig geïmplementeerd. Het is een behoorlijk complex model om te implementeren. Er is veel informatie nodig van de omgeving. Ook wordt er in het model niets gezegd over de te gebruiken emoties en hoe het waarderen van gebeurtenissen daadwerkelijk gebeurt.

Het OCC is het momenteel het bekendste en meest geaccepteerde model. Het OCC model beschrijft duidelijk meerdere emoties en zoals A.J. van Kesteren [Kes01] ook al duidelijk maakt bestaat er in het OCC model de mogelijkheid om de emoties een intensiteit mee te geven. Het grootste nadeel van het OCC model is dat er niets verteld wordt over wat er met de emoties moet gebeuren. Tevens is ook dit model redelijk complex.

Voor de Virtuele Verhalenverteller willen we een goed model dat vrij gemakkelijk geïmplementeerd moet kunnen worden. We maken uiteindelijk de keuze voor het OCC model. Dit model is algemeen geaccepteerd. De nadelen genoemd door Bartneck schuiven we terzijde. Er zijn namelijk meerdere implementaties gemaakt (zie paragraaf 3.4) die de nadelen wegwerken en als voorbeeld kunnen dienen.

4.4 Reeds gemaakte implementaties van het OCC model

In paragraaf 3.3.3 hebben we de keuze gemaakt om onze implementatie te gaan baseren op het OCC model. Aan het gebruik van een model gebaseerd op het OCC model kleven natuurlijk voor en nadelen. We willen door het bekijken van enkele andere implementaties van het OCC model voorkomen dat we grote fouten maken en/of het wiel opnieuw uitvinden.

4.4.1 SHAME

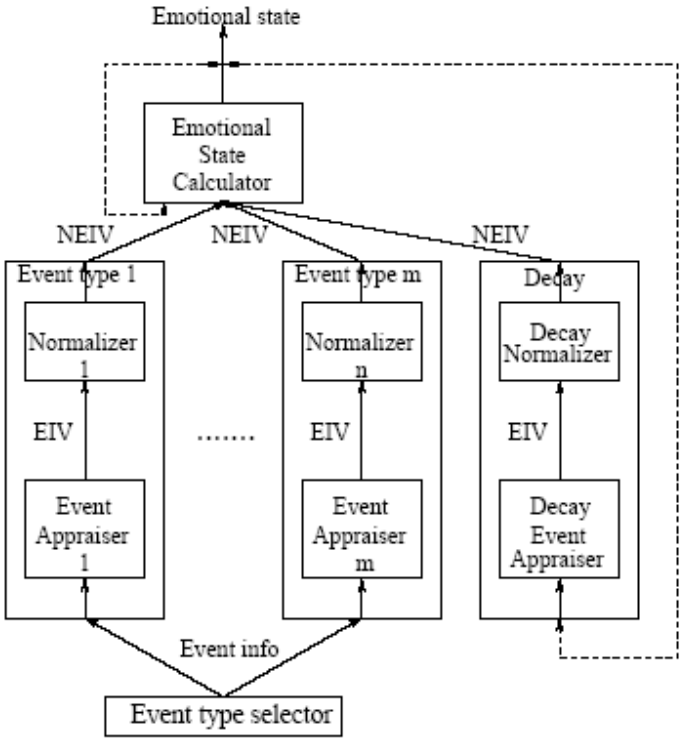
A.j. van Kesteren presenteert met zijn SHAME [Kes01] [Poe02] een neuraal netwerk architectuur voor het leren van emoties. Dit systeem is gebaseerd op het OCC model.

Het systeem maakt gebruik van emotieparen (zie figuur 4.5). Een emotiepaar kan een waarde hebben tussen de -100 en +100. Als de waarde onder nul is wordt de negatieve emotie ervaren en als de waarde van het emotiepaar boven nul is, wordt de positieve emotie ervaren.

Pos. emotion types	Neg. emotion types
Joy	Distress
Hope	Fear
Satisfaction	Fears-confirmed
Relief	Disappointment
Pride	Shame
Admiration	Reproach
Happy-for	Resentment
Gloating	Pity
Love	Hate

Figuur 4.5 Emotie paren in SHAME

Een “event appraiser” berekent voor elke event de “emotion impulse” (EIV). Elk event type wordt gewaardeerd door een specifiek neuraal netwerk. De nieuwe emotionele staat wordt vervolgens berekend door de “emotional state calculator” (ESC).



Figuur 4.6 De SHAME architectuur

De ESC geeft deze emotionele staat als uitvoer. Deze emotionele staat wordt vervolgens weer gebruikt als invoer voor een ander neuraal netwerk dat zorgt voor de tijdelijke aard van bepaalde emoties (“Decay”).

De emotionele staat van een agent heeft geen invloed op de actie of de selectie van de doelen va de agent. Het heeft de functie om de als een simulatie voor verschillende van te voren gedefinieerde persoonlijkheden.

4.4.2 Action Selection Mechanism for Emotional Agents (ASMEA)

G.J. van Burghouts gaat met zijn ASMEA ([Bur02]) verder met SHAME. In ASMEA bepalen vier sequentiële subprocessen de intensiteit van een emotie: waardering (“appraisal”), activering (“activation”), remming (“inhibition”) en zelfbeheersing (self-control).
De waardering van de emotie gebeurt met de zogenaamde “ event appraiser” die is overgenomen van SHAME. (zie vorige paragraaf).

$$eventAppraisal :: Event \rightarrow [(Emotion, Intensity)]$$

Figuur 4.7 De waardering van een emotie in ASMEA

De emoties die door de “event appraiser” zijn gekozen moeten in de volgende fase geactiveerd worden. Van invloed hierop zijn zaken zoals geheugen, verwachtingen en het wereld model. De emoties krijgen vervolgens een “potency value”. Deze “potency value” geeft de kans aan dat deze emotie ervaart wordt.

$$\begin{aligned} &emotionActivation :: \\ &\quad [(Emotion, Intensity)] \rightarrow \\ &\quad (ElapsedTime, PreviousIntensity) \rightarrow \\ &\quad Expectancy \rightarrow Discrepancy \rightarrow \\ &\quad [(Emotion, Potency)] \end{aligned}$$

Figuur 4.8 Activatiemechanisme in ASMEA

Remming kan vervolgens leiden tot het onderdrukken van bepaalde emoties. Vooral in negatieve omstandigheden worden vaak positieve emoties benadrukt en negatieve emoties onderdrukt. Deze factor is afhankelijk van de persoonlijkheid van een agent.

$$\begin{aligned} &emotionsInhibition :: \\ &\quad [(Emotion, Potency)] \rightarrow \\ &\quad [(Emotion, EmotionImpact)] \rightarrow \\ &\quad [(Emotion, Intensity)]. \end{aligned}$$

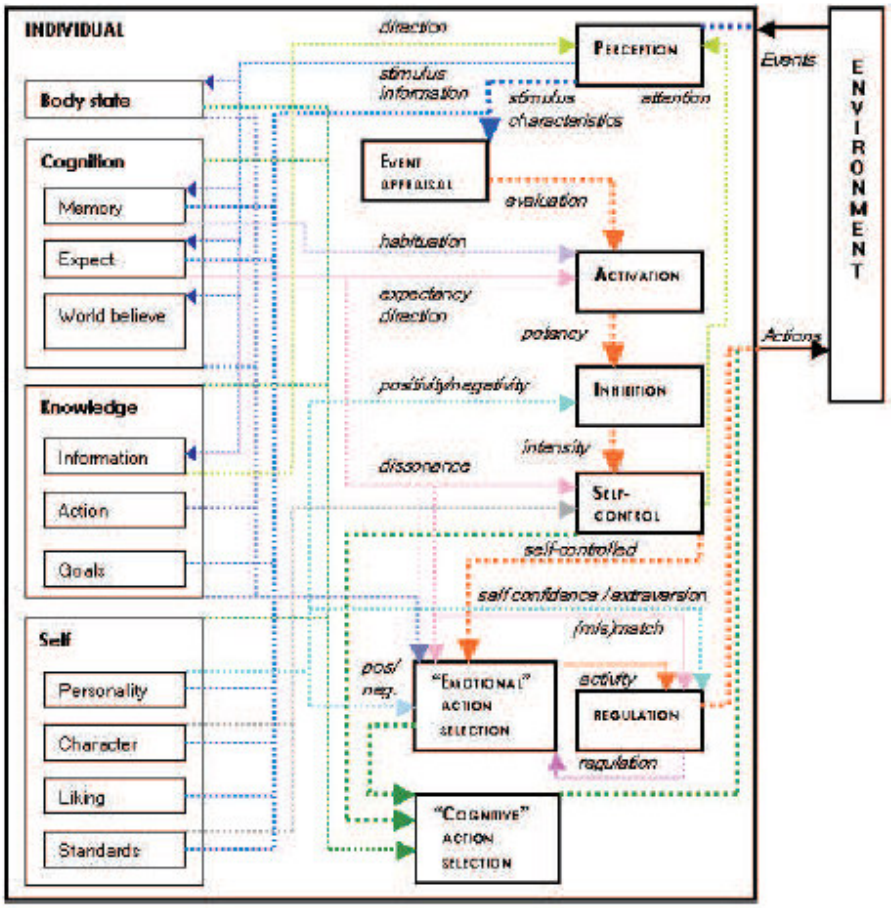
Figuur 4.9 Remming in ASMEA

Als laatste bepaalt de zelfbeheersing van een persoon de uiteindelijke intensiteit van een emotie. Van invloed hier zijn onder andere de normen en het karakter van een agent

$$
\begin{aligned}
&selfcontrolExpectedBehavior :: \\
&\quad [(Emotion, Intensity)] \rightarrow \\
&\quad Expectancy \rightarrow Strategies \rightarrow \\
&\quad [(Action, PraiseworthinessOfAction)] \rightarrow \\
&\quad [(Emotion, Intensity)]
\end{aligned}
$$

Figuur 4.10 Zelfbeheersing in ASMEA

Dit alles heeft uiteindelijk geleid tot de architectuur die te zien valt in figuur 4.11. Het gedrag van een agent is zoals te zien niet alleen van de emoties afhankelijk, maar ook van de staat van het lichaam, perceptie van gebeurtenissen, waarneming, kennis en de persoonlijkheid. Het gedrag zal vervolgens verantwoordelijk zijn voor de daadwerkelijke actie.

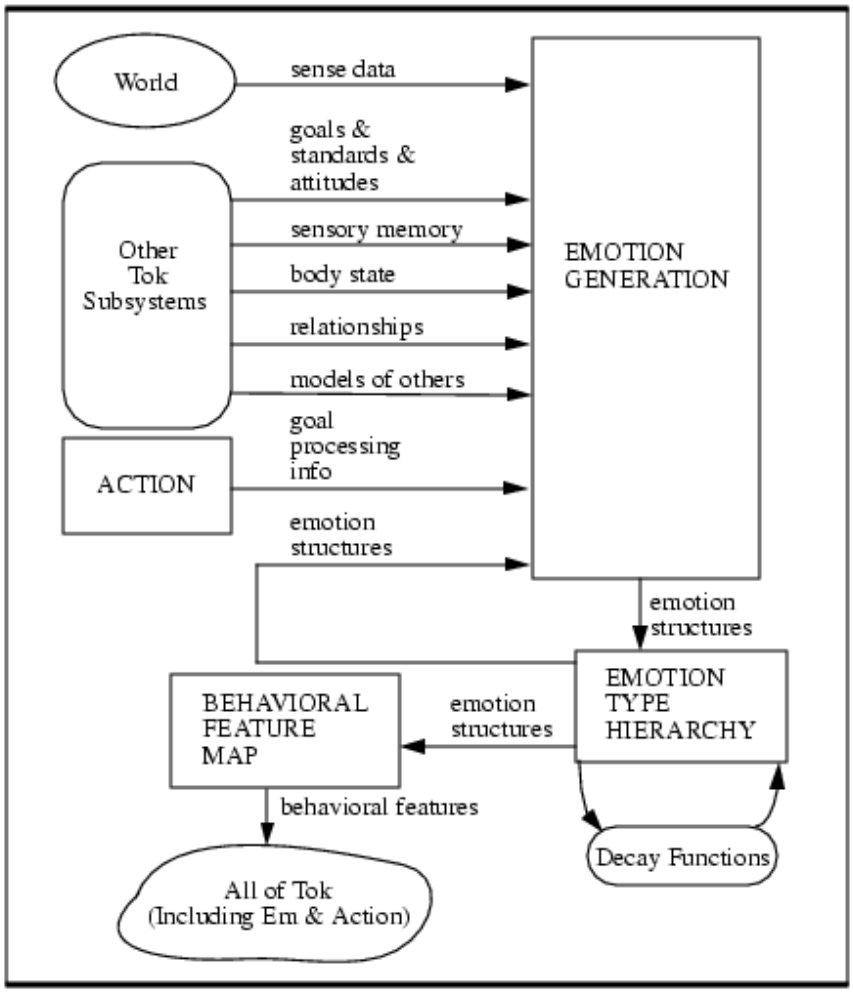


Figuur 4.11 De architectuur gepresenteerd door A J. van Burghouts

Hoewel uit de architectuur niet helemaal duidelijk wordt hoe er wordt omgegaan met doelen zien we in de implementatie ([Bur02]) terug, dat het gedrag van een agent leidt naar bepaalde acties zoals eten, drinken en aanvallen. Zo zijn het aanvallen van andere agents en het pakken van lekkere dingen voorbeelden van superieur gedrag. Deze acties worden gezien als subdoelen van het hoofddoel, dat overleven is.

4.4.3 Em & Hap

W. Scott Neill Reilly heeft voor de agent architectuur TOK een emotie architectuur gemaakt, genaamd Em [Rei96]. TOK is onderdeel van het OZ project (zie paragraaf 2.5). In figuur 4.12 is de Em architectuur met zijn interactie met de rest van TOK te zien. Em krijgt informatie binnen van de wereld, van TOK en van het actieselectiesysteem (Hap). Uit deze informatie worden emoties gegenereerd.



Figuur 4.12 De Em architectuur

Het genereren van deze emoties gebeurd met behulp van zogenaamde Demons. Een demon is een regel die onder bepaalde voorwaarden wordt aangeroepen. Informatie die bij de emotie generator binnenkomt, doorloopt alle demons. Zodra aan de voorwaarde van een demon voldaan wordt, wordt een zogenaamde “Emotion Structure” gecreëerd. Zo wordt er in de demon van figuur 4.13 een “Emotion Structure” gecreëerd voor de emotie frustratie.

```
Demon:
 Name: em-update-frustration-demon
 if (and (not-empty(plan-failures)))
         (B := first-elmt(plan-failures))
         (Importance(B) > 0))
 then {
   struct = make-emotion-structure {
              type = FRUSTRATION
              cause = B
              direction = NIL
              intensity = Importance(B)
             };
   store(struct);
   remove(B,plan-failures);
 }
```

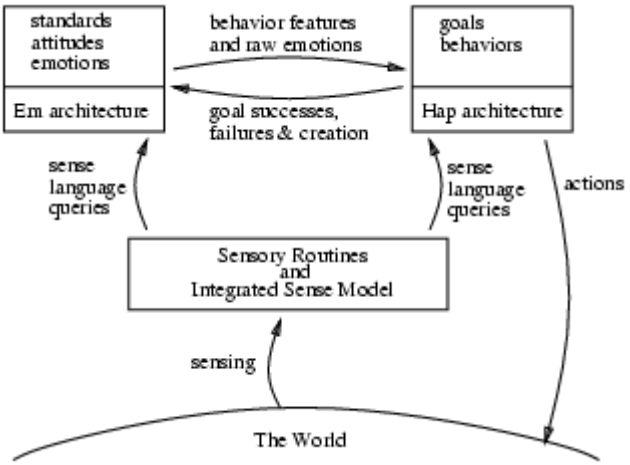
Figuur 4.13 Het creëren van een Emotion Structure in Em

Er kunnen onbeperkt veel demons aangemaakt worden door de maker. Het is ook mogelijk dat er meerdere “Emotion Structures” worden gemaakt voor dezelfde emotie. Per cyclus worden vervolgens uit alle “Emotion Structures” via een logaritmische functie de daadwerkelijke intensiteiten van de emoties bepaald. Ook worden er per cyclus functies op de emoties uitgevoerd om de tijdelijke aard weer te geven.

```
unfriendly(a) := dislike(a) + anger(a)/2 +
                 distress/3 + reproach(a)/3
                 + resentment(a)/5
```

Figuur 4.14 Het maken van een “behaviour”

Uit de daadwerkelijke emoties worden waardes toegekend aan gedragskenmerken (“Behavior features”). Voorbeelden van gedragskenmerken zijn vriendelijk, onvriendelijk, agressief, trots en superieur. Binnen Em bestaat een vaste lijst met meer dan twintig gedragskenmerken. Het toekennen van waardes aan deze gedragskenmerken gebeurt met behulp van scripts (zie figuur 4.14).



Figuur 4.15 de complete Tok architectuur

Deze gedragskenmerken, worden doorgegeven aan een ander onderdeel van het Tok systeem (zie figuur 4.15) genaamd Hap. Hap is het door Loyall gemaakte actieselectiesysteem van Tok [Loy97]. Het krijgt naast de gedragskenmerken ook informatie binnen over de wereld.

De gedragskenmerken hebben op veel manieren invloed op de doelen van een agent. De belangrijkste en meest krachtige manieren waarop gedragskenmerken van invloed zijn op de doelen, zijn volgens Reilly:

- Ze kunnen een doel toevoegen. (Als de agent erg blij is kan ze het doel krijgen om te gaan dansen),
- Ze kunnen de belangrijkheid van het doel veranderen. (Als een agent droevig is, is het belangrijker voor haar om iets leuks te gaan doen, dan als ze al heel blij is)
- Ze kunnen een doel makkelijker laten slagen. (Als je in een negatieve stemming bent, neem je vaak genoegen met een slechtere uitvoering van een doel) Voorbeeld: Job moet een verslag maken, maar heeft er geen zin in. Job vindt het vervolgens niet nodig om er plaatjes bij te doen, teveel werk.

Naast het invloed uitoefenen op de doelen, oefenen de gedragskenmerken ook invloed uit op de daadwerkelijke acties. We noemen weer de belangrijkste:

- Ze beïnvloeden de keuze hoe een doel uit te voeren. (Voorbeeld: Bij het poolen moet je een bal proberen te maken. Als je nerveus bent kies je voor de makkelijke optie, maar als je je superieur voelt, kies je voor de moeilijke).
- Verandering van gedrag. (Voorbeeld: Jack is woedend en gaat lopend naar huis. Na een kwartier klaart hij op, loopt terug en vraag John hem met de auto te brengen).
- Kiezen tussen meerdere mogelijkheden. (Als je kunt kiezen om iemand mee te nemen, dan zijn de gedragskenmerken t.o.v. de personen die je mee kunt nemen bepalend).

4.4.4 Conclusies over reeds gemaakte implementaties

Alle drie genoemde implementaties hebben hun eigen voor- en nadelen. We laten ons inspireren door alle gemaakte implementaties. In het bijzonder het gebruik van emotiekoppels (SHAME), het gebruiken van emoties voor het beschrijven van het gedrag van een agents, al dan niet met behulp van scripts (ASMEA en Em) en het gebruik van het gedrag voor het bepalen van doelen (implementatie ASMEA).

In het volgende hoofdstuk zullen we onze eigen architectuur presenteren. We zullen hierbij meermaals wijzen op eerder gedane implementaties.

Onderdeel 2: Onze Aanpak

5. Architectuur en de structuur van verhalen

We willen voor de nieuwe Virtuele Verhalenverteller een architectuur maken, die zorgt voor een goede structuur in de verhalen. We nemen hierbij voor de nieuwe architectuur de architectuur gemaakt door Sander Faas [Fa02] als uitgangspunt. Net als Sander Faas zullen wij ons ook richten op het maken van Sprookjes.

5.1 Aanwezige grammatica

In paragraaf 3.1.5 hebben we het al gehad over de door Sander Faas ontwikkelde grammatica. We hebben daar reeds de nadelen van de grammatica beschreven en willen dan ook geen gebruik van deze grammatica maken.

Toch staat het idee achter de grammatica ons wel aan. Autonome agents worden binnen de structuur gehouden door een Director Agent. Voordat een Actor agent een actie uitvoert, moet hij eerst toestemming vragen aan de Director Agent.

Het idee van toestemming vragen aan de Director voor elke actie nemen we over, maar we zullen voor de structuur waarbinnen de agents zich moeten houden iets anders gebruiken dan een grammatica. We zullen gebruik gaan maken van episodes.

5.2 Episodes

Voor de nieuwe Virtuele Verhalenverteller willen we gebruik maken van een iets andere aanpak. Graag willen we wel de architectuur wat betreft het gebruik van een Director en karakters behouden, maar we willen binnen de structuur plaats maken voor meer creativiteit en meerdere karakters. De karakters moeten zo vrij mogelijk gelaten worden om deze creativiteit te verkrijgen, maar toch zal de Director de structuur van het verhaal onder controle moeten hebben. Dit willen we bereiken door gebruik te maken van zogenaamde episodes. We zullen nu eerst uitleggen wat precies episodes zijn en hoe ze werken. In paragraaf 5.2.3 gaan we verder in op de keuze voor episodes.

5.2.1 Wat is een episode?

We willen een verhaal opbouwen uit verschillende episodes. We definiëren hierbij een episode als:

Een onderdeel van een verhaal, waarin één belangrijke functie voor het verhaal wordt uitgewerkt.

De functies die binnen een episode worden uitgewerkt baseren we op de vijf functies van Greimas (zie paragraaf 3.1.3). We hebben gekozen voor de volgende functies, die dus allen, afzonderlijk, in één episode worden uitgewerkt:

- state of equilibrium
- disruption of a state of equilibrium
- mission of hero
- return to state of equilibrium

Zoals te zien is hebben we de functies “trial of the hero” en “task of the hero accomplished” samengenomen tot “mission of the hero”. We doen dit omdat we van mening zijn, dat de functies apart, te specifiek zijn voor één episode. Bovendien willen we niet al van te voren vastleggen dat een held altijd zijn missie volbrengt.

We geven kort weer wat elke functie inhoudt:

State of equilibrium
In de episode die deze functie uitwerkt, zal allereerst de wereld moeten worden beschreven en de karakters moeten worden geïntroduceerd, die van belang zijn voor de twee middelste episodes.

Niet alle karakters moeten direct worden geïntroduceerd. Het zou kunnen dat een karakter niet eens in de buurt komt van de hoofdpersoon en dus eigenlijk niet deelneemt aan het verhaal. Zo’n helper hoeft niet te worden geïntroduceerd. Hij komt pas in beeld als hij actief gaat deelnemen. Dit moet de verscheidenheid aan verhalen bevorderen.

Disruption of a state of equilibrium
In de eerste episode is de zogenoemde “state of equilibrium” beschreven. In de volgende episode wordt: *“Disruption of a state of equilibrium ”* beschreven. De beginstaat zal dus worden verstoord. Een schurk moet dan bijvoorbeeld een prinses gevangen nemen. In deze episode wordt dus de missie die de held moet uitvoeren geïntroduceerd.

Mission of hero
In de episode, die deze functie beschrijft, zal de held moeten proberen om zijn missie te halen die in de vorige episode is geïntroduceerd.

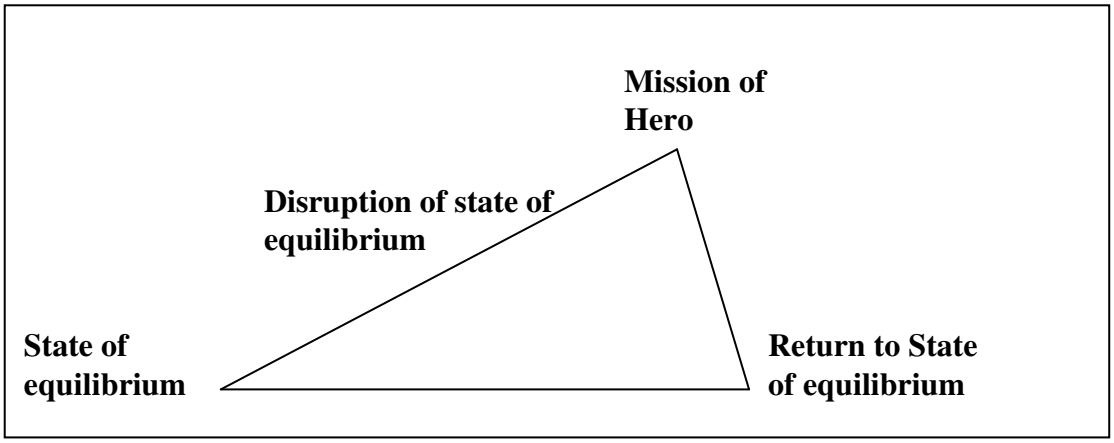
Voorbeeld:
Als in de vorige episode de prinses gevangen is genomen, dan zal in deze episode de held de opdracht krijgen om haar te bevrijden.

Return to state of equilibrium
De episode waarin de functie *“return to state of equilibrium”* moet worden uitgewerkt, zal in deze versie niet veel voorstellen. Het zal uit niet veel meer bestaan dan een concluderende zin zoals: “*en ze leefden nog lang en gelukkig*”, maar het zou ook kunnen dat het einde droevig is en de concluderende zin luidt: *“Vele jaren later huilden de mensen nog vanwege deze treurnis”*. Hoewel deze episode in deze versie nog niet veel voorstelt, moet er met deze episode wel degelijk rekening worden gehouden in de architectuur. In latere versies kan er misschien een volwaardige beschrijving van het einde worden gemaakt.

De keuze voor de functies is niet zomaar gemaakt. Deze vier functies zijn afgeleid van de éénendertig functies van Propp[Pro68] (zie ook paragraaf 3.1.2 en 3.1.3). Propp heeft voor het maken van zijn éénendertig functies honderdenvijftien sprookjes geanalyseerd. De gekozen functies zijn algemener en minder specifiek. In theorie zouden er dus minimaal honderdenvijftien verschillende verhalen mee gemaakt moeten kunnen worden. Waarschijnlijk ligt het aantal, door het algemene karakter van de functies nog veel hoger.

De functies moeten bovendien zorgen voor structuur en een climax. Dit lijkt het geval. De driehoek van Freytag is aan te wijzen (zie paragraaf 3.2.1) en de functies lijken sterk op de onderdelen van “*The hero’s journey”* van Kimberly Appelcline (zie paragraaf 3.2.2.).

Bovendien hebben ook alle honderdenvijftien Russische sprookjes een structuur, die met behulp van deze functies te maken is.



Figuur5.2 De driehoek van Freytag weergegeven in de functies voor de episodes

De keuze van deze functies heeft natuurlijk wel gevolgen voor de karakters die gebruikt kunnen worden. Meer hierover vindt u in paragraaf 5.2.2 onder het kopje “Karakters”.

5.2.2 Hoe werken episodes?

Een episode wordt opgebouwd uit 5 elementen die ervoor moeten zorgen dat de autonome acteurs zich onder de invloed van de Director aan de verhaallijn houden. De vijf elementen waaruit een episode opgebouwd wordt zijn:

- Setting constraints
- Episodische constraints
- Episodische doelen
- Karakters
- Informatie

Hieronder worden alle punten één voor één uitgelegd.

Setting constraints
Setting constraints geven aan waar de setting aan moet voldoen. Zo wordt bepaald welke objecten en locaties aanwezig moeten zijn in de bewuste episode. Er kan ook wat gezegd worden over hun eigenschappen. De Director zal hiermee rekening moeten houden als hij de wereld gaat maken waarin het verhaal zich afspeelt. We geven twee voorbeelden van een setting constraint:

Voorbeeld:
- *Een acteur moet op een bepaalde locatie beginnen*
- *Een object moet aanwezig zijn*

De setting constraints zullen tevens bepaalde problemen opwerpen die de agent moet zien te overwinnen.

Voorbeeld:
Het verborgen houden van objecten achter gesloten deuren moet een agent noodzaken om een sleutel te zoeken.

Het invoegen van deze problemen in de setting moet leiden tot creativiteit. Bovendien zullen op deze manier bepaalde plannen van karakters mislukken. Dit is aantrekkelijk voor het verhaal.

Episodische constraints
Episodische constraints zijn de constraints waarbinnen de agents zo autonoom mogelijk kunnen werken. De constraints moeten zo min mogelijk aanwezig zijn en verborgen voor de toehoorder. Episodische constraints zullen bestaan uit acties die niet worden toegestaan in de episode. We geven hieronder een voorbeeld van een episodische constraint:

Voorbeeld:
De prinses mag de schurk niet doden

Episodische constraints zijn aanwezig vanwege de volgende redenen:

- Deze constraints zijn nodig voor de structuur van het verhaal. Bepaalde acties kunnen niet in een te vroeg stadium worden toegestaan. Een karakter dat later nog een heldenrol moet proberen te vervullen, mag niet te vroeg gedood worden.
- Ook kunnen de constraints zorgen voor creativiteit. Door het verbieden van een actie wordt een karakter genoodzaakt om een andere oplossing te zoeken.

Episodische constraints van de laatste episode kunnen doorwerken in de eerste episode. Als een agent nodig is in de laatste episode, dan moet er in de eerste episode automatisch de constraint komen dat de desbetreffende agent niet dood mag.

Voor elke actie die een karakter uitvoert zal hij toestemming moeten vragen aan de Director. Dat is het punt waarop de Director controleert of de actie binnen de episodische constraints uitgevoerd kan worden.

Episodische doelen
Om de functie van de episode daadwerkelijk uit te werken zijn er zogenaamde “Episodische doelen ”. Bepaalde karakters zullen een episodisch doel moeten nastreven. Op deze manier wordt de functie, die in een episode moet worden uitgewerkt, vervuld. Als een episodisch doel behaald is, dan is de episode voorbij en kan er overgegaan worden naar de volgende episode.

Niet elke agent zal een episodisch doel hebben. Alleen de karakters, die belangrijk zijn voor het vervullen van de functie van de episode zullen een episodisch doel krijgen. Dit is belangrijk. Door zo min mogelijk episodische doelen voor te schrijven, is de kans op een verscheidenheid aan verhalen groter. Ook de creativiteit binnen de verhalen wordt groter omdat karakters zelf hun weg bepalen en oplossingen voor problemen kunnen zoeken, zonder dat deze oplossingen vooraf zijn vastgelegd.

Episodische doelen zijn zeker nodig. Het ideaalbeeld laat de acteurs zelf een interessant verhaal maken, maar om de structuur te waarborgen moeten er episodische doelen zijn.

Een ander voordeel van het gebruik van episodische doelen is de verrassing die toegevoegd kan worden. Een karakter zal een episodisch doel moeten vervullen ook al past deze niet bij zijn emoties. Wel moet hierbij in de gaten gehouden worden, dat het verhaal realistisch blijft.

Karakters
Om een functie uit te werken in een episode zijn natuurlijk karakters nodig.

Voorbeeld:
Om de functie "Mission of theHero" uit te werken, is een held nodig

Naast de karakters die nodig zijn om de functie uit te werken, is het interessant om gebruik te maken van meerdere karakters om het verhaal op te fleuren. De 7 karakters die Propp onderscheidt in de Russische sprookjes (paragraaf 3.1.2), lijken uitermate geschikt om bijrollen te vervullen. Zo kan het zijn dat een er een helper aanwezig is, die de held helpt op zijn missie.

Informatie
Agents hebben soms aan het begin van een episode bepaalde informatie nodig om hun episodische doelen na te kunnen gaan streven. Deze informatie kan aan het begin van een episode onthuld worden aan de karakters.

Voorbeeld:
Een held moet bijvoorbeeld weten dat er een magische ring nodig is om een deur te openen. Zodat hij op zoek kan gaan naar de ring

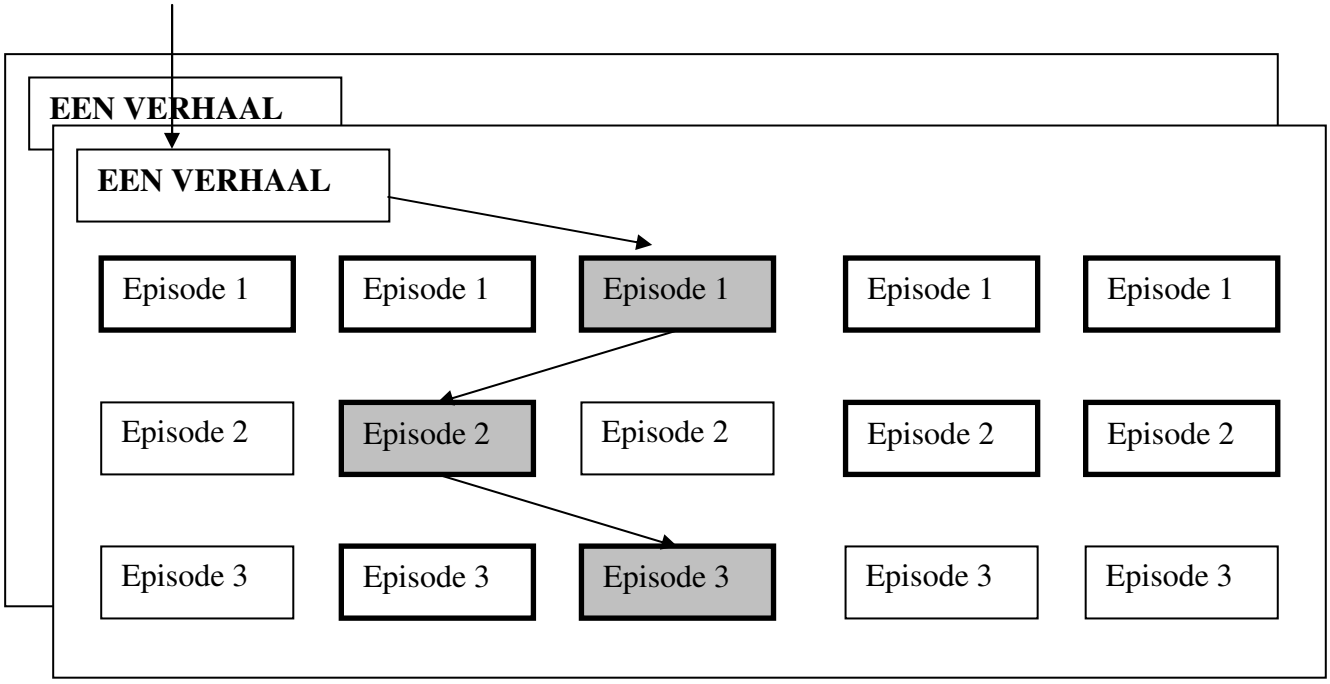
Deze informatie kan een agent simpelweg krijgen van de Director aan het begin van een episode, maar beter is het gebruik van simpele "triggers", zoals een oud mannetje dat toevallig langskomt en de informatie wil vertellen, of een stuk perkament waarop deze informatie staat.

5.2.3 De selectie van episodes

Hoewel er binnen een episode veel ruimte voor variatie aanwezig is, moet er voor echt grote variatie ook verschillende versies van episodes aanwezig zijn. Hoe stellen we ons dit voor?

Zoals te zien in de vorige paragraaf wordt episode4 gegenereerd aan de hand van de uitkomst van episode3. Maar hoe wordt besloten welke episodes eraan vooraf gaan? In figuur 5.3 laten we zien hoe we volgens ons de selectie van de eerste drie episodes zou moeten gaan.

Zoals te zien is in figuur 5.3 zal er eerst een keuze voor een verhaal gemaakt moeten worden (pijl naar het gekozen verhaal). Bij dit verhaal horen bepaalde episodes1. Zodra er een keuze gemaakt is voor episode1 zullen er slechts enkele episodes2 gekozen kunnen worden, omdat niet alle episodes precies op elkaar aansluiten qua episodische doelen. De keuze voor episode3 zal op dezelfde manier gaan als de keuze voor episode2.
De epsiodes waaruit een keuze gemaakt kan worden, hebben in de figuur een dikkere rand. De daadwerkelijk gekozen episodes hebben een grijze achtergrondkleur.



Figuur 5.3 De selectie van episodes: De pijlen geven de keuzes weer.

Het daadwerkelijk selecteren van de episodes zal gedaan kunnen worden door zowel de Director Agent als de gebruiker.

5.2.4 De keuze voor het gebruik van episodes

In de vorige paragrafen is uitgelegd wat we precies onder episodes verstaan en hoe deze werken. In deze paragraaf willen we de keuze voor het gebruik van episodes verder motiveren.

Met het gebruik van episodes worden meerdere doelen nagestreefd:

- Een duidelijke structuur aanbrengen in de verhalen.
- Het verzorgen van een climax.

Deze doelen worden ook nagestreefd door het gebruik van grammatica's. Waarom dan geen grammatica's gebruiken.

- Het grote verschil met een grammatica, is dat de creativiteit van de karakters veel groter kan zijn. Karakters zullen slechts in enkele gevallen een verplicht doel krijgen van de Director en zullen niet constant binnen een grammatica moeten opereren. Dit kan leiden tot verrassingen.
- Een ander voordeel is de moeilijkheidsgraad. We verwachtten dat het gebruik van episodes makkelijker te implementeren is dan een grammatica.
- De architectuur van Sander Faas die aanwezig is, lijkt uitstekend geschikt voor Episodes. Het is voor een Director gemakkelijk om te controleren of een actie van een karakter voldoet aan de episodische constraints. De Director hoeft nu over geen enkele kennis van de grammatica te beschikken.

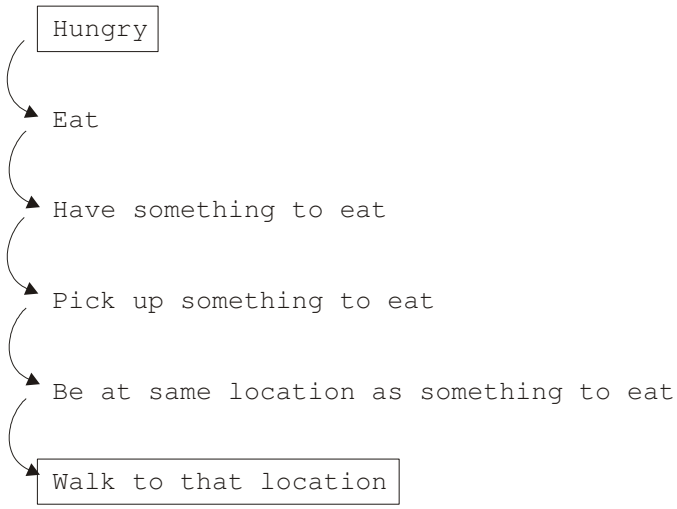
6. Architectuur en geloofwaardige karakters

In dit hoofdstuk wordt er bekeken of de bestaande architectuur van Sander Faas [Fa02] overweg kan met geloofwaardige karakters en wat er eventueel aan de architectuur verbeterd zou moeten worden.

6.1 Reeds aanwezige karakters

Bij de architectuur gemaakt door Faas is er één karakter aanwezig. Dit karakter maakt gebruik van een database met regels en acties. Via "backward chaining" probeert het karakter om zijn doel te bereiken.

Backward chaining is een redeneermechanisme waarbij gekeken wordt naar het doel en vervolgens elke keer een stapje terug geredeneerd wordt, om uiteindelijk bij de actie terecht te komen die het eerst moet worden uitgevoerd. In figuur 6.1 wordt het backward chaining mechanisme uitgelegd met behulp van het doel om de honger te stillen. Een agent die honger heeft, weet dat het effect van de actie eten de honger weg zal nemen. Om de actie eten te kunnen uitvoeren, moet hij echter eerst iets in zijn bezit hebben om te eten. Om iets in zijn bezit te hebben om te eten moet hij echter eerst oppakken dat eetbaar is. Om iets op te pakken dat eetbaar is, moet hij zich echter eerst op een locatie bevinden waar iets ligt, dat eetbaar is. Om zich op een locatie te bevinden waar iets eetbaar ligt, moet hij naar die locatie toelopen. Om zijn doel van honger te stillen te bevredigen zal hij dus eerst de actie “lopen naar een locatie waar wat eetbaars ligt” uit moeten voeren.



Figuur 6.1 Backward chaining van “Hungry”

Voor elke actie die een karakter wil uitvoeren vraagt hij toestemming aan de Director. In de bestaande architectuur geeft de Director nog toestemming voor elke actie. De bedoeling van Faas is, dat als een karakter niet binnen de grammatica bleef, hij geen toestemming kreeg voor de actie. Ook blijft de Director op die manier op de hoogte van de acties die de karakters uitvoeren.

Het karakter heeft op het moment slechts de beschikking over een kleine database met regels en kan dus een beperkt aantal acties uitvoeren. Het karakter krijgt aan het begin zijn doel opgelegd en als hij deze gehaald heeft, is het verhaal afgelopen.

Het karakter dat aanwezig is, is zeer beperkt. Hij streeft wel doelen na, maar waarom hij dat doet weet hij zelf niet. Ook heeft het karakter geen enkele emoties. Hij redeneert simpelweg welke acties hij moet uitvoeren om het doel te bereiken en voert de acties 1 voor 1 uit.

Zoals uit literatuuronderzoek is gebleken (paragraaf 2.9), dat als een karakter geloofwaardig wil overkomen, het gebruik van emoties vereist is. Een emotioneel model voor de karakters moet dus in de architectuur worden opgenomen.

6.2 De emoties

In deze paragraaf beschrijven we, hoe er met emoties moet worden omgegaan in de nieuwe architectuur. Als eerst vertellen we wat over de emoties die aanwezig moeten zijn. Vervolgens vertellen we hoe de emoties gewaardeerd worden.

6.2.1 Aanwezige emoties

Allereerst maken we de keuze om gebruik te gaan maken van emotieparen, net zoals A.j van Kesteren heeft gedaan [Kes01]. We zijn van mening dat het vormen van paren uiterst effectief is. De emoties vormen gevoelsmatig al tegenpolen. Je kunt bijvoorbeeld niet tegelijk trots zijn en je schamen. Dit is dus een afwijking van het OCC model.

Als tweede maken we de keuze om niet alle emoties van het OCC model te implementeren. We zijn van mening dat het niet nodig is om alle emoties te implementeren om geloofwaardige karakters te maken voor een goed verhaal. We beweren niet dat dit juist is, maar zien onze mening onderbouwd door ervaringen van o.a. Burghouts [Bur02] en Reilly [Rei96]. In figuur 6.2 is te zien welke emoties er gebruikt worden:

Positieve emotie	**Negatieve emotie**
Happy for	Pity
Gloating	Resentment
Hope (for self & others)	Fear (for self & others)
Joy	Distress
Pride	Shame
Admiration	Reproach
Love	Hate

Figuur 6.2 Emotieparen in de Virtuele Verhalenverteller

De aanwezige emoties staan gelijk aan de emoties uit het OCC model die niet uit andere emoties voorkomen. De interpretatie van de emoties is iets anders dan in het OCC model. We

hebben de interpretatie overgenomen van het Em model gemaakt door Reilly [Rei96]. De interpretaties staan in figuur 6.3.

De emoties happy for/pity, gloating/resentment, Hope(other)/Fear(other), Admiration/reproach en Love/Hate zijn emoties die een richting hebben t.o.v. een ander karakter.

Voorbeeld:
De held is blij voor de prinses als ze de schurk slaat, en verkneukelt zich over de schurk

De emoties die een richting hebben, worden pas actief op het moment dat een karakter weet dat een ander karakter bestaat.

Happy for	Pity	Je bent blij voor een karakter die je lief hebt/ je hebt medelijden met een karakter die je lief hebt
Gloating	Resentment	Je verkneukelt je over een karakter die je haat/ Je bent zeer afgunstig over een karakter die je haat.
Hope (self)	Fear (self)	Een doel lijkt wel/niet te slagen
Hope (other)	Fear (other)	Je hebt hoop voor iets of iemand/ Je bent bang voor iets of iemand
Joy	Distress	Een doel wordt wel/niet gehaald of komt wel/niet dichterbij
Pride	Shame	De agent voert zelf een actie uit, die wel/niet voldoet aan zijn eigen standaard
Admiration	Reproach	Een andere agent voert een actie uit, die wel/niet voldoet aan de eigen standaard
Love	Hate	Een object/persoon is geliefd of niet

Figuur 6.3 Interpretatie van de emotieparen in de Virtuele Verhalenverteller

6.2.2 Waardes van emoties

In deze paragraaf kijken we allereerst naar de waardes die de intensiteiten van emoties kunnen krijgen. Vervolgens kijken we naar de beïnvloeding van die waardes en als laatste naar de intensiteit van de beïnvloeding.

Toekennen van de waardes
Kijkend naar eerdere werken (SHAME, Em) zien we dat het mogelijk is om een module te maken die alle gebeurtenissen binnenkrijgt en vervolgens de nieuwe emotionele staat berekent. Om dit goed te kunnen doen, moeten aan de emoties een bepaalde waarde worden toegekend. We kiezen ervoor om elke emotiepaar (zie figuur 6.2) een waarde te geven van -100 tot +100. Als de waarde van een emotiepaar positief is, wordt de positieve emotie ervaren en als de waarde negatief is, wordt de negatieve emotie ervaren.

Voorbeeld:
Een waarde van +20 voor joy/distress betekent dat een karakter joy ervaart met een waarde van +20. Er wordt geen distress ervaren.

Beinvloeding van de intensiteit
De intensiteit van de emoties zullen worden beïnvloed door gebeurtenissen. We maken hierbij onderscheid tussen twee soorten gebeurtenissen:

- Gebeurtenissen die van buitenaf plaatsvinden. Hieronder vallen de gebeurtenissen in de wereld. Aan elke gebeurtenis in de wereld moet een emotionele waarde worden toegekend:

  Voorbeeld:
  Gebeurtenis: *Het zien van een heks*
  Waardetoekenning: *Hope/Fear(Heks) -20*
  Joy/Distress -30
  Deze interpretatie van alle gebeurtenissen samen vormen de invloed van buitenaf op de emoties.
- Gebeurtenissen van binnenuit. Binnen in een karakter vinden ook processen plaats. Deze kunnen ook zijn emoties beïnvloeden:

  Voorbeeld:
  Gebeurtenis *Een karakter denkt dat hij zijn doel gaat halen*
  Waardetoekenning: *Hope/Fear (Self) +20*

In de architectuur moet dus plaats zijn voor een module die de gebeurtenissen van zowel binnenuit als buitenaf een waarde toekent.

Intensiteit van beïnvloeding

De intensiteit van de beïnvloeding van de emoties verschilt per persoon. Hier onderscheiden we net zoals in Em en Shame twee eigenschappen van emoties:

- De sterkte van de emotie:

  Voorbeeld:
  Een klein meisje schrikt heel erg van voetstappen op de gang van een hotel, terwijl haar ouders zich er nauwelijks druk om maken

- De duur van de emotie:

  Voorbeeld:
  Het kleine meisje blijft wel een half uur bang voor de voetstappen op de gang, terwijl haar broertje het na vijf minuten alweer vergeten is.

In de architectuur moet de intensiteit van de beïnvloeding van emoties verschillen per persoon.

6.3 Action Tendencies

Als de emoties eenmaal berekend zijn, hebben we een vector van emotieparen met bijbehorende waardes. De vraag is nu, wat er verder met deze emoties moet gebeuren.

We willen hier graag hetzelfde systeem toepassen, dat Reilly in zijn Em systeem gebruikt. Hij gebruikt de emoties voor het toekennen van waardes aan gedragskenmerken. Hij noemt deze gedragskenmerken “Emotional Behaviour”. Wij prefereren de naam “Action Tendency”. Deze naam wordt namelijk ook al gebruikt door Freijda [Fre86] en ook hier dient het als een soort “mapping” van emoties naar acties en doelen.

Deze “action tendencies” kunnen vervolgens gebruikt worden om te bepalen welk doel of actie er gekozen wordt. Voor de Virtuele Verhalenverteller willen we gebruik maken van de action tendencies die te zien zijn in figuur 6.4.

Action Tendency	Betekenis
Vriendelijk	Het vriendelijk gedragen
Vriendelijk (Karakter)	Het vriendelijk gedragen t.o.v. een ander karakter.
Onvriendelijk	Het onvriendelijk gedragen.
Onvriendelijk (Karakter)	Het onvriendelijk gedragen t.o.v. een ander karakter.
Passief	Het karakter gedraagt zich passief/bangig.
Passief (Karakter)	Het passief/bangig gedragen t.o.v een ander karakter.
Offensief (Karakter)	Het offensief gedragen t.o.v. een ander karakter.
Superieur	Het karakter gedraagt zich superieur.
Superieur (Karakter)	Het karakter gedraagt zich superieur t.o.v een ander karakter.
Paniekerig	Het karakter gedraagt zich paniekerig.
Agressief	Het karakter gedraagt zich agressief t.o.v. een ander karakter.
Vrolijk	Een karakter gedraagt zich vrolijk

Figuur 6.4 Action Tendencies in de Virtuele Verhalenverteller

Wat ons vooral aanspreekt is de mogelijkheid, om de gebruiker te laten bepalen hoe de waarde van een action tendency gemaakt wordt. Hier kunnen simpele scripts voor gebruikt worden (zie figuur 4.14).

Het moet dus binnen de architectuur mogelijk zijn om gebruik te maken van scripts, die gebruik maken van emoties om de waardes van zogenaamde “Action Tendencies” te bepalen.

6.4 Doelen

Karakters zullen bepaalde doelen hebben die ze proberen na te streven. In deze paragraaf beschrijven we eerst welke doelen er aanwezig zijn, vervolgens hoe deze doelen een waarde krijgen toegekend en als laatste hoe de doelen worden gekozen.

6.4.1 Aanwezige doelen

Karakters zullen bepaalde doelen hebben die ze proberen na te streven. In figuur 6.5 ziet u de doelen waaruit een karakter in de Virtuele Verhalenverteller kan kiezen (agent betekent in het figuur een ander karakter).

Doel (omschrijving)	**Doel (feitelijk)**
Zijn bij(agent)	Me.location == Agent.location
Aanvallen(disliked agent)	Decreasing disliked Agent.Strength
Gevangen nemen(disliked agent)	Disliked agent.location == building && building.door == locked
Vluchten van (agent)	Me.location != agent.location
Roepen om hulp	Scream
Ontdekken wereld	Unknown paths == none
Random bewegen	Me.location != currentLocation
Zingen	Sing
Aanvallen(willekeurige agent)	Decreasing Agent.Strength
Sterker worden(objects om sterker te worden zoeken)	Increasing Me.Strength
Helpen(agent)	Agent.doel succeeds
Doden(agent)	Agent.strength == 0

Figuur 6.5 Doelen in de Virtuele Verhalenverteller

We kiezen ervoor om elke agent over dezelfde, vaste doelen te laten beschikken. Dit benadert de reële wereld het best. Iedereen kan uit dezelfde doelen kiezen, maar of ze daadwerkelijk nagestreefd worden hangt van de persoon af.

6.4.2 Waardes van een doel

Om een keuze te maken uit de alle aanwezige doelen moeten er aan de doelen waardes worden toegekend. We maken hierbij gebruik van de “Action Tendencies”. De belangrijkheid van een doel wordt gekoppeld aan de belangrijkheid van 1 of meerdere “Action Tendencies”. Een voorbeeld van hoe dit zou kunnen gaan ziet u in figuur 6.6.

Doel (omschrijving)	**Belangrijkheid**
Zijn bij(agent)	Vriendelijk(agent)
Aanvallen(disliked agent)	Onvriendelijk(agent) + Superieur(agent)
Gevangen nemen(disliked agent)	Onvriendelijk(agent) + Superieur(agent)
Vluchten van (agent)	1.5 * Passief
Roepen om hulp	0.2 * Passief + Paniekerig
Ontdekken wereld	Superieur
Random bewegen	Paniekerig
Zingen	Vrolijk
Aanvallen(willekeurige agent)	Agressief + Paniekerig
Sterker worden(objecten om sterker te worden zoeken)	100 - Me.sterkte
Helpen(agent)	Vriendelijk(agent)
Doden (agent)	Offensief(agent) + Superieur(agent) + Aggressief

Figuur 6.6 Voorbeeld van hoe de belangrijkheid van een doel bepaald wordt.

Het gebruiken van de “Action Tendencies” om de waarde van een doel te bepalen lijkt sterk op het mechanisme dat Burghouts gebruikt in zijn implementatie (zie paragraaf 4.4.2).

De belangrijkheid van een doel zal uitgedrukt worden in een getal tussen de 0 en de 100. Heeft een doel een belangrijkheid van 0, dan zal hij totaal niet belangrijk zijn, maar als het een waarde heeft van +100 is het zeer belangrijk. De volgende paragraaf zal verder in gaan op het daadwerkelijk kiezen van een doel.

6.4.3 Het kiezen van een doel

De keuze van het doel dat nagestreefd wordt, zal afhangen van twee dingen:

- de belangrijkheid van het doel (zie paragraaf 6.4.2) .
- aanwezigheid episodisch doel (zie ook paragraaf 5.2.2).

Afwezigheid episodisch doel
Als er geen episodische doelen aanwezig zijn, dan gaan we ervan uit dat alleen de belangrijkheid van een doel van invloed is op de keuze van het doel. De belangrijkheid van een doel wordt bepaald door de waardes van de “Action Tendencies” (paragraaf 6.4.2).

We zouden gewoon een keuze kunnen maken op grond van belangrijkheid. We zouden dan gewoon het belangrijkste doel kunnen gaan nastreven. Dit maakt de keuze voor de doelen echter erg voorspelbaar. Door een semi-willekeurige keuze te maken zal de keuze van een doel verrassingen kunnen brengen. Voor een semi-willekeurige keuze nemen we de belangrijkheid van een doel als de kans dat het gekozen wordt.

Voorbeeld: We hebben doel X Y en Z met de belangrijkheden: 20, 50, 60. De kans dat X gekozen word is nu 20/(20+50+60) = 20/130. De kans dat Y gekozen word is 50/130 en van Y 60/130.

Oncontroleerbare emoties
We zijn het eens met Burghouts [Bur02], dat er een vorm van zelfcontrole is op de emoties. Bij zelfcontrole worden op het ervaren van emoties functies losgelaten waardoor bepaalde emoties worden onderdrukt of juist versterkt.

In de Virtuele Verhalenverteller zullen we dit iets anders tot uiting brengen. We laten de intensiteit van de emoties ongemoeid, maar zullen de invloed van de emoties op de doelen gebruiken. De beïnvloeding van de emoties op de doelen gebeurt via "Action Tendencies" (paragraaf 6.4.2). Indien de intensiteit van een emotie hoog oploopt, zal via de "Action tendencies" de waarde van belangrijkheid van één of meerdere doelen hoog op kunnen lopen. We willen nu een vorm van zelfcontrole bewerkstelligen door het invoeren van een grenswaarde. Indien de belangrijkheid van een doel onder deze grenswaarde ligt, dan zullen de emoties nog onder controle zijn en zal de keuze van een doel op de normale wijze gebeuren. Stijgt de waarde van belangrijkheid echter boven deze grenswaarde uit, dan zijn de emoties dusdanig hoog, dat het karakter ze niet langer zelf kan controleren. Het doel dat een dusdanige hoge waarde krijgt zal dan dus ook de absolute prioriteit krijgen. Naar andere "normale" doelen zal niet langer gekeken worden. Er zal indien er een doel met een waarde boven de grenswaarde is, dus ook geen semi-willekeurige keuze zijn tussen doelen.

Zijn er echter meerdere doelen met een belangrijkheid boven de grenswaarde, dan zal er de keuze moeten gemaakt worden tussen die doelen op een geheel willekeurige basis.

Aanwezigheid episodisch doel
Een karakter kan een episodisch doel opgelegd krijgen door de Director (zie paragraaf 5.2.2). Dit doel moet echter wel worden nagestreefd door het karakter. Het is mogelijk om het karakter dit episodische doel altijd na te laten streven. In dat geval zijn echter de emoties totaal niet meer van belang voor dat karakter. We kiezen ervoor om doelen met een belangrijkheid, boven de grenswaarde voor oncontroleerbare emoties, voorrang te verlenen boven het episodische doel. Het eventueel aanwezige episodische doel krijgt op zijn beurt weer voorrang boven de overige doelen.

We zijn het eens met Kline en Blumberg [Kli99] dat, om een karakter geloofwaardig over te laten komen, hij constant door moet werken aan het bereiken van zijn doel en netjes moet omgaan met onverwachte situaties. Een karakter moet dus niet om de haverklap een nieuw doel adopteren. Het uitvoeren van het huidige doel krijgt dus voorrang boven het episodische doel, maar blijft minder belangrijk dan een doel met een belangrijkheid boven de grenswaarde voor oncontroleerbare emoties.

Algoritme

Uit het bovenstaande valt het volgende te concluderen:

- als een doel een belangrijkheid heeft boven een grenswaarde, dan zijn de emoties van de agent oncontroleerbaar en heeft het doel de hoogste prioriteit.
- indien er meerdere doelen zijn met een belangrijkheid boven de grenswaarde, moet er een random keuze tussen hen gemaakt worden.
- Heeft het karakter al een doel dat hij bezig is te behalen, dan heeft dit een hogere prioriteit dan het episodische doel.

- indien er een episodisch doel aanwezig is, heeft het de hoogste prioriteit als er geen doelen zijn met een belangrijkheid boven de grenswaarde of als de acteur reeds een doel heeft dat hij bezig is te behalen.
- als er geen doelen zijn die een waarde van belangrijkheid hebben boven de grenswaarde en er is geen episodisch doel aanwezig, dan moet er een semi-willekeurige keuze (op basis van belangrijkheid) worden gemaakt tussen de doelen.

De bovenstaande eisen kunnen worden verwerkt in een algoritme. Dit algoritme is te zien in figuur 6.7.

```
Public Goal selectGoal(Goals[] goals, Goal currentGoal,
Goal episodicGoal)
{

     Goal chosenGoal;
     Enumeration uncontrolableGoals;
     Enumeration controlableGoals;

     for( int i =0; i<goals.length; i++)
     {
          /* als een goal boven de grenswaarde uitkomt
stop hem
          dan in de lijst uncontrolableGoals en anders in
de lijst
          contrableGoals
          */
          if (goals[i] > GRENS)
          {
               addGoal(uncontrolableGoals,goals[i]);
          }
          else
          {
               addGoal(controlableGoals,goals[i]);
          }
     }
     /* als er oncontroleerbare goals zijn dan
     wordt de gekozen goal een goal uit die lijst
     */
     if(checkEmpty(uncontrolableGoals) == false)
     {
          chosenGoal ==
ChooseRandomGoal(uncontrolableGoals);
     }
```

```
     /*zijn er geen oncontroleerbare goals en is er nog
     geen goal in behandeling, dan wordt de goal de
episodic goal, of
     een goal uit de lijst van controleerbare goals
     */
     else if (currentGoal == null)
     {
          if (episodicGoal != null)
          {
               chosenGoal == episodicGoal;
          }
          else
          {
               chosenGoal == ChooseGoal(controlableGoals);
          }
     }
     else
     {
          chosenGoal == currentGoal;
     }
     return chosenGoal;
}
```

Figuur 6.7 Algoritme voor het kiezen van een doel.

6.5 Actieselectie

Zodra er een doel gekozen is, moet er een actie worden gekozen. Dit noemen we actieselectie. In deze paragraaf beschrijven we hoe een geloofwaardig karakter een keuze zou kunnen maken voor een actie, om zo zijn doel dichterbij te brengen.

6.5.1 De Acties

Een actie zorgt voor de overgang van de ene toestand naar de andere toestand. Een actie kan elementair zijn (het lopen van de ene locatie naar de andere), dan wel erg ingewikkeld (het gevangen nemen van iemand). We geven een voorbeeld van hoe een actie eruit kan zien:

```
WalkTo(agent, location1,location2)

Preconditie:   (1)  agent.location == location1;
               (2)  Adjacent(Location1,location2);

Effect:        (3)  agent.location == location2
```

Figuur 6.8 de Actie WalkTo

In figuur 6.8 ziet u een voorbeeld van de actie “WalkTo”. Het karakter (agent) wil lopen van locatie1 naar locatie2. Om dit te mogen doen, moet hij aanwezig zijn op locatie1 (1) en tevens moet locatie1 naast locatie2 liggen (2). Het effect van de actie zal zijn dat de nieuwe locatie van de agent locatie2 is (3).

Voor de Virtuele Verhalenverteller hebben we in eerste instantie gekozen voor de aanwezigheid van de acties, die te zien zijn in figuur 6.9. Met deze acties moet het volgens ons mogelijk zijn om de vier functies uit paragraaf 5.2 op verschillende manieren te beschrijven.

```
Walk        Hit
Enter       Strangle
Give        Stab
Take        Scream
PickUp      Sing
Eat         Imprison
Kick
```

Figuur 6.9 De aanwezige acties in de Virtuele Verhalenverteller

Er moet bij de implementatie van de architectuur echter wel rekening worden gehouden met de mogelijke uitbreiding van het aantal acties.

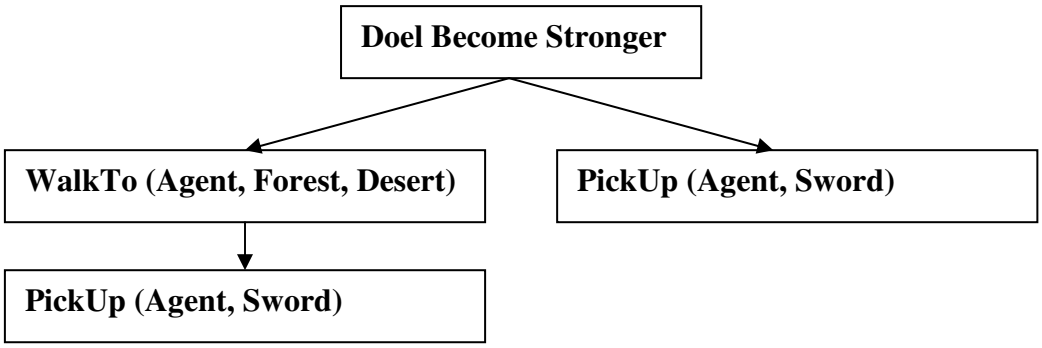
6.5.2 Planning

Om te bepalen welke actie gekozen zal worden moet een karakter eerst een plan opstellen om zijn doel te behalen. Hij zal hiervoor het wereldmodel moeten gebruiken. In figuur 7.8 ziet u een voorbeeld van een wereldmodel. Het karakter bevindt zich samen met een zwaard in het bos. In de nabijgelegen woestijn ligt ook een zwaard.

```
Location ( Agent, Forest)
Location ( Sword, Forest)
Location ( Sword, Desert)
Adjacent ( Forest, Desert)
Adjacent ( Desert, Forest)
```

Figuur 6.10 Een klein wereldmodel

Een karakter kan met behulp van een wereldmodel en acties een plan opstellen om een doel te behalen. In figuur 6.11 ziet u dat de agent voor de doel “Become Stronger” met het wereldmodel uit figuur 6.10 twee plannen kan opstellen.



Figuur 6.11 Planning in de Virtuele Verhalenverteller

De twee plannen zijn dus:

Plan1: WalkTo(Agent, Forest, Desert), PickUp(Agent,Sword)
Plan2: PickUp(Agent,Sword)

Voor het plannen wordt gebruik gemaakt van de precondities en de effecten van acties. We leggen met behulp van figuur 6.12 uit hoe plan1 gemaakt wordt.

```
(1)Become Stronger [Have(Agent,Sword)]
(2)➔PickUp(Agent,Sword)
(3)  Effect: Have(Agent,Sword)
(4)  Preconditie: Location (Agent, Desert)
(5)➔ WalkTo(Agent, Forest, Desert)
(6)  Effect: Location (Agent, Desert)
(7)  Preconditie: Location (Agent, Forest)
(8)  Preconditie: Adjacent(Forest, Desert)
```

Figuur 6.12 Het maken van een plan

Het doel “Become Stronger” wordt eerst vertaald naar het in bezit hebben van het zwaard (1). Zoals in figuur 6.10 te zien is, heeft het karakter geen zwaard in zijn bezit. Immers, er staat in figuur 6.10 nergens Have (Agent, Sword) wat staat voor het in bezit hebben van een zwaard door het karakter. Het planningsmechanisme ziet nu dat er geprobeerd moet worden om een zwaard in bezit te hebben. Het mechanisme ziet dat de actie oppakken (2) het effect heeft dat het zwaard in bezit komt (3). Nu zal geprobeerd worden om de preconditie van deze actie te vervullen. De preconditie om het zwaard op te pakken, is dat het karakter op dezelfde locatie moet zijn als het zwaard(4). Het mechanisme ziet vervolgens dat het lopen naar de woestijn(5) dat effect heeft (6). Vervolgens ziet het planningsmechanisme dat de precondities voor het lopen naar het bos (7 & 8) al aanwezig zijn (zie figuur 6.10).

6.5.3 Actiekeuze en Planning

Zodra het planningsmechanisme een actie moet selecteren, zal het planningsmechanisme alle mogelijke plannen, behorende bij het doel, maken. Vervolgens zal het plan met de minste stappen gekozen worden (in Figuur 6.11 dus het rechterplan).

Het lijkt niet logisch om, als er een tweede actie gekozen moet worden om hetzelfde doel te bereiken, alle plannen opnieuw te maken en dan het plan met de minste stappen te kiezen. Dit heeft echter één groot voordeel. Mocht er in de tijd tussen twee acties iets verandert zijn in het wereldmodel, dan zal het karakter automatisch zijn plan veranderen. We geven hier een voorbeeld van:

Voorbeeld:
Een agent heeft als preconditie het verkrijgen van een wapen. Hij weet dat er een wapen in het bos ligt. Hij moet eerst naar het meer om bij het bos te komen. Als hij bij het meer is, ziet hij een helper die hem vertelt dat het wapen niet meer in het bos ligt. Het zou stom zijn om nu toch nog naar het bos te lopen, terwijl hij al weet dat het wapen daar niet ligt (ervan uitgaande dat er niet gelogen word).

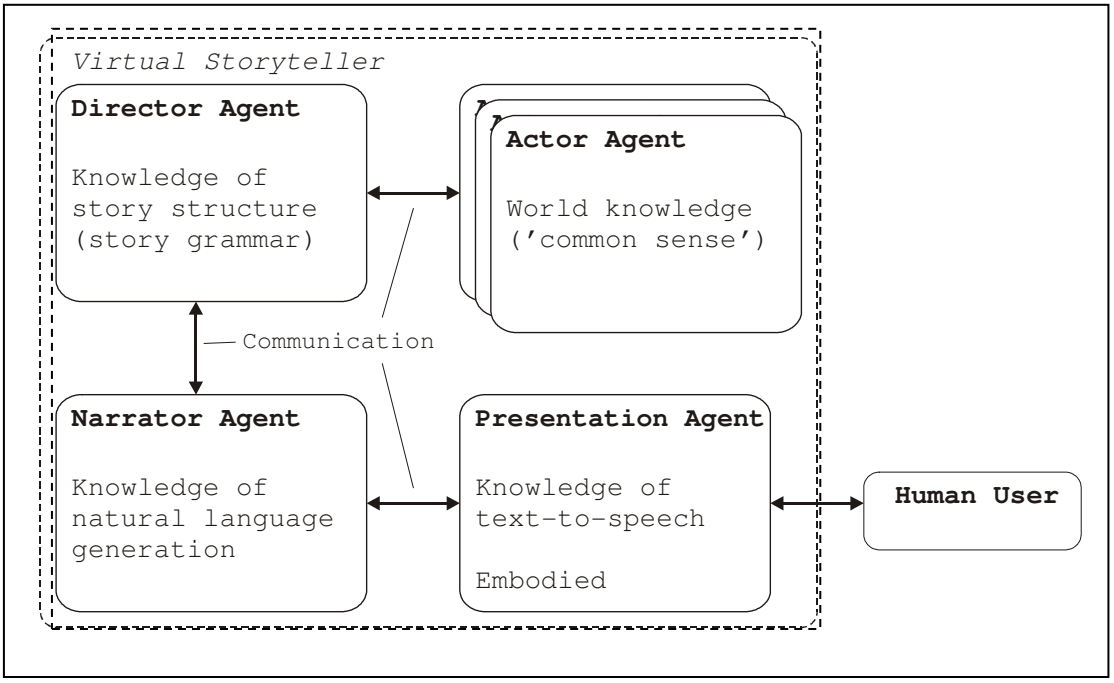
Mochten er twee plannen aanwezig zijn met even weinig stappen, dan zal een karakter random moeten kiezen tussen deze twee.

7. De nieuwe Architectuur

In de hoofdstukken 5 en 6 is gekeken hoe de bestaande architectuur gebruikt zou kunnen worden om aan de eisen verkregen uit literatuuronderzoek te voldoen.

Nu duidelijk is waar de nieuwe architectuur aan moet voldoen, kunnen we de nieuwe architectuur maken.

De nieuwe architectuur zal gebruik blijven maken van de architectuur van de bestaande Virtual Storyteller van Faas [Fa02] (zie figuur 7.1).



Figuur 7.1 De bestaande architectuur van Sander Faas

Elk onderdeel van de architectuur is een agent. Maar wat is eigenlijk een agent? Een agent wordt door [Rus95] gedefinieerd als:

"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that envirmonment through effectors. Furhtermore an agent has to be autonomous meaning that the agent decides on his own actions and that the behavior depends at least partly on the information perceived by means of the sensor"

Een agent is dus een autonoom computerprogramma dat reageert op gebeurtenissen in zijn omgeving.

Ook in de nieuwe architectuur zullen we gebruik maken van deze agents. Wel zullen de verschillende agents enkele duidelijke veranderingen ondergaan. In dit hoofdstuk zullen we voor alle onderdelen (Director, Actor (een karakter), Narrator (de verteller) en Presentation Agent) een architectuur presenteren en bespreken.

7.1 De Director Agent

De Director agent is het onderdeel van het systeem, dat via interactie met de gebruiker bepaalt welk soort verhaal er moet worden gemaakt en verteld. De Director is verantwoordelijk voor de communicatie met de karakters en de verteller.

In deze paragraaf zullen we eerst weergeven over welke kennis de Director bezit. Vervolgens kijken we welke acties de Director precies moet uitvoeren en als laatste presenteren we een architectuur voor het uitvoeren van deze acties.

7.1.1 Aanwezige kennis

Om zijn taken uit te voeren beschikt de director over kennis. We kunnen deze kennis splitsen in 5 onderdelen:

- Episodische doelen (zie paragraaf 5.2.2)
- Episodische constraints (zie paragraaf 5.2.2)
- Wereldmodel. De Director weet als enige excact hoe de wereld eruitziet.
- Karakterinformatie. De Director weet alle eigenschappen van de karakters.
- Standaard constraints. Onder standaard constraints verstaan we de algemene kennis die de Director heeft over verhalen. Een Director moet bijvoorbeeld weten dat het niet interessant is voor een verhaal als een karakter tien keer achter elkaar dezelfde actie uitvoert.

De Director zal deze kennis kunnen gebruiken om zijn acties uit te voeren. We beschrijven deze acties in de volgende paragraaf.

7.1.2 Uit te voeren acties

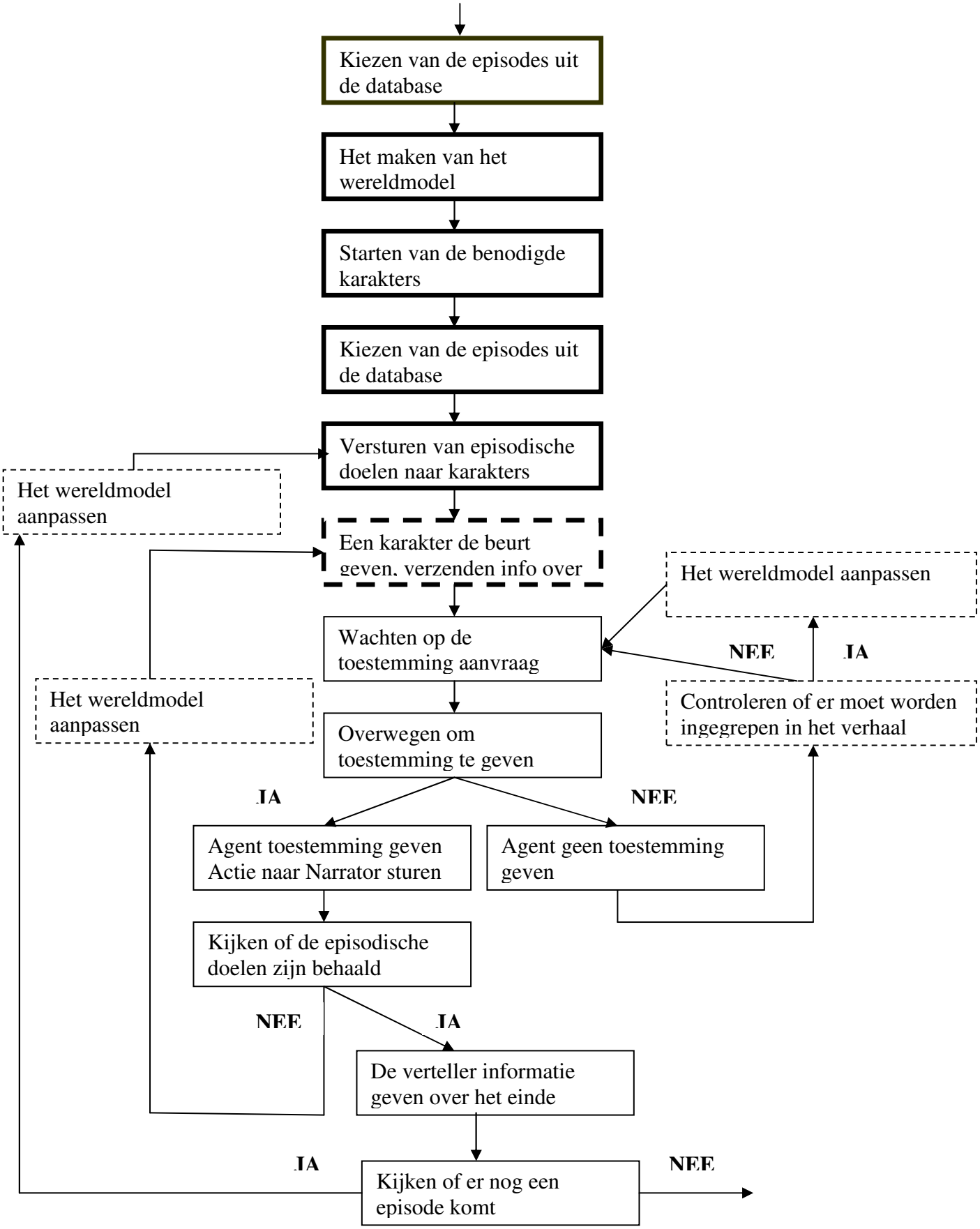
Aan de hand van de informatie die in hoofdstukken 5 en 6 wordt weergegeven is het mogelijk alle acties die de Director zou moeten uitvoeren weer te geven in een figuur (zie figuur 7.2)

De pijltjes in figuur 7.2 geven aan welke acties er na elkaar uitgevoerd moeten worden. Indien er twee mogelijkheden zijn, staat er bij de pijltjes in welk geval er welke actie uitgevoerd moet worden.

We delen de acties in vier groepen in:

- Het vergaren van informatie voor het verhaal (dikke ononderbroken lijn)
- Het initialiseren van een karakter voor een beurt (dikke stippellijn)
- Het verwerken van de aanvraag van een actie (dunne ononderbroken lijn)
- Aanpassen eigen Wereldmodel (dunne stippellijn)

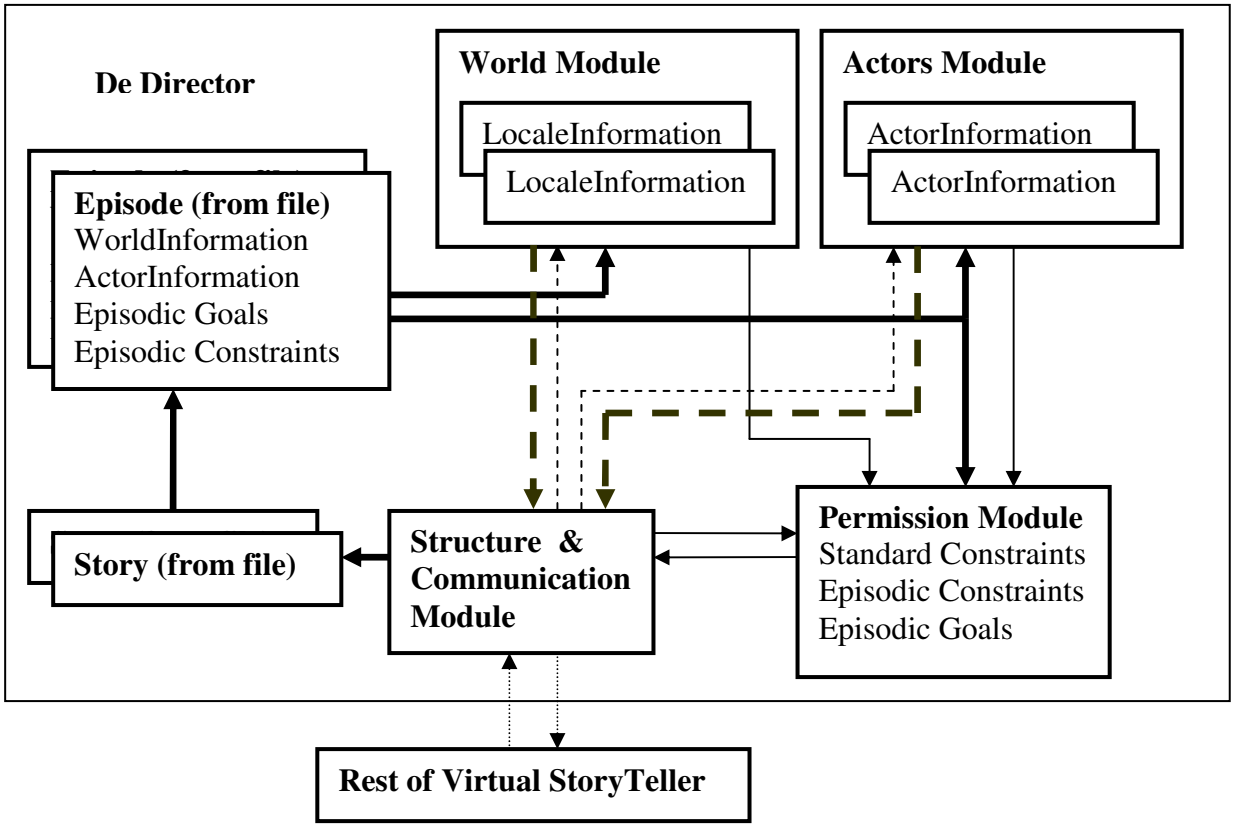
In de volgende paragraaf presenteren we een architectuur voor het uitvoeren van deze acties.



Figuur 7.2 De acties van de Director

7.1.3 Architectuur

Aan de hand van de acties die de Director moet uitvoeren (zie figuur 7.2) hebben we een architectuur gemaakt (zie figuur 7.3). De lijnen van de pijlen uit figuur 7.3 komen overeen met de lijnen van de acties uit figuur 7.2.



Figuur 7.3 De Director

We geven even kort weer wat elk onderdeel inhoudt:

- De Story zal bestaan uit een file met daarin informatie over de aanwezige episodes. Om uit meerdere soorten verhalen te kunnen kiezen zullen er meerdere Story files aanwezig moeten zijn.
- Per Story zullen er meerdere Episodes aanwezig zijn. Deze bevatten alle informatie over een episode.
- De World Module zal de informatie over de wereld bevatten. Over elke locatie zal bekend zijn wie of wat zich er bevindt en wat de aansluitende locaties zijn.
- De Actors Module zal informatie bevatten over de karakters. Per karakter zullen zijn eigenschappen, zijn eventuele episodische doel, zijn locatie en de gedane acties worden opgeslagen.
- De Permission Module zal het onderdeel zijn dat kijkt of een actie toegestaan is.
- De Structure & Communication module moet zorgen voor de communicatie met de rest van het systeem en houdt tevens bij welk karakter er aan de beurt is.

We geven ook nog even weer hoe de acties overeenkomen met de architectuur. We doen dit door middel van het volgen van de chronologische acties.

Het vergaren van informatie voor het verhaal (dikke ononderbroken lijn).
Om een verhaal te vertellen moet de Director, al dan niet geholpen door de gebruiker via een interface, een verhaal uitkiezen, met de daarbij behorende episodes. Deze episodes zullen informatie bevatten (zie paragraaf 5.2.2). Deze informatie zal eenmalig, bij het begin van een episode doorgestuurd worden naar de WorldModule, de ActorsModule, de Pemission Module.

Een karakter de beurt geven (dikke stippellijn)
De Structure & Communication zal van de ActorsModule doorkrijgen welk karakter er aan de beurt is. De Structure & Communication zal dit doorgeven aan het desbetreffende karakter samen met de benodigde informatie over de locatie van het karakter (LocaleInformation) en het karakter zelf (ActorInformation).

Het verwerken van de aanvraag voor een actie (dunne ononderbroken lijn)
Zodra een karakter, dat aan de beurt is, een actie wil uitvoeren zal het toestemming vragen aan de Director. De Permission Module zal deze actie binnen krijgen via de Structure & Communication module en zal vervolgens bekijken of de actie wordt toegestaan. Het overwegen om toestemming te geven zal gebeuren met behulp van informatie die reeds verkregen is bij het starten van de episode, met informatie afkomstig van het World Model en met informatie van het Actors Model. Het karakter zal vervolgens al dan niet toestemming krijgen.

Het aanpassen van het eigen wereldmodel (dunne stippellijn)
Zodra een actie uitgevoerd wordt door een karakter zal de Structure & Communications module het eigen wereldmodel aanpassen door informatie te sturen naar de World Module en de Actors Module. Het kan ook zijn dat er geen toestemming gegeven kan worden voor een actie, voordat de wereld aangepast wordt door de Director.

7.2 Een Actor agent

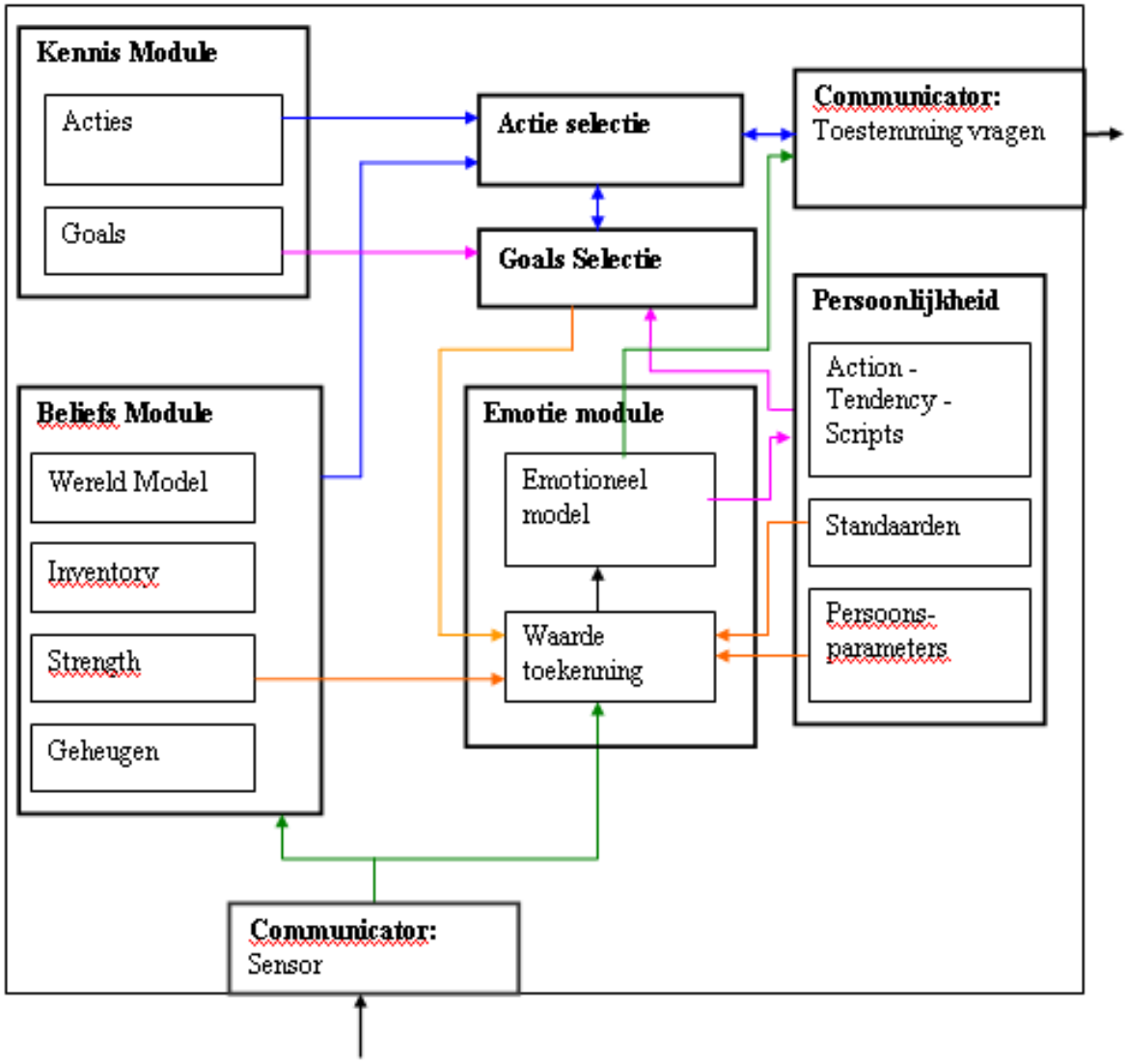
Een Actor agent is het onderdeel van het programma dat één karakter vertegenwoordigd. Voor het vertellen van het verhaal zijn vaak meerdere karakters nodig. Er zal dan voor elk karakter dus één Actor agent aanwezig zijn. In hoofdstuk 6 is reeds beschreven waar een geloofwaardig karakter aan moet voldoen. In de volgende paragrafen wordt een complete architectuur beschreven voor een Actor agent.

7.2.1 De architectuur

In figuur 7.4 staat de architectuur voor de Actor agent (de kleuren van de pijlen in de figuur zijn slechts voor de duidelijkheid en hebben geen aparte betekenis). Zoals te zien bestaat de nieuwe architectuur uit zeven onderdelen (De Communicator bestaat in de figuur uit twee delen). We zullen elk onderdeel even kort behandelen.

De communicator
De communicator is het onderdeel dat communiceert met de Director.

- Allereerst zorgt de communicator voor het ontvangen van de gebeurtenissen in de wereld (events) van de Director. Deze events geeft hij door aan het onderdeel voor de emotie waardering en naar de beliefsmodule.
- De communicator heeft als tweede functie het vragen van toestemming voor het uitvoeren van een actie. Hij stuurt hierbij ook informatie over de emoties mee, zodat er over de emoties kan worden verteld door de verteller (Narrator).



Figuur 7.4 De architectuur voor een Actor agent (de kleuren hebben geen betekenis)

De Emotie Module

De emotie module bestaat uit twee onderdelen:

- Waarde toekenning: Dit onderdeel zorgt ervoor dat er aan binnenkomende events en de vorderingen betreffende de doelen een emotionele waarde wordt toegekend . Deze emotie waardering maakt hierbij gebruik van de sterkte van het karakter, de standaarden en de persoonsparameters. Zie hiervoor ook de paragrafen 6.2.2. en 7.2.2.
- Emotioneel model: Het emotionele model bevat alle emoties (zie paragraaf 6.2.1). Deze emoties veranderen onder invloed van de informatie afkomstig van het Emotie waardering onderdeel. Zie ook paragraaf 6.2.2.

De Beliefs Module
In de beliefsmodule staan alle overtuigingen van de agent. Deze overtuigingen kunnen veranderen door de binnenkomende events. De onderdelen zijn:

- wereldmodel: de overtuiging van de agent over hoe de wereld eruitziet.
- de inventory: de objecten die een agent bij zich heeft.
- de sterkte: de sterkte van de agent.
- het geheugen: alle acties waarvan hij weet dat ze zijn uitgevoerd.

Zoals te zien is in de architectuur, wordt de sterkte van de agent gebruikt in het emotionele model.
Alle opvattingen van de agent worden bovendien gebruikt voor de actie selectie.

De kennismodule
De kennismodule bevat alle algemene kennis die een agent heeft. De kennismodule bestaat uit de onderdelen acties en doelen. De agent heeft de kennis over hoe hij bepaalde acties moet uitvoeren en hoe hij er doelen mee kan bereiken. De doelen worden gebruikt voor de doelen selectie en de acties worden gebruikt voor de actie selectie.

De Goals selectie
De Goals selectie is verantwoordelijk voor het selecteren van de doelen. Dit gebeurt onder invloed van de doelen, de “Action Tendency Scripts” en de emoties. Zie hiervoor ook paragraaf 6.4.

Actie selectie
Deze module is verantwoordelijk voor het selecteren van de actie behorende bij het gekozen doel. Dit gebeurt onder invloed van de kennis over acties, de beliefs van de agent en het geselecteerde doel. Meer hierover kunt u vinden in paragraaf 6.5. De actie selectie zal voordat het daadwerkelijk een actie uitvoert, eerst toestemming moeten vragen aan de Director via de communicator. Indien hij geen toestemming krijgt zal hij moeten proberen om een ander plan uit te voeren. Indien dit niet kan, moet hij de doelen selectie duidelijk maken, dat er een nieuw doel moet worden gekozen.

De persoonlijkheid
Deze module bevat alle onderdelen die een karakter persoonlijkheid geven. Meer hierover kunt u lezen in paragraaf 7.2.2.

7.2.2 Persoonlijkheid

In hoofdstuk 6 is beschreven welke onderdelen een architectuur zal moeten bevatten om een karakter, dat volgens de architectuur wordt gemaakt, geloofwaardig over komt. Hier willen we echter nog even duidelijk uitleggen waar het verschil tussen de karakters precies aanwezig zal zijn.

De module die voor de daadwerkelijke vorming van het karakter zorgt, is de persoonlijkheid (zie figuur 7.4). De persoonlijkheid module bestaat uit drie onderdelen:

- Persoonsparameters
- Standaarden
- Action Tendency Scripts

Persoonsparameters
Zoals te lezen valt in paragraaf 6.2.2 zijn er twee eigenschappen die een emotie beïnvloeden:

- De sterkte van de emotie:
- De duur van de emotie:

Elke gebeurtenis (van binnenuit en buitenaf zie paragraaf 6.2.2) krijgt een emotionele waarde toegekend. De persoonsparameters moeten de waardering van emoties per persoon laten verschillen. We zullen een voorbeeld geven om dit duidelijk te maken:

Voorbeeld:
*Een held heeft een parameter van 1 voor Hope/Fear en een prinses een parameter van 3. Stel er komt nu een perceptie binnen dat er een schurk op dezelfde locatie komt. Dit zal normaal gesproken de waarde van de emotie hope/fear veranderen met een waarde van -15. Door de parameters verandert de waarde van de hope/fear emotie bij de held slechts met -15 * 1 = -15. Bij de prinses zal de waarde van de hope/fear emotie veranderen met maar liefst -15 * 3 = -45.*

De persoonsparameter zal naast de sterkte van de emotie ook van invloed zijn op de duur van de emotie.

We proberen dit duidelijk te maken met behulp van hetzelfde voorbeeld:

Voorbeeld:
Een held heeft een parameter van 1 voor Hope/Fear en een prinses een parameter van 3. Stel er komt nu een perceptie binnen dat er een schurk op dezelfde locatie komt. In de vorige paragraaf hebben we gezien dat dit een verandering teweeg brengt in de hope/fear emotie voor de prinses met waarde -45 en voor de held -15. Door gebruik van persoonsgebonden parameters zal de emotie bij de held slechts 1 tijdseenheid aanhouden. Bij de prinses zal dit 3 tijdseenheden aanhouden.

Standaarden
De standaarden geven aan, welke acties door de agent gewaardeerd worden en welke niet. Door het gebruik van de standaarden zullen de emoties pride/shame en admiration/reproach ten uitvoer kunnen worden gebracht. In figuur 7.5 kunt u een voorbeeld van hoe een standaard het emotiepaar pride/shame zou kunnen beïnvloeden.

```
Public void createStandard(Action o)
{
     if (o.equals(attack(me,other)) && (lovehate (other)>20))
     {
          prideshame = prideshame – 30
     }
}
```

Figuur 7.5 Een voorbeeld van een standaard

In de figuur is te zien dat, als een het karakter een actie ontvangt en deze actie gelijk staat aan het aanvallen van een ander, hij indien hij deze persoon liefheeft met een intensiteit van boven de twintig, zijn waarde voor prideshame met 30 zal zien dalen.

Action Tendency Scripts
Als derde en belangrijkste factor zijn er de “Action Tendency Scripts”. De “Action Tendency Scripts” zullen per persoon verschillend zijn. Deze scripts zijn bepalend voor het bepalen van de waardes voor de “Action Tendencies”. Meer over action tendencies leest u in paragraaf 6.3. Een voorbeeld van hoe zo’n “Action Tendency Script” eruit kan zien vindt u in figuur 4.14.

7.3 De Narrator Agent

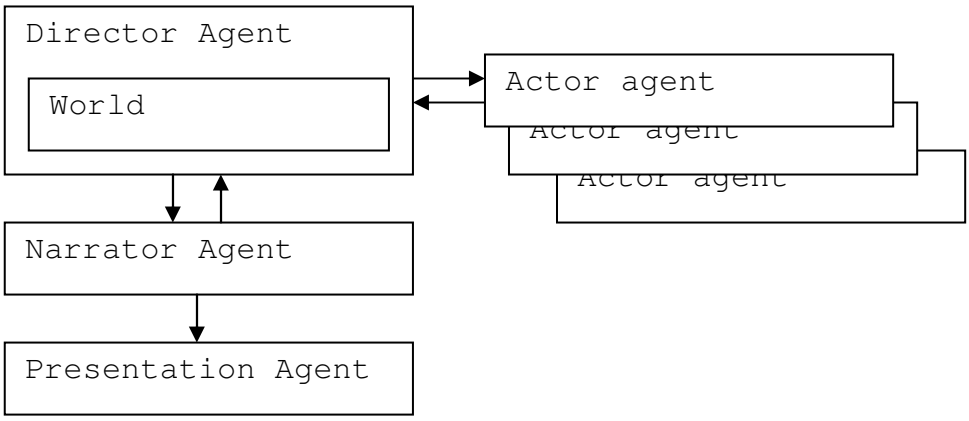
De Narrator Agent is het onderdeel van het systeem dat de acties, die in de wereld gebeuren omzet naar natuurlijke taal. In deze paragraaf vertellen we wat over de architectuur van deze Narrator Agent en de rol van de Narrator Agent binnen het systeem.

7.3.1 De architectuur

Voor de nieuwe Virtuele Verhalenverteller zullen we gebruik maken van de Narrator Agent zoals die gepresenteerd is door Sander Faas [Fa02]. We passen de architectuur dus niet aan. Deze Narrator beschikt over een Natural Language Generator die in staat is om acties om te zetten naar tekst. Voor een duidelijke beschrijving in detail van de Narrator agent verwijzen we u naar het verslag gemaakt door Sander Faas [Fa02].

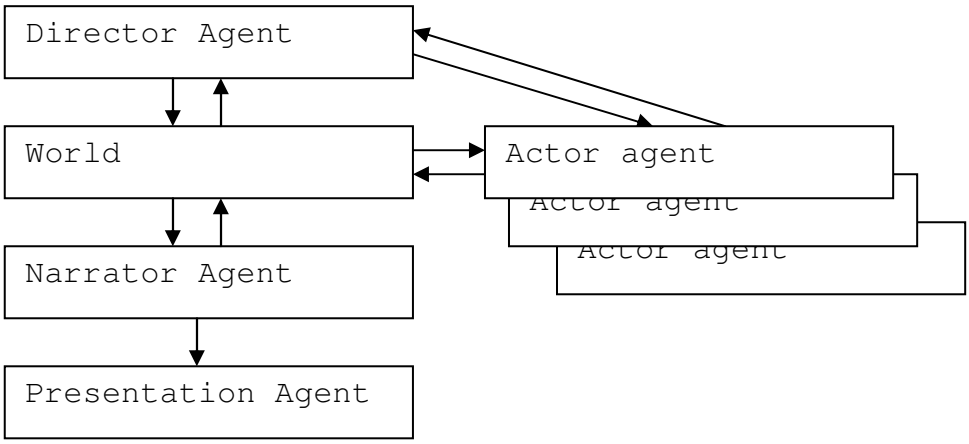
7.3.2 De rol van de Narrator binnen het systeem

Hoewel we in deze versie van de Virtuele Verhalenverteller niet zullen ingrijpen in de architectuur van de Narrator, willen we even de aandacht vestigen op de rol van de Narrator binnen het systeem. In de huidige vorm (zie figuur 7.6), krijgt de Narrator van de Director binnen welke acties er gebeuren in de wereld. De Director kan dus invloed uitoefenen op wat er verteld gaat worden. De Director heeft in dit geval als enige de beschikking over het totale wereldmodel en vertelt aan de Actor agents en de Narrator agents wat er precies gebeurt.



Figuur 7.6 De Narrator Agent in de huidige architectuur

Dit is niet een helemaal ideale situatie. Indien we een volledig autonome verteller willen, dan zou hij zelf de wereld moeten observeren en vervolgens vertellen wat er gebeurt (zie figuur 7.7). Op deze manier staat de Director los van wat er verteld wordt.



Figuur 7.7 De Narrator als pure beschrijver van de wereld

Toch maken we in deze versie van de Virtuele Verhalenverteller de keuze voor de huidige architectuur en niet voor die uit figuur 7.7. Dit doen we, omdat dit veel werk bespaart. Bij de eerste architectuur (Figuur 7.6) hoeft de Narrator Agent niets te weten over de wereld, alleen over de acties. Bij de tweede architectuur (Figuur 7.7) moet de architectuur van de Narrator overhoop worden gehaald, omdat deze de wereld moet kunnen interpreteren. Ook zal de wereld in het tweede systeem zelf moeten kunnen communiceren met zowel de Narrator als de Actor agents.

Door onze keuze, zal de Narrator alle informatie over de wereld binnenkrijgen via de Director. We willen echter wel graag een autonome verteller. Dit probleem lossen we zoveel mogelijk op, door de keuze te maken dat de Director alle acties die gebeuren in de wereld door zal geven naar de Narrator zonder enige aanpassing.

7.4 De Presentation Agent

De Presentation Agent is het onderdeel dat de verkregen tekst van de Narrator Agent omzet naar uitgesproken taal. Net zoals bij de Narrator Agent zullen we gebruik maken van de Presentation Agent die Sander Faas reeds gemaakt heeft. Deze Presentation Agent is in feite niets meer dan de Microsoft Agent [MSA99]. We zijn van mening dat deze agent een leuke aanvulling op het systeem geeft en zal voldoen voor de nieuw te maken Virtuele Verhalenverteller. Voor een beschrijving in meer detail van de Presentation Agent verwijzen we naar het verslag van Sander Faas[Faa02].

Onderdeel 3: Het Prototype

8. Van architectuur naar implementatie

In hoofdstuk 7 hebben we de architectuur gepresenteerd voor de nieuwe Virtuele Verhalenverteller. In dit hoofdstuk beschrijven we hoe de implementatie van de architectuur in zijn werk is gegaan.

8.1 Het systeem

In deze paragraaf zullen we enkele implementatie kwesties behandelen die betrekking hebben op het hele systeem.

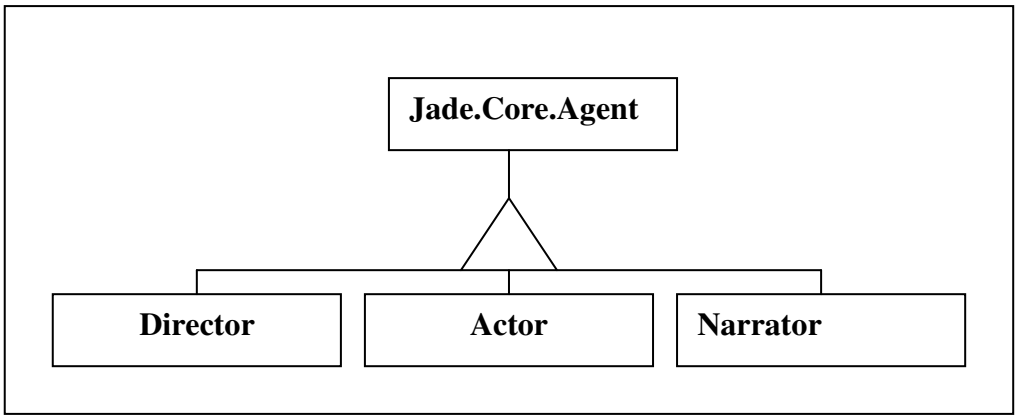
8.1.1 Het Agent raamwerk

Voor het implementeren van de architectuur, moeten er verschillende agents gemaakt worden. Deze zullen moeten samenwerken in een raamwerk. Sander Faas [Fa02] maakte gebruik van Jade (Java Agent Development Tool) om de agents te ontwikkelen. Jade bestaat uit een agent platform en een package voor het ontwikkelen van agents.

Faas noemt de volgende voordelen op voor het gebruik van Jade:

- Er hoeft geen eigen agent raamwerk worden gemaakt.
- Jade voldoet aan de Fipa standaards voor agents [Fip02]. Deze standaard wordt door vele ontwikkelaars aangehouden.
- Jade is open source. De ontwikkelaar heeft kan dus volledige controle uitoefenen over het raamwerk.
- Het Jade ontwerp team geeft een goede ondersteuning via een actieve mailinglist.

Gezien deze voordelen, het eindoordeel waarin Faas zeer positief is over het gebruik van Jade en de mogelijkheid om onderdelen over te kunnen nemen uit de implementatie van Faas, hebben we besloten om ook gebruik te maken van Jade.



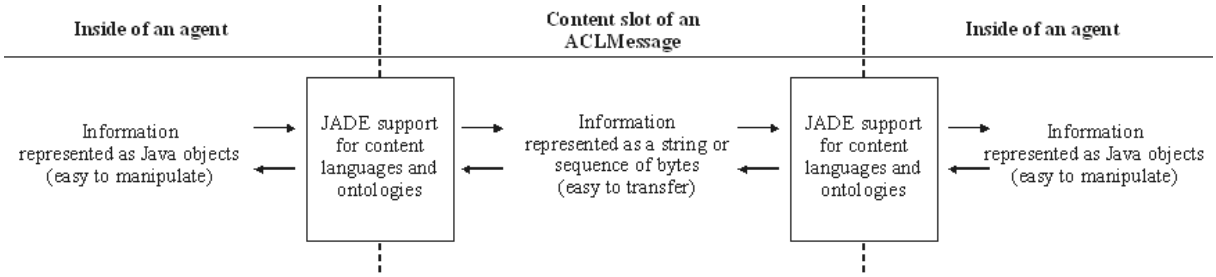
Figuur 8.1 De structuur van het systeem

In figuur 8.1 is de opbouw van het prototype te zien. Het prototype is opgebouwd uit drie packages: een package voor de Actor agents, een package voor de Director Agent en een package voor de Narrator Agent. Al deze packages maken gebruik van het Jade package.

8.1.2 De communicatie tussen de Agents

Binnen Jade vindt de communicatie plaats via zogenaamde ACLMessages. In ACLMessages wordt informatie gerepresenteerd als een string of een opvolging van bytes. Jade onderhoudt uit zichzelf een platform voor deze communicatie.

Binnen Jade bestaat de mogelijkheid om een Java object om te zetten naar de inhoud van een ACLMessage. Wel moeten de gebruikte Java objecten van tevoren gespecificeerd zijn. De specificatie van deze Java objecten noemen we een ontologie. Indien alle agents binnen een Jade platform beschikken over dezelfde ontologie, dan kan gebruik worden gemaakt van de Jade support voor ontologieën [Cai02].



Figuur 8.2 Agent-Agent communicatie in Jade

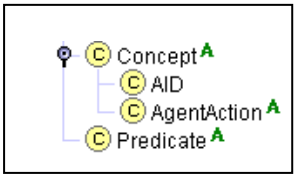
In figuur 8.2 is te zien hoe binnen Jade gebruik gemaakt wordt van een ontologie. Binnen de agent is informatie gerepresenteerd als een Java object. Dit Java object moet behoren tot de ontologie die de agents gebruiken. Jade zal dit object vervolgens representeren als een string (opeenvolging van karakters) en doorzenden naar de andere agent, die ook kennis heeft over dezelfde ontologie. Jade zet vervolgens voor deze agent de string weer om naar een Java object. Het Java object kan vervolgens weer gebruikt worden door deze andere agent.

In Jade kunnen dus, met behulp van een ontologie, Java objecten gebruikt worden voor het overbrengen van informatie. Dit is handig. Java objecten zijn gemakkelijk te maken, gemakkelijk aan te passen en gemakkelijk in te lezen.

8.1.3 Het maken van een ontologie

In deze paragraaf kijken we allereerst naar de eisen die Jade stelt aan een ontologie en als tweede kijken we naar hulpmiddelen die we hebben gebruikt om een ontologie te maken.

Eisen Jade
Een ontologie moet van Jade aan bepaalde eisen voldoen om daadwerkelijk gebruikt te kunnen worden. De grootste eis van Jade betreft de top van de ontologie. Deze ligt binnen Jade vast. De top van de ontologie(figuur 8.3) moet bestaan uit Concepten en Predikaten. Onder Concepten moeten in ieder geval een AID (Agent Identifier Object) object voor het identificeren van acties en een AgentAction object voor het beschrijven van acties van agents vallen.



Figuur 8.3 Top van de ontologie zoals vereist bij Jade

Naast deze eis vereist Jade dat er voor elk Java object bepaalde methodes aanwezig zijn om de eigenschappen van dat object te verkrijgen.

Hulpmiddelen
Het maken van een ontologie voor Jade is een arbeidsintensief werk zijn. Voor elke stukje informatie dat je van de ene agent naar een andere agent wilt sturen moet immers een Java Object (ook wel "Bean" genoemd) gemaakt worden. Gelukkig konden we voor het maken van een ontologie gebruik maken van Protégé 2000 versie 1.7. Met dit programma is het mogelijk om op eenvoudige wijze een ontologie te creëren. Ook is het in Protégé mogelijk om gebruik te maken van plugins. Door gebruik te maken van een plugin, speciaal bestemd voor Jade en gemaakt door Chris van Aart is het eenvoudig om de ontologie om te zetten naar Java Beans. Deze plugin wordt ook wel "Ontology Beangenerator" genoemd.

Sander Faas maakte voor de eerste Virtual Storyteller ook gebruik van deze versie van Protege en de Jade plugin. Voor een uitvoerige beschrijving, hoe dit programma werkt en hoe een ontologie aangepast kan worden verwijzen we u dan ook naar zijn thesis [Fa02].

8.1.4 De gebruikte ontologie

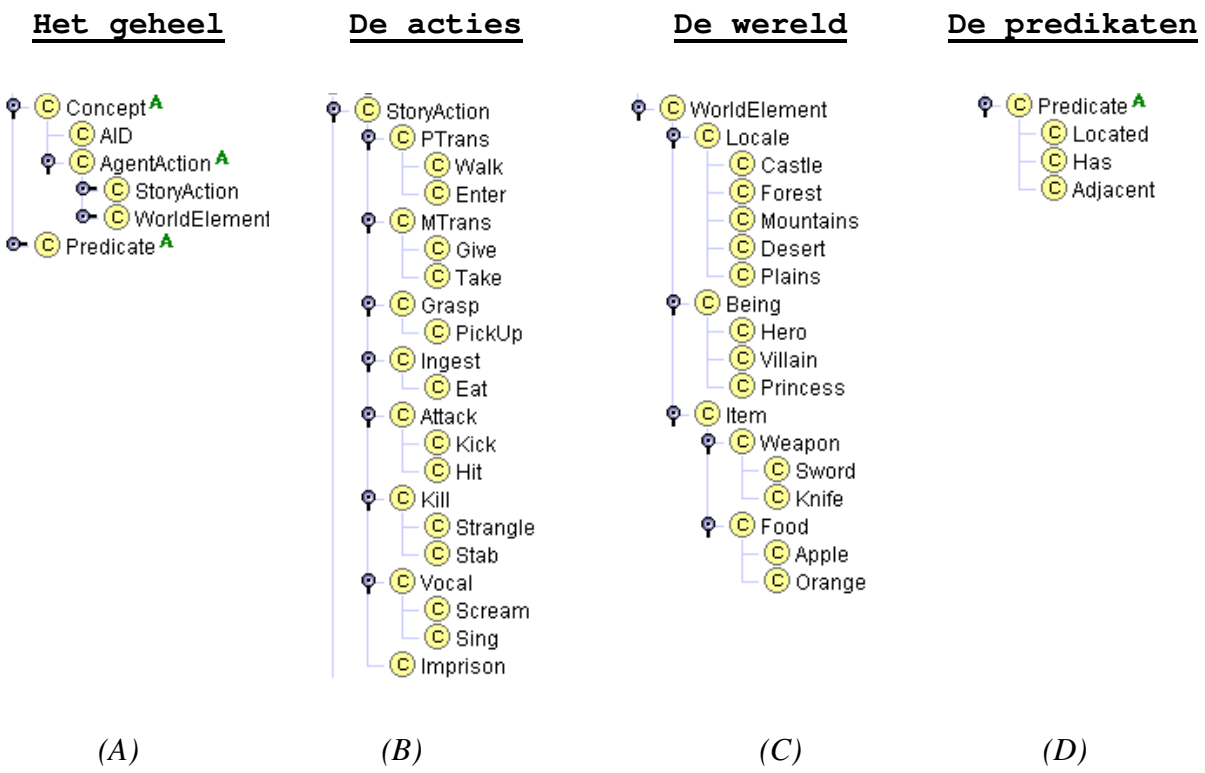
In de Virtuele Verhalenverteller hebben we gebruik gemaakt van twee ontologieën. De eerste ontologie beschrijft alle objecten die te maken hebben met het beschrijven van de wereld en alle acties die uitgevoerd kunnen worden, de tweede wordt gebruikt voor het uitwisselen van boodschappen.

Het beschrijven van de wereld en de acties
In figuur 8.4 ziet u een overzicht van de ontologie die de wereld beschrijft. Het geheel (8.4 A) is samengesteld volgens de normen van Jade. Zoals te zien is bestaat de top van de ontologie uit Concepten en Predikaten.

Het onderdeel Concepten bevat een identificatieobject (AID) waarmee een agent geïdentificeerd kan worden en agentacties (AgentAction). Onder agentacties vallen acties die tijdens het verhaal kunnen worden uitgevoerd en objecten die de wereld beschrijven (8.4C).

De acties die tijdens het verhaal uitgevoerd kunnen worden, zijn allemaal te zien in figuur (8.4B). Zoals te zien is, wordt er gebruik gemaakt van een boomstructuur. Een hoofdactie kan zich splitsen in andere acties. De "Attack" actie wordt bijvoorbeeld gesplitst in een "Kick" en een "Hit" actie. De reden hiervoor is, dat als bepaalde acties dezelfde kenmerken hebben er gekeken kan worden naar een hoofdactie. We geven een voorbeeld:

Voorbeeld:
"Het kleine meisje kan geen "Attack" acties uitvoeren", is gemakkelijker te implementeren dan "Het kleine meisje kan geen Kick actie uitvoeren" en "Het kleine meisje kan geen Hit actie uitvoeren".



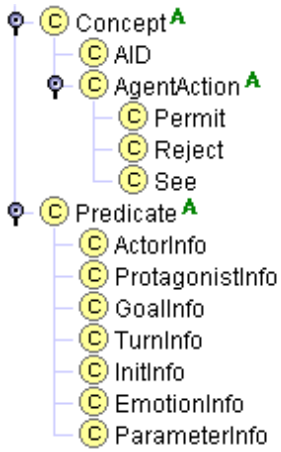
Figuur 8.4 De Ontologie die de wereld met zijn acties beschrijft.(gemaakt m.b.v. Protégé)

De elementen waar de wereld uit opgebouwd kan worden, zijn te zien in figuur 8.4C. Ook hier wordt er gebruik gemaakt van bepaalde hoofdobjecten, waar andere objecten onder vallen.

De objecten waar de wereld uit opgebouwd is (world elements), zouden eigenlijk niet onder Agent Acties moeten vallen. Jade staat echter alleen onder Agentacties overerving toe en om de wereld te beschrijving is het gebruik van overerving erg gemakkelijk. Deze plaatsing heeft dus met implementatie te maken.

Naast Concepten zijn er ook predikaten (zie figuur 8.4D). Predikaten geven de feiten uit de wereld neer. Zo betekent Located(Jos, Forest) dat het karakter met de naam Jos in het bos is.

Het uitwisselen van boodschappen
Voor het uitwisselen van boodschappen is een andere ontologie gemaakt. Ook deze ontologie bevat de voor Jade verplichte elementen Concepten en Predikaten. De ontologie wordt weergegeven in figuur 8.5.



Figuur 8.5 De ontologie voor het uitwisselen van opdrachten

De Permit en Reject acties worden gebruikt voor het vragen van toestemming voor bepaalde acties. Deze ontologie maakt dus gebruik van de ontologie voor het beschrijven van de wereld. De actie See is om implementie-redenen in deze ontologie geplaatst, maar zou eigenlijk thuishoren in de ontologie voor het beschrijven van de wereld.

De predikaten worden in deze ontologie gebruikt voor het versturen van informatie, die niet de wereld beschrijft, van de ene naar de andere agent. Zo stuurt een Director agent een DoelInfo object naar een Actor agent om te vertellen wat het episodische doel van het betreffende karakter is.

8.2 De Director Agent

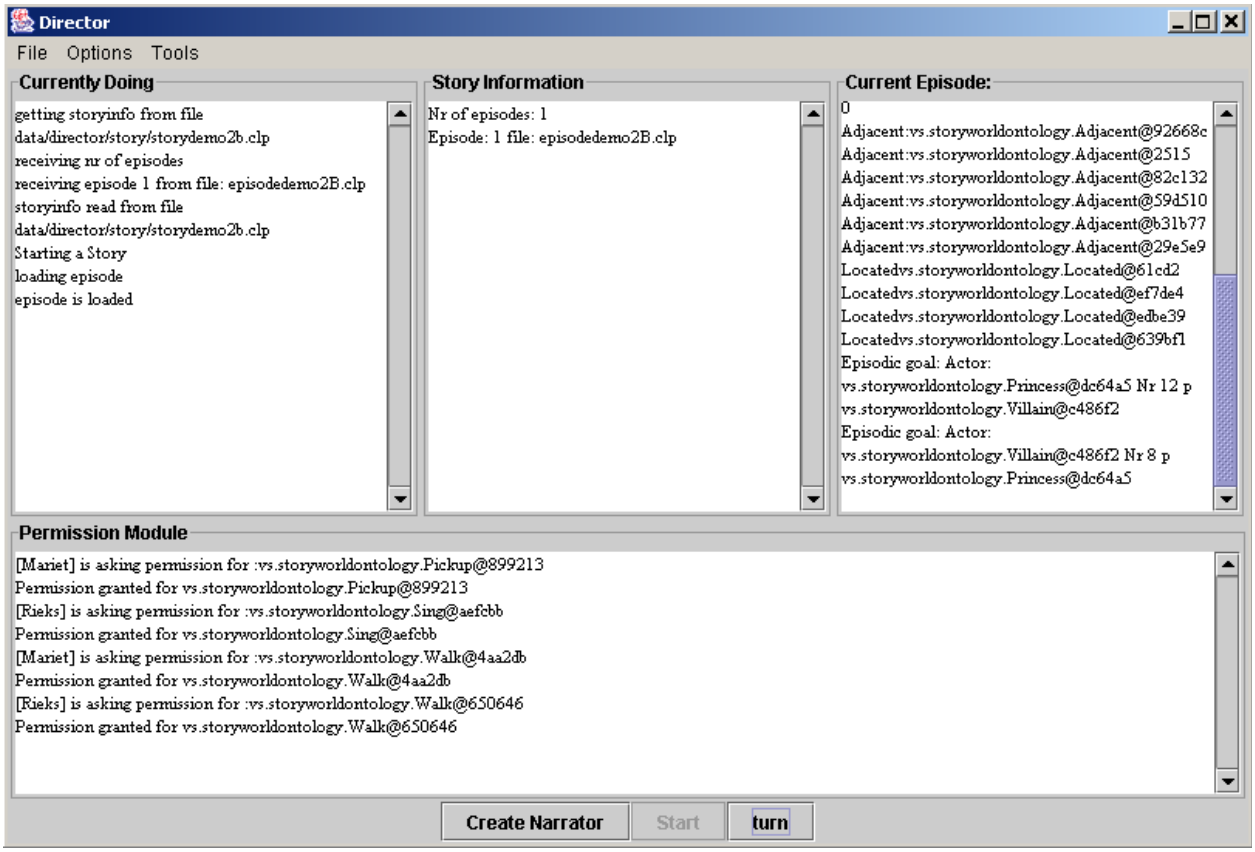
In deze paragraaf gaan we nader in op de implementatie van de Director Agent.

8.2.1 Het programma

De Director agent is de spil van ons prototype. Het is het programma wat als eerste wordt opgestart en dat tevens verantwoordelijk is voor het opstarten van andere programma's.

De Director agent uit het prototype voldoet vrijwel volledig aan de architectuur die gepresenteerd is in paragraaf 7.1. In ons prototype is de Director echter nog niet in staat om zelf een verhaalstructuur inclusief episodes te kiezen. De gebruiker zal een verhaalstructuur moeten kiezen en vervolgens zorgt de Director voor de invulling van het verhaal door het opstarten van de Actor agents die de karakters representeren.

De gebruiker is ook verantwoordelijk voor het daadwerkelijk laten uitvoeren van een zogenaamde beurt door een karakter. De gebruiker zal elke keer op een knop (turn; zie figuur 8.6) moeten drukken om een karakter zijn actie te laten berekenen.



Figuur 8.6 De Director

Zoals te zien is in figuur 8.6 kan de gebruiker, tijdens het verhaal de volgende informatie aflezen:

- Informatie over wat de Director op dat moment doet
- globale informatie over het hele verhaal
- gedetailleerde informatie over de huidige episode
- informatie over het al dan niet toestemming verlenen aan een karakter voor het uitvoeren van een actie

Op deze manier blijft de gebruiker op de hoogte van de ontwikkelingen wat betreft zaken die de Director aangaan.

8.2.2 Voorbeeld story en episode

De gebruiker moet voor het starten van een verhaal allereerst een verhaalstructuur selecteren. Hij doet dit door het openen van een file. Een voorbeeld van zo'n file ziet u in figuur 8.7.

```
;-------------------------------------------
; This file contains information about the story
; which episodes should be played !!
;-------------------------------------------

(deffunction createStory()

     (setNrOfEpisodes 1)
     (episode 1 "episodedemo2B.clp")
)
```

Figuur 8.7 een file met de verhaalstructuur

Zoals te zien is, bevat een file met de verhaalstructuur slechts gegevens over de aanwezige episodes. De informatie over de aanwezige episodes haalt een Director dus ook uit een file. Een voorbeeld van zo'n file ziet u in figuur 8.8.

Een file met informatie voor de episode moet van tevoren gemaakt worden. De informatie die zo'n file bevat is geschreven in de taal Jess [Fri97]. Jess is eigenlijk een redeneertaal, maar biedt een uitstekende mogelijkheid om op runtime files in te lezen. Het nadeel van het gebruik van Jess is dat om zo'n file te maken enige kennis van Jess vereist is, hoewel de taal niet al te moeilijk is. Een groot voordeel van het gebruik van Jess, is dat er functies gebruikt kunnen worden. We geven hier een voorbeeld van:

Voorbeeld:
"(bind ?Amalia (bactor Princess "Amalia" "female"))" geeft aan dat de variabele ?Amalia gebonden moet worden aan de uitkomst van de functie bactor die de parameters Princess, "Amalia"en "female"heeft. De functie bactor is hierbij een zelfgemaakte functie.

Er is dan ook een aparte Jess-file geschreven die functies bevat die gebruikt kunnen worden voor het schrijven van files met episode-informatie. Deze file is toegevoegd in bijlage 2.

```
;---------------------------------------------
; This file contains episode information
;---------------------------------------------
(deffunction createEpisode()
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; present actors: (-type- -name- -sex-)
      ;; the first actor is the protagonist !!
      (bind ?Amalia (bactor Princess "Amalia" "female"))
      (bind ?Brutus (bactor Villain "Brutus" "male"))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; present objects (-type- -strength-)
      (bind ?sword1 (bobject Sword 9))
      (bind ?sword2 (bobject Sword 9))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; number of locations in the world ( -number-)
      (nrOfLocs 7)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; present locations: maximum is nrOfLocs
      (bind ?forest1 (blocation Forest "het kleine bos"))
      (bind ?forest2 (blocation Forest "het grote bos"))
      (bind ?desert1 (blocation Desert "de woestijn1"))
      (bind ?desert2 (blocation Desert "de woestijn2"))
      (bind ?castle (blocation Castle "het kasteel"))
      (bind ?plains1 (blocation Plains "de kale vlakte"))
      (bind ?mountains1 (blocation Mountains "de bergen"))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; adjacents  -- don't make a cycle !!
      (adjacent ?forest1 ?desert1)
      (adjacent ?forest2 ?desert1)
      (adjacent ?castle ?desert1)
      (adjacent ?desert2 ?desert1)
      (adjacent ?plains1 ?desert1)
      (adjacent ?plains1 ?mountains1)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; located actors or objects
      (located ?Amalia ?forest1)
      (located ?Brutus ?desert2)
      (located ?sword1 ?mountains1)
      (located ?sword2 ?forest2)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; episodic goals
      (episodicgoalX ?Amalia 12 ?Brutus)
      (episodicgoalX ?Brutus 8 ?Amalia)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; episodic constraints
      (constraintX Kill ?Brutus ?Amalia)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
)
```

Figuur 8.8 Een file met episode-informatie.

8.2.3 Een Actor agent een actie laten uitvoeren

In paragraaf 7.2 is beschreven dat een agent voor elke actie die hij uitvoert toestemming moet vragen aan de Director. De Director is er dus verantwoordelijk voor wanneer en welke Agent een actie mag uitvoeren. De gepresenteerde archtectuur in hoofdstuk 7 laat bewust open hoe de Director hiermee om moet gaan. In het ideale geval zou een Agent actie op actie uit kunnen voeren met toestemming van de Director, zonder dat hij op een andere Agent hoeft te wachten. Een Agent zou een zelfde actie ook veel sneller kunnen uitvoeren dan een andere Agent. De implementatie hiervan is behoorlijk complex. Wat zou een Director bijvoorbeeld moeten doen als twee agents tegelijk toestemming vragen om dezelfde appel op te pakken?

Om problemen te voorkomen hebben we gekozen voor het geven van beurten. Elke agent krijgt een beurt, waarin hij één actie mag uitvoeren. Het toekennen van beurten gebeurt via het Round Robin principe. Het Round Robin principe houdt in, dat de agents op rij af de beurt krijgen.

Voorbeeld:
Er zijn drie agents.Agent1, Agent2, Agent3. Volgens het Round Robin principe, kijgt eerst Agent1 de beurt 1, vervolgens Agent2, vervolgens Agent 3. Zodra ze allemaal geweest zijn krijgen ze opnieuw in dezelfde volgorde de beurt. Dus weer Agent1, Agent2 en dan Agent3 enz.

De Director zal voordat hij een Actor agent aangeeft dat hij een actie mag uitvoeren eerst alle informatie sturen die hij nodig heeft. Hieronder verstaan we alle informatie over de locatie, zoals aanwezige objecten en karakters, maar ook acties die het karakter, dat door de Actor agent wordt vertegenwoordigd, heeft gezien. We geven hiervan een voorbeeld:

Voorbeeld
Setting: Prinses is in het bos. Held is in het bos. Schurk is in het bos. Het bos en de kale vlakte liggen naast elkaar. Er ligt een zwaard in het bos.
Beurt1 [Prinses]: Prinses loopt van het bos naar de hei.
Beurt2 [Schurk]: Schurk pakt zwaard op.

De held krijgt nu de volgende Java objecten binnen(engelstalig):
Located(Villain,Forest), WalkTo(Princess,Forest,Plains), PickUp(Villain,Sword), Has(Villain, Sword), Adjacent(Forest,Plains).

Een Actor agent weet dus pas vlak voordat hij toestemming krijgt om een actie uit te voeren, hoe de wereld er om hem heen uitziet en wat er in de wereld is gebeurd.

8.2.4 Het controleren van de acties

Een Actor agent moet toestemming vragen voor het uitvoeren van een actie. De Director gebruikt 3 soorten kennis voor het overwegen om toestemming te geven:

- De “episodische constraints”. Zoals aangegeven is paragraaf 5.2.2 zullen er bij een episode “episodische constraints” zijn. De Director kijkt of de acties wel binnen de “episodische constraints” toegestaan zijn.
- De Director bevat kennis over de wereld. Deze kennis staat bepaalde acties niet toe. De Director weet bijvoorbeeld, dat een prinses nooit de kracht kan hebben om een schurk op te pakken. Ook weet de Director dat een prinses nooit langs een schurk kan vluchten.
- De Director bevat beperkte kennis over verhalen. De Director weet bijvoorbeeld dat het drie maal achter elkaar uitvoeren van de “Attack” actie, het verhaal saai maakt en zal het niet toestaan.

De kennis die de Director van zichzelf bezit is nog vrij beperkt en toegespitst op de reeds gemaakte episodes. Wanneer er meerdere episodes en grotere werelden worden gemaakt, zal deze kennis moeten worden vergroot.

8.2.5 Problemen en tekortkomingen Director

Bij het implementeren van de architectuur van de Director, gepresenteerd in paragraaf 7.1, zijn we geen grote problemen tegengekomen. Wel is er echter 1 duidelijke tekortkoming van de Director aan te wijzen. De Director zou eigenlijk 4 episodes aan moeten kunnen die de volgende functies beschrijven:

- state of equilibrium
- disruption of state of equilibrium
- mission of hero
- return to state of equilibrium

De Director in het prototype kan echter nog niet omgaan met complexe verhaalstructuren van meerdere episodes achter elkaar. De Director kan slechts omgaan met episode-informatie zoals te zien is in figuur 8.8. In zo’n file worden eigenlijk de state of equilibrium en de disruption of the state of equilibrium beschreven. De Director weet deze twee wel te scheiden. De Director kan echter nog niet omgaan met episodeinformatie die de functie “mission of the hero” beschrijft. De Director in het prototype sluit het verhaal af met de laatste episode (“return to state of equilibrium”), zodra de functie “disruption of a state of equilibrium” is beschreven en een karakter zijn episodische doel heeft behaald. De beschrijving van de laatste episode hangt af van de uitkomst van de episode ervoor.

De beschreven tekortkoming van de Director is ontstaan door een tekort aan tijd. Tijdens het implementeren van het prototype hebben we echter wel zoveel mogelijk rekening gehouden met een eventuele volgende versie, waarin het gebruik van meerdere episodes wel tot uiting komt.

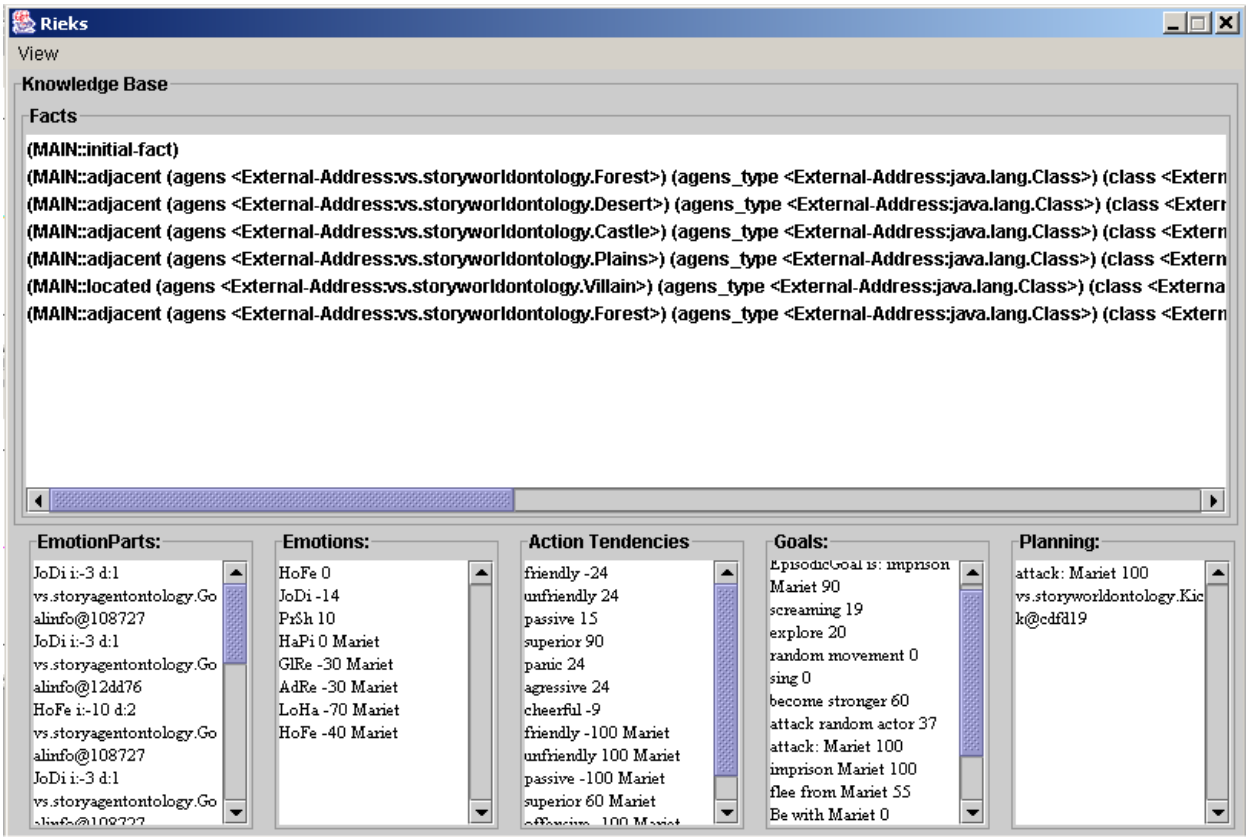
8.3 De Actor agent

Een karakter in een verhaal wordt vertegenwoordigd door een Actor agent. In deze paragraaf zullen we de gemaakte Actor agents eens nader bekijken .

8.3.1 Het Programma

Een Actor agent zal worden opgestart door een Director. Zodra dit gebeurt, zal er een venster verschijnen met daarin alle informatie betreffende de Actor agent (zie figuur 8.9).

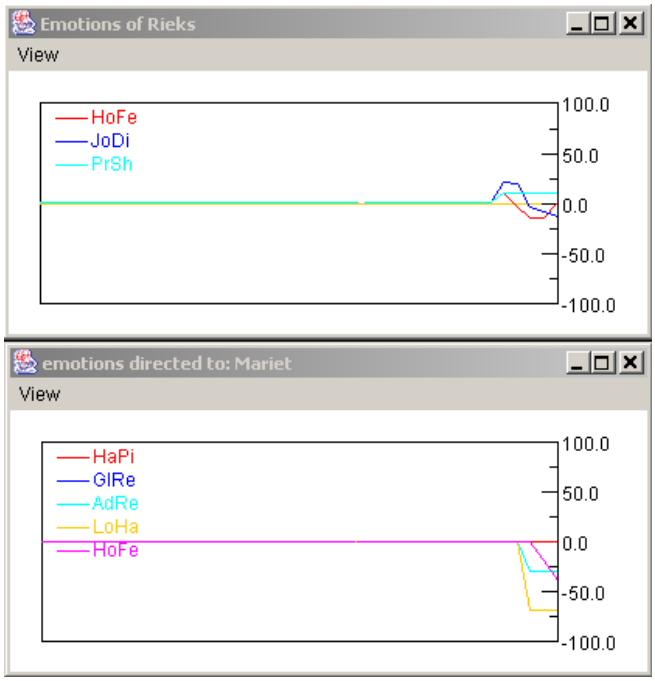
Zoals te zien is in figuur 8.9 geeft het venster informatie over het huidige wereldmodel (bovenste tekstscherm) en het hele proces van waarderen van gebeurtenissen tot aan het plannen van een actie (onderste 5 tekstschermpjes).



Figuur 8.9 Een Actor agent

Naast dit Venster met informatie is het ook nog mogelijk voor de gebruiker om de verandering van de intensiteit van de emoties in de loop van de tijd te zien. Deze worden getoond in grafiekjes (zie figuur 8.10).

Zoals te zien is in figuur 8.10 zijn er verschillende grafiekjes. Er is één grafiekje dat de emoties aangeeft, die een persoon altijd ervaart. Verder kan er voor elk ander karakter dan het karakter zelf een grafiekje worden geopend met emoties ten opzichte van dat andere karakter.



Figuur 8.10 De intensiteiten van de emoties van een karakter in de loop van de tijd.

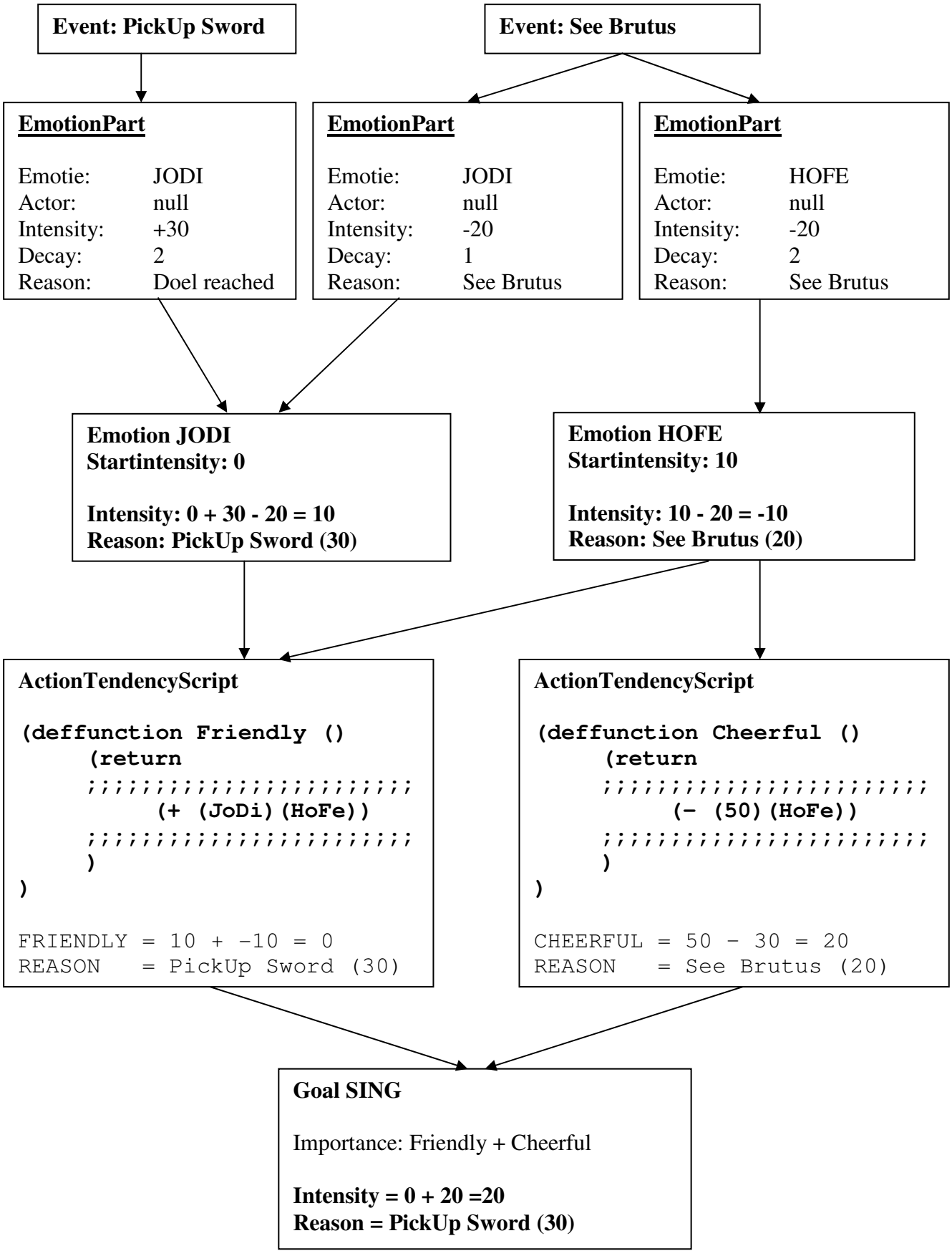
8.3.2 Van gebeurtenissen naar waardes van doelen

In hoofdstuk 6 hebben we beschreven hoe een karakter van het waarderen van gebeurtenissen naar het waarderen van doelen moet komen. De implementatie verschilt hier niet veel van, maar we zullen het proces nog even in detail weergeven.

Van gebeurtenissen naar het waarderen van doelen gaat in vijf stappen.

1 Er vinden gebeurtenissen plaats.
2 Deze gebeurtenissen worden gewaardeerd en aan de hand daarvan worden er “Emotion Parts” gemaakt. Per gebeurtenis kunnen er meerder “Emotion Parts” worden gegenereerd. Deze “Emotion Parts” zijn kleine Java objecten die weergeven hoe een gebeurtenis van invloed is op een bepaalde emotie.
3 Aan de hand van de “Emotion Parts” worden de intensiteiten van de emoties bepaald.
4 Vervolgens worden d.m.v de “Action Tendency Scripts” de waardes van de “Action Tendencies” bepaald
5 Aan de hand van de “Action Tendencies” worden de waardes van de doelen bepaald.

In figuur 8.11 geven we deze stappen precies weer.



Figuur 8.11 Van Gebeurtenis naar Doel

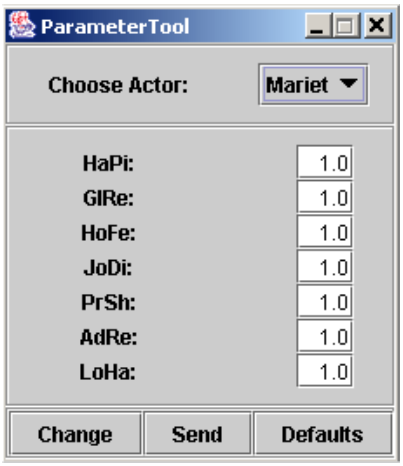
8.3.4 Persoonlijkheid

De persoonlijkheid van een karakter wordt bepaald door drie onderdelen (zie ook paragraaf 7.2.2):

- Persoonsparameters
- Standaards
- Action Tendency Scripts

Al deze onderdelen zijn in het prototype aanwezig. We beschrijven per onderdeel hoe een gebruiken de persoonlijkheid van een karakter kan aanpassen.

Persoonsparameters
Voor de persoonsparameters is er een hulpprogrammaatje gemaakt, waarmee de persoonsparameters voor of tijdens het verhaal zijn in te stellen. In figuur 8.11 wordt dit hulpprogrammaatje weergegeven.



Figuur 8.12 Hulpprogramma om de persoonsparameters in te stellen

Zoals te zien is, is voor elk karakter en elke emotie de persoonsparameter in te stellen. Het is dus mogelijk om een prinses die standaard bang is (parameters allen op 1.0) te veranderen in een dappere prinses door de parameters aan te passen.

Standaards
De standaards zijn verantwoordelijk voor het waarderen van het emotiepaar shame/reproach (zie paragraaf 7.2.2). Deze standaards verschillen in het prototype per type karakter. Er zijn standaards ontwikkeld voor de prinses en de schurk. Een gebruiker kan in principe niets veranderen aan de instellingen hiervan. Om de standaards te veranderen zullen er wijzigingen in de code moeten worden aangebracht.

Action Tendency Scripts
Action Tendency Scripts worden gebruikt om de waardes van de “Action Tendencies” vast te stellen (zie paragraaf 7.2.2). In figuur 8.10 is een gedeelte van zo’n “Action Tendency Script” te zien. Een compleet “Action Tendency Script” is te vinden in Bijlage 3.

“Action Tendency Scripts” zijn opgesteld in de taal Jess. De “Action Tendency Scripts “ zijn dusdanig opgesteld dat ze gemakkelijk zijn aan te passen.

8.3.4 Problemen en tekortkomingen Actor agent

Tijdens de implementatie zijn we twee behoorlijk grote problemen tegengekomen die belangrijk genoeg zijn om te worden genoemd.

- **Gebruik Java versie**
  Tijdens het uitproberen van de implementatie die gemaakt was door Sander Faas, bleek de Actor agent niet te werken onder een andere versie dan Java 1.4.0. Om onverklaarbare redenen bleef het programma elke keer op andere momenten hangen. Vermoedelijk ligt dit aan het gebruik van Jess. Ook bij de huidige versie van de Virtuele verhalenverteller is dit het geval. Hierin wordt ook Jess gebruikt. We hebben dit niet op kunnen lossen. HeWe hebben dit opgelost door bij het programma de juiste Java omgeving mee te leveren.

- **Planning**
  Sander Faas gebruikte in zijn implementatie van het karakter, de redeneertaal Jess voor de planning [Fa02]. In Jess werd gebruik gemaakt van regels die precies aangaven wat de precondities en wat de effecten van de acties waren. Helaas ging deze implementatie niet erg ver.

  Zelf hebben we geprobeerd deze taal ook te gebruiken. Helaas bleek dat zodra het verhaal iets complexer werd (met meerdere karakters en objecten) deze taal niet gemakkelijk gebruikt kon worden. Na een maand van mislukte pogingen is dan ook overgeschakeld naar een zelfgemaakt planningmechanisme in Java dat slechts in beperkte mate gebruik maakt van Jess. Dit planningsmechanisme is toegespitst op de soort verhalen en de acties behorende bij de nieuwe Virtuele Verhalenverteller. We denken niet dat deze implementatie erg geschikt is voor grotere en complexere verhalen, maar door tijdnood waren we genoodzaakt deze stap te nemen.

Naast deze problemen moeten we helaas ook één tekortkoming noemen.

- **Karakters**
  Op het moment is het slechts mogelijk om twee types karakters te gebruiken in een episode: een prinses en een schurk. Voor het maken van alle episodes, beschreven in paragraaf 5.2.2 is eigenlijk een derde karakter vereist, te weten een held. Zoals reeds aangegeven in paragraaf 8.2.5 zijn we door een gebrek aan tijd niet aan toegekomen om alle episodes te maken. Dit is ook de oorzaak van het niet aanwezig zijn van de held.

  Het maken van een type karakter is niet zo moelijk. Er moet alleen een persoonlijkheidsmodule voor dat karakter worden gemaakt:
  - Er moet een actiontendencyscript worden gemaakt voor dat type karakter. De locatie van het actiontendencyscript moet in de code worden opgegeven
  - De persoonsparameters moeten worden ingesteld
  - De standaarden voor het type karakter moeten worden gemaakt.

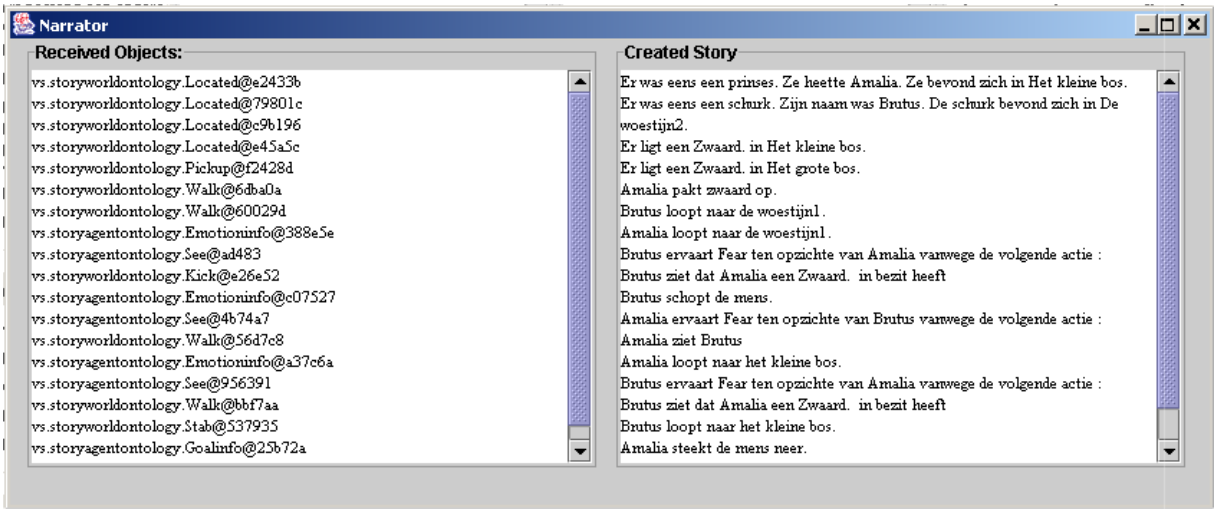
8.4 De Narrator en de Presentatie Agent

Zoals verteld in hoofdstuk 7 hebben we voor het prototype gebruik gemaakt van de reeds door Sander Faas gemaakte Narrator Agent en de Presentatie Agent. Door de geringe aanpassingen, zullen we deze onderdelen slechts kort beschrijven.

8.4.1 De Narrator

Zodra de Director is opgestart is het voor de gebruiker mogelijk om een Narrator Agent op te starten. Zoals te zien is in figuur 8.13 kan de gebruiker precies zien welke objecten de Narrator Agent binnenkrijgt en hoe deze objecten vertaald worden.

De enige aanpassing die gemaakt is, zijn het toevoegen van woorden aan de Natural Language Generator, zodat omgegaan kan worden met de nieuwe ontologie (zie paragraaf 8.1.4).



Figuur 8.13 De Narrator Agent

De Narrator kent nog grote tekortkomingen. We hebben de implementatie van Sander Faas gebruikt en Faas heeft de tekortkomingen reeds beschreven [Fa02]. De grootste tekortkoming van de Narrator is de gebruikte taal. De zinnen lopen niet vloeiend en zijn niet geheel in correct Nederlands.

8.4.2 De Presentatie Agent

Aan de Presentatie Agent zoals ook Sander Faas die gebruikte, is helemaal niets veranderd. Het is een klein figuurtje (zie figuur 8.14) dat overweg kan met de Nederlandse taal. Helaas weet de gebruikte Microsoft Agent niet zo goed gebruik te maken van intonatie, dus een verhaal zal met deze Presentatie Agent nooit echt mooi of spannend verteld kunnen worden.

Figuur 8.14 De Presentatie agent

9. Evaluatie

In dit hoofdstuk willen we het door ons gemaakte prototype evalueren. We doen dit door enkele verhalen, gemaakt door de Virtuele Verhalenverteller, eens nader onder de loep te nemen. We zullen voor de evaluatie kijken naar de onderdelen waar in deze thesis de nadruk op is gelegd: Structuur en geloofwaardige karakters.

9.1 Twee verhalen

In deze paragraaf laten we twee verhalen zien die gemaakt zijn door de Virtuele Verhalenverteller aan de hand van de file met episode-informatie uit figuur 8.8. De verhalen zijn te zien in figuur 9.1 en 9.2.

```
(1)  Er was eens een prinses. Ze heette Amalia. Ze bevond zich in
     Het kleine bos.
(2)  Er was eens een schurk. Zijn naam was Brutus. De schurk
     bevond zich in de woestijn2.
(3)  Er ligt een zwaard in de bergen.
(4)  Er ligt een zwaard in het grote bos.
(5)  Amalia loopt naar de woestijn1.
(6)  Brutus loopt naar de woestijn1.
(7)  Amalia ervaart angst ten opzichte van Brutus vanwege de
     volgende actie:
     Amalia ziet Brutus
(8)  Amalia loopt naar het grote bos.
(9)  Brutus loopt naar het grote bos.
(10) Amalia pakt zwaard op.
(11) Brutus ervaart angst ten opzichte van Amalia vanwege de
     volgende actie:
     Amalia pakt zwaard op.
(12) Brutus schopt de mens.
(13) Amalia steekt de mens neer.
(14) en ze leefde nog lang en gelukkig!!!
```

Figuur 9.1 Verhaal 1 gemaakt door de Virtuele Verhalenverteller

```
(1)  Er was eens een prinses. Ze heette Amalia. Ze bevond zich
     in Het kleine bos.
(2)  Er was eens een schurk. Zijn naam was Brutus. De schurk
     bevond zich in de woestijn2.
(3)  Er ligt een zwaard in de bergen.
(4)  Er ligt een Zwaard in het grote bos.
(5)  Amalia loopt naar de woestijn1.
(6)  Brutus loopt naar de woestijn1.
(7)  Amalia ervaart angst ten opzichte van Brutus vanwege de
     volgende actie:
     Amalia ziet Brutus
(8)  Amalia loopt naar het kleine bos.
(9)  Brutus loopt naar het kleine bos.
(10) Amalia schreeuwt.
(11) Brutus slaat de mens.
(12) Amalia ervaart angst ten opzichte van Brutus vanwege de
     volgende actie:
     Amalia ziet Brutus
(13) Amalia schopt de mens.
(14) Brutus schopt de mens.
(15) Amalia schreeuwt.
(16) Brutus pakt de mens op.
(17) Amalia schreeuwt.
(18) Brutus ervaart hoop ten opzichte van Amalia vanwege de
     volgende actie:
     Brutus pakt de mens op.
(19) Brutus loopt naar de woestijn1.
(20) Amalia schreeuwt.
(21) Brutus ervaart hoop ten opzichte van Amalia vanwege de
     volgende actie:
     Brutus pakt de mens op.
(22) Brutus slaat de mens.
(23) Amalia schreeuwt.
(24) Brutus gaat het kasteel binnen.
(25) Amalia schreeuwt.
(26) Brutus neemt in het kasteel de mens gevangen.
(27) en de mensen spraken jaren later nog over deze treurnis
```

Figuur 9.2 Verhaal 2 gemaakt door de Virtuele Verhalenverteller

Wat direct opvalt aan de verhalen is de gebrekkige taal, waaruit het verhaal is opgebouwd. Zinnen lopen hakkelend en er worden duidelijke taalfouten gemaakt. De focus van deze thesis ligt niet op deze gebieden (zie paragraaf 2.9). De Narrator zal duidelijk nog verder ontwikkeld moeten worden om goede taal te kunnen produceren.

9.2 Structuur

In deze paragraaf bekijken we de structuur van de verhalen die gepresenteerd zijn in de vorige paragraaf.

9.2.1 De structuur en de functies

Kunnen we kijkend naar de verhalen uit paragraaf 9.1 een duidelijke structuur aanwijzen? Zoals gezegd in paragraaf 8.2.5 heeft de Director de tekortkoming dat er nog niet omgegaan kan worden met de episode die de functie “mission of hero” moet beschrijven.

We geven weer, hoe in het eerste verhaal (figuur 9.1) de structuur naar voren zou moeten komen:

- Regel 1-4: Beschrijving van de begintoestand
- Regel 5-13: Introductie van de missie van de held
- Regel 14: Terugkeer naar de begintoestand

Wat meteen opvalt, is dat de tweede episode niet echt aansluit bij het verhaal. In de regels 5 tot en met 13 wordt verteld hoe de prinses de schurk doodt. Dit is geen beschrijving van de introductie van de missie van de held.

Kijken we naar het tweede verhaal (figuur 9.2) dan komt de structuur die we graag wilden wel sterk naar voren:

- Regel 1-4: Beschrijving van de begintoestand
- Regel 5-26: Introductie van de missie van de held
- Regel 27: terugkeer naar de begintoestand

In dit verhaal komt de introductie van de missie van de held wel goed tot uiting. In regel 5 t/m 26 wordt verteld hoe de schurk de prinses gevangen neemt. Hier zou prima een episode missie van de held op kunnen volgen. De missie van de held zou natuurlijk zijn het bevrijden van de prinses.

Hoe komt het nu, dat de structuur in het eerste verhaal niet zo goed tot uiting komt en in het tweede verhaal wel?

Dit komt door de tekortkoming van de Director. De Director Agent kan nog niet omgaan met de “mission of hero”. We wilden toch graag zien dat het verhaal een positief dan wel negatief einde kon hebben. Om dit te bereiken hebben we in de episode-informatie het episodische doel voor de prinses ingevoerd:

```
(episodicgoalX ?Amalia 12 ?Brutus)
```

12 is een interne code voor het predikaat doden. De prinses (Amalia) heeft dus als episodisch doel het doden van de schurk (Brutus). Dit doel sluit niet aan bij het introduceren van de missie van de held. De prinses zou juist als constraint moeten krijgen dat ze de schurk niet mag doden. De schurk zou immers nog een belangrijke rol moeten spelen in de episode die de missie van de held beschrijft. Bij een verdere ontwikkeling van de Virtuele Verhalenverteller zou zo'n doel dan ook niet moeten worden toegevoegd in de tweede episode.
Indien het episodische doel voor de prinses weggelaten zou worden, zou de introductie van de missie van de held goed verlopen.

9.2.2 De Director en de structuur van het verhaal

In paragraaf 8.2.4 is reeds aangegeven op welke manieren de Director de Actor agents binnen de structuur zou moeten houden. Komt dit nu ook naar voren in het verhaal?

Om dit te onderzoeken bekijken we voor het tweede verhaal (figuur 9.2) eens de informatie die van de Director komt, die betrekking heeft op het geven van toestemming voor het uitvoeren van een karakter.Deze informatie is te zien in figuur 9.3.

Zoals te zien is in de figuur 8.2.4 geeft de Director niet altijd toestemming voor een actie. Voor de duidelijkheid geven we deze informatie nog even weer:

(10) Action ptrans rejected for Amalia Reason: Princess can't pass a villain
(16) Action ptrans rejected for Amalia Reason: Princess can't pass a villain
(18) Action pickup rejected for Amalia Reason: Princess isn't strong enough to pickup Villain
(24) Action ptrans rejected for Amalia Reason: Princess can't pass a villain
(28) Action attack rejected for Brutus Reason: Too many attacks gets boring
(30) Action attack rejected for Brutus Reason: Too many attacks gets boring

De afwijzingen in regel 10, 16, 16 en 24 hebben allen betrekking op de kennis die de Director van de wereld heeft. Een prinses (Amalia) mag van de Director niet langs een schurk lopen en tevens is een prinses niet sterk genoeg om een schurk op te pakken.

De afwijzingen in regel 28 en 30 komen door de kennis die de Director heeft over verhalen. Het is niet aantrekkelijk voor het verhaal als Brutus de prinses maar blijft slaan en schoppen.

Een afwijzing doordat de Director constateert dat een actie binnen de "episodische constraints" niet mag worden uitgevoerd komt in dit verhaal niet voor.

De gebruiker, die nietsvermoedend het verhaal aanhoort, heeft niets in de gaten van alle afwijzingen. Een karakter dat een actie afgewezen ziet, vraagt gewoon toestemming voor een andere actie en het verhaal loopt vlotjes door.

Kijkend naar de 2 verhalen gepresenteerd in paragraaf 9.1 en naar de verhalen in Bijlage 4 kunnen we concluderen dat de Director de karakters elke keer binnen de aangegeven structuur weet te houden en bovendien de karakters geen onmogelijke acties laat uitvoeren.

```
(1)   [Amalia] is asking permission for :vs.storyworldontology.Walk@e708b2
(2)   Permission granted for vs.storyworldontology.Walk@e708b2
(3)   [Brutus] is asking permission for :vs.storyworldontology.Walk@6bd9e0
(4)   Permission granted for vs.storyworldontology.Walk@6bd9e0
(5)   [Amalia] is asking permission for :vs.storyworldontology.Walk@ff5c98
(6)   Permission granted for vs.storyworldontology.Walk@ff5c98
(7)   [Brutus] is asking permission for :vs.storyworldontology.Walk@5ce40
(8)   Permission granted for vs.storyworldontology.Walk@5ce40
(9)   [Amalia] is asking permission for :vs.storyworldontology.Walk@7fc8b2
(10)  Action ptrans rejected for Amalia Reason: Princess can't pass a villain
(11)  [Amalia] is asking permission for :vs.storyworldontology.Scream@6f947b
(12)  Permission granted for vs.storyworldontology.Scream@6f947b
(13)  [Brutus] is asking permission for :vs.storyworldontology.Hit@b41166
(14)  Permission granted for vs.storyworldontology.Hit@b41166
(15)  [Amalia] is asking permission for :vs.storyworldontology.Walk@bdb375
(16)  Action ptrans rejected for Amalia Reason: Princess can't pass a villain
(17)  [Amalia] is asking permission for :vs.storyworldontology.Pickup@c07527
(18)  Action pickup rejected for Amalia Reason: Princess isn't strong enough to pickup Villain
(19)  [Amalia] is asking permission for :vs.storyworldontology.Kick@8346a3
(20)  Permission granted for vs.storyworldontology.Kick@8346a3
(21)  [Brutus] is asking permission for :vs.storyworldontology.Kick@7881db
(22)  Permission granted for vs.storyworldontology.Kick@7881db
(23)  [Amalia] is asking permission for :vs.storyworldontology.Walk@54777e
(24)  Action ptrans rejected for Amalia Reason: Princess can't pass a villain
(25)  [Amalia] is asking permission for :vs.storyworldontology.Scream@5a2dc4
(26)  Permission granted for vs.storyworldontology.Scream@5a2dc4
(27)  [Brutus] is asking permission for :vs.storyworldontology.Hit@a69b6b
(28)  Action attack rejected for Brutus Reason: To many attacks gets boring
(29)  [Brutus] is asking permission for :vs.storyworldontology.Kick@13515e
(30)  Action attack rejected for Brutus Reason: To many attacks gets boring
(31)  [Brutus] is asking permission for :vs.storyworldontology.Pickup@5878d2
(32)  Permission granted for vs.storyworldontology.Pickup@5878d2
(33)  [Amalia] is asking permission for :vs.storyworldontology.Scream@89ec59
(34)  Permission granted for vs.storyworldontology.Scream@89ec59
(35)  [Brutus] is asking permission for :vs.storyworldontology.Walk@4c7a98
(36)  Permission granted for vs.storyworldontology.Walk@4c7a98
(37)  [Amalia] is asking permission for :vs.storyworldontology.Scream@70ec24
(38)  Permission granted for vs.storyworldontology.Scream@70ec24
(39)  [Brutus] is asking permission for :vs.storyworldontology.Hit@3461d1
(40)  Permission granted for vs.storyworldontology.Hit@3461d1
(41)  [Amalia] is asking permission for :vs.storyworldontology.Scream@a5e65f
(42)  Permission granted for vs.storyworldontology.Scream@a5e65f
(43)  [Brutus] is asking permission for :vs.storyworldontology.Enter@1ee017
(44)  Permission granted for vs.storyworldontology.Enter@1ee017
(45)  [Amalia] is asking permission for :vs.storyworldontology.Scream@fb24d3
(46)  Permission granted for vs.storyworldontology.Scream@fb24d3
(47)  [Brutus] is asking permission for :vs.storyworldontology.Imprison@d9e5ad
(48)  Permission granted for vs.storyworldontology.Imprison@d9e5ad
```

Figuur 9.3 Director en toestemmingsinformatie

9.2.3 Variatie binnen de structuur

We hebben in paragraaf 9.2.2 laten zien dat de karakters goed binnen de verhaallijn worden gehouden door de Director. De structuur van een verhaal lijkt goed nageleefd te worden, maar het is de vraag, of er nog wel genoeg variatie binnen de verhalen aanwezig. De door ons gebruikt episodes (paragraaf 5.2) zouden immers de karakters zoveel mogelijk vrijheid moeten geven, waardoor er variatie in de verhalen ontstaat.

We hebben dit getest, door de Virtuele Verhalenverteller vijf verhalen achter elkaar te laten vertellen met allen dezelfde episode-informatie. Deze vijf verhalen zijn te vinden in Bijlage 4.

Als we naar de verhalen kijken, zien we dat slechts één verhaal een goede afloop kent en vier niet. Van de vier verhalen die verkeerd aflopen lijken er twee verhalen behoorlijk sterk op elkaar (verhaal 2 en 5).

Voor het maken van deze verhalen is slechts een kleine wereld gecreëerd met maar twee karakters.

Hoewel deze test niet volledig representatief is, menen we wel te mogen concluderen dat er genoeg variatie aanwezig is binnen een verhaal met slechts één file voor episode-informatie.

9.3 Geloofwaardige Karakters

We hebben geprobeerd door middel van emoties geloofwaardige karakters te creëren. In deze paragraaf gaan we bekijken welke emoties de karakters gedurende een verhaal ervaren en welke invloed deze emoties op de verhalen hebben. Als laatste gaan we kijken of de geloofwaardige karakters die we gemaakt hebben ook daadwerkelijk geloofwaardig zijn.

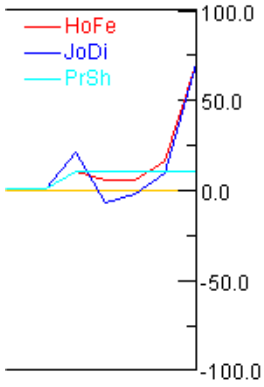
9.3.1 Emoties die karakters ervaren

De karakters ervaren emoties tijdens de verhalen, maar in welke mate ervaren ze eigenlijk deze emoties en komen deze intensiteiten overeen met het daadwerkelijke verhaal? We onderzoeken dit door te kijken naar de intensiteit die de emoties van karakters hebben tijdens het verhaal.

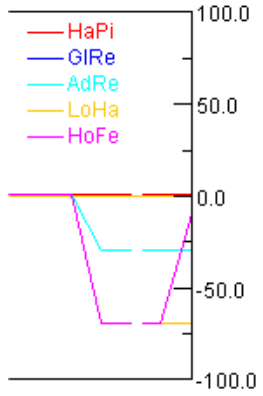
In figuur 9.4 ziet u de grafiekjes met de intensiteiten van de emoties in de loop van de tijd voor verhaal 1.

In figuur 9.4 (A) zien we de emoties van Amalia zelf. Zoals te zien is, ervaart Amalia vanaf het midden van het verhaal alleen positieve emoties. Dit komt sterk overeen met het verhaal. Eerst komt ze de schurk tegen en wordt ze bang dat ze haar doel niet kan bereiken, ook is ze niet blij. Ze weet echter te vluchten en een zwaard te bemachtigen. Haar emoties worden dan ook steeds positiever.
In figuur 9.4 (B) zien we de emoties van Amalia ten opzichte van Brutus. We zien hier duidelijk dat ze erg negatief tegenover Brutus staat. Ze is erg bang voor hem en haat hem. We zien ook, dat op het eind van het verhaal, wanneer ze het zwaard heeft en hem neersteekt niet meer bang is voor hem.

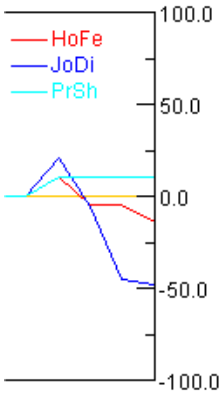
(A) Emoties Amalia zelf



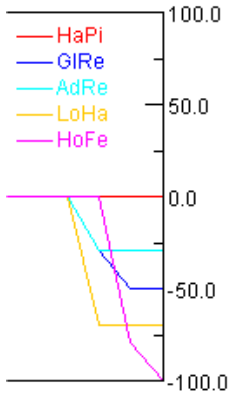
(B) Emoties Amalia t.o.v. Brutus



(C) Emoties Brutus zelf



(D) Emoties Brutus t.o.v Amalia

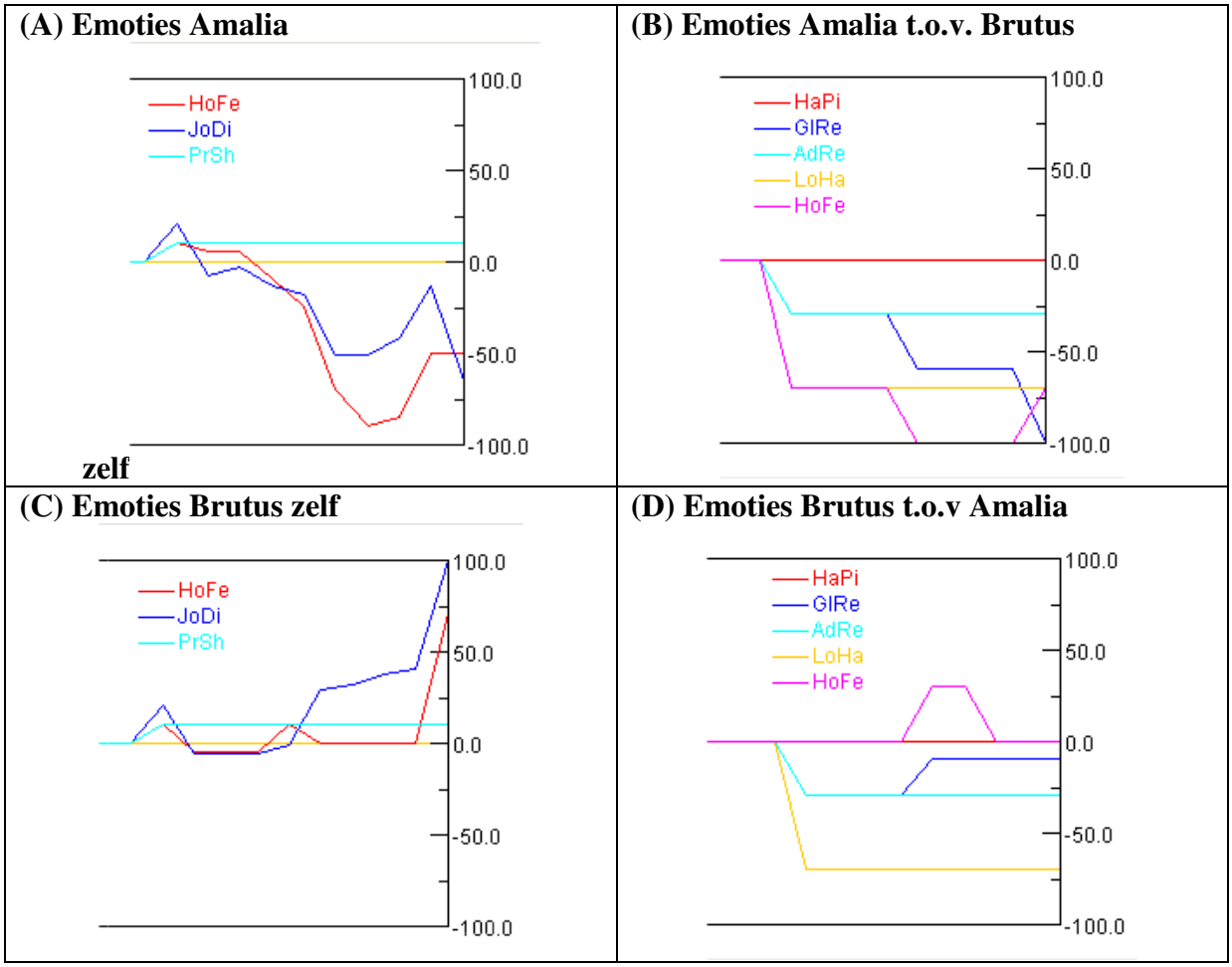


Figuur 9.4 Emoties verhaal 1

De emoties van Brutus zelf zijn te zien in figuur 9.4(C). We kunnen goed zien, dat Brutus gaandeweg het verhaal steeds minder positief is. Hij wordt boos, omdat de prinses hem ontvlucht en een zwaard weet te pakken. Op het moment dat de prinses een zwaard weet te pakken wordt hij erg boos en zelfs een beetje bang voor haar.
In figuur 9.4 (D) zijn de emoties van Brutus t.o.v. van Amalia te zien. Zoals te zien is, staat hij vanaf het begin erg negatief tegenover Amalia, maar is niet bang voor haar. Op het moment dat Amalia echter een zwaard weet te pakken, wordt hij ineens erg bang voor haar.

In figuur 9.5 ziet u de grafiekjes met de intensiteiten van de emoties in de loop van de tijd voor verhaal 2.

In figuur 9.5 (A) zien we de emoties van van Amalia zelf. Zoals te zien is, ervaart Amalia vanaf het moment dat ze de schurk tegenkomt meer negatieve dan positieve emoties. Ze is erg boos en is bovendien bang dat ze haar doel niet zal halen. Deze emoties worden steeds sterker.
De emoties van Amalia ten opzichte van Brutus zijn te zien in figuur 9.5 (B). We zien hier duidelijk dat ze erg negatief tegenover Brutus staat. Ze is erg bang voor hem en haat hem. Dit blijft gedurende het hele verhaal zo.



Figuur 9.5 Emoties verhaal 2

In figuur 9.5 (C) zien we de emoties van Brutus zelf. We kunnen goed zien, dat Brutus gaandeweg het verhaal steeds positiever wordt. Hij wordt steeds blijer (Jo/Di). Wat verder opvalt, is dat hij blijkbaar tot aan het eind er niet echt zeker van is of hij zijn doel zou halen. Pas als hij de prinses gevangen zet is hij er echt zeker van (Ho/Fe).
In figuur 9.5 (D) zijn de emoties van Brutus t.o.v. Amalia te zien. Zoals te zien is, staat hij vanaf het begin erg negatief tegenover Amalia, maar is hij niet bang voor haar (Ho/Fe). Dit blijft gedurende het hele verhaal zo.

Als we kijken naar de intensiteit van de emoties in de loop van de tijd, dan zien we dat de emoties van een karakter behoorlijk goed overeenkomen met het daadwerkelijke verhaal. De emoties van de karakters sluiten dus aan bij het daadwerkelijke verhaal.

Wat ook opvalt, is dat de emotieparen Admiration/Reproach en Pride/Shame nauwelijks veranderen. Dit komt, omdat de gebeurtenissen die plaatsvinden nauwelijks invloed hebben op deze emotieparen.

De intensiteiten van emoties zouden naar onze mening uitstekend kunnen helpen bij het goed animeren van de karakters. De intensiteiten kunnen perfect gebruikt worden om de gemoedstoestand van een karakter weer te geven.

9.3.2 De invloed van emoties op de verhalen

Nu we in paragraaf 9.3.1 hebben gezien dat de emoties overeenkomen met het verhaal, vragen we ons af of de emoties ook invloed hebben op het uiteindelijke verhaal.

De invloed van emoties op het verhaal zien we op twee manieren terug:

- Er wordt zichtbaar voor de gebruiker verteld in het verhaal wat een karakter voelt, waarom en wat de volgende actie is daardoor.
- Er wordt onzichtbaar voor de gebruiker een keuze voor een actie gemaakt die niet overeenkomt met het episodische doel.

We verduidelijken beide manieren nog even. Zoals te zien was in figuur 9.1 en 9.2 wordt voordat een karakter soms een actie uitvoert, verteld wat dit karakter voelt en waarom. We zien dit ook terug in het fragment in figuur 9.6 dat komt uit een door de Virtuele Verhalenverteller gegenereerd verhaal. Regel 1 laat zien welke emotie er met hoge intensiteit door het karakter wordt ervaren. Regel 2 geeft weer door welke actie deze hoge intensiteit van de emotie wordt veroorzaakt. Regel 3 toont de actie die gekozen is als gevolg van deze hoge intensiteit voor de emotie.

```
(1)  Amalia ervaart angst ten opzichte van Brutus vanwege de
     volgende actie:
(2)  Amalia ziet Brutus
(3)  Amalia schreeuwt
(4)  Brutus pakt de mens op.
```

Figuur 9.6 Fragment uit een verhaal

Uit figuur 9.6 kan tevens worden opgemaakt dat een karakter door zijn emoties soms inderdaad andere acties kiest dan zijn episodische doel. Hoewel dit niet te zien is in het fragment ligt er namelijk een zwaard op de locatie waar Amalia is. Als ze haar episodische doel had nagestreefd om Brutus te doden, dan had ze in plaats van te schreeuwen het zwaard op moeten pakken. In plaats daarvan zijn haar emoties blijkbaar dusdanig hoog, dat de waarde van het doel om te schreeuwen belangrijker wordt dan de waarde voor het episodische doel. De emoties lopen dus dusdanig op, dat er geen vorm van zelfcontrole meer is over de emoties. Amalia besluit dus om te schreeuwen.

Het besluit van Amalia om te schreeuwen heeft een zeer grote invloed op het verhaal. Nu kan Brutus haar oppakken en uiteindelijk gevangen nemen. Had ze echter het zwaard opgepakt, dan had de schurk haar niet zomaar gevangen kunnen nemen, maar had ze de schurk kunnen doden.

9.3.3 De geloofwaardigheid van de geloofwaardige karakters

Zijn de geloofwaardige karakters die we gemaakt hebben nu echt geloofwaardig? Deze vraag is erg moeilijk te beantwoorden.

We hebben de verhalen laten lezen door vijf andere personen. Alle vijf personen waren van mening dat de karakters geloofwaardig overkomen. Toen we echter de vraag stelden waarom

ze dat vonden, zeiden ze dat ze zich goed konden voorstellen dat een schurk en een prinses dergelijke acties zouden uitvoeren.

De acties die uitgevoerd werden pasten dus goed bij de vastomlijnde karakters. De schurk en de prinses deden wat van een schurk en een prinses werd verwacht. Dit past precies in onze verwachtingen, zoals we die al geuit hadden in paragraaf 4.2.

Het plot dat gegenereerd werd vonden ze in deze uitvoering geen van allen echt interessant. Allen vonden wel dat er een plot aanwezig was, maar dat het een beetje simpel was. De manier waarop er verteld werd (met hakkelende zinnen) moest aangepast worden. De meeste personen zagen echter wel in, dat als er over een grotere wereld werd verteld met meerdere karakters er een interessant plot zou kunnen ontstaan.

We mogen concluderen dat Virtuele Verhalenverteller in staat is om vastomlijnde karakters aan de algemene verwachtingen te laten voldoen. Op de vraag of de Virtuele Verhalenverteller echt in staat is om geloofwaardige karakters te maken zullen we het antwoord schuldig moeten blijven. Dit zal pas echt bekeken kunnen worden, als de Virtuele Verhalenverteller verhalen zal maken met gecompliceerde karakters die niet stereotype zijn.

Onderdeel 4: Discussie

10. Conclusies en aanbevelingen

10.1 Conclusies

We hebben een architectuur gepresenteerd, waarin geloofwaardige karakters met emoties de basis vormen voor een goed verhaal. De geloofwaardige karakters worden vertegenwoordigd door autonome Actor agents, die leven in een virtuele wereld. Doordat de Director Agent de karakters binnen een van tevoren gedefinieerde plotstructuur, bestaande uit episodes, laat blijven, kunnen er verhalen gemaakt worden met een interessante plot.

De architectuur die gepresenteerd is, geeft een grote vrijheid aan de gebruiker. Een gebruiker kan de structuur van een verhaal naar eigen inzicht aanpassen en zo een interessant plot creëren.

Daarnaast kan de gebruiker ook de karakters die meespelen in een verhaal aanpassen. De gebruiker kan een karakter veranderen door het wijzigen van persoonsgebonden parameters die van invloed zijn op de emotionele waardering van gebeurtenissen, of door het aanpassen van het “Action Tendency Script” van een karakter. Dit “Action Tendency Script” bepaalt het gedrag van een karakter op basis van zijn emoties.

De architectuur is geïmplementeerd als de Virtuele Verhalenverteller. De huidige implementatie van de Virtuele Verhalenverteller is in staat om korte verhalen te genereren met simpele plots.

Na het evalueren van het prototype van de Virtuele Verhalenverteller kunnen we het volgende concluderen:

- Het gebruik van “Episodes” die een functie beschrijven bieden een goede mogelijkheid voor het maken van een verhaal met een goede structuur en een plot.
- Een Director Agent is prima in staat om de karakters binnen de structuur (episode) van een verhaal te houden.
- Autonome Agents met emoties kunnen geloofwaardige karakters representeren in een virtuele wereld.
- Het gebruik van episodes beperkt de variatie van verhalen niet te veel.
- De gepresenteerde architectuur is uitstekend geschikt om korte verhaaltjes met interessante plots te maken.

10.2 Aanbevelingen

De Virtuele Verhalenverteller kan nog lang geen complexe verhalen maken met mooie zinnen. Toch hebben we een goede aanzet gegeven tot verdere ontwikkeling. We noemen enkele punten waaraan verder ontwikkeld zou kunnen worden:

- De virtuele wereld waarin op dit moment de karakters leven is nog erg beperkt. Het is een uitdaging om een grotere wereld te maken met vele karakters. Dit zal een complex verhaal ook nodig zijn.
- De karakters zouden veel uitgebreider gemaakt kunnen worden. De karakters hebben nu slechts de beschikking over beperkte doelen en acties. Door dit aantal te vergroten, zal een verhaal veel meer wendingen kunnen nemen

- De Director Agent zal verder ontwikkeld kunnen worden, door hem meer kennis van verhalen mee te geven. Hij moet in staat zijn om daadwerkelijk in te grijpen in verhalen door karakters en/of objecten te introduceren.

De architectuur die gepresenteerd is, treedt niet in detail wat betreft de Narrator Agent en de Presentation Agent. Het is belangrijk dat er voor deze Agents ook een goede architectuur wordt gemaakt.

Er zijn vele mogelijkheden om de Narrator Agent verder te ontwikkelen. We noemen de volgende:

- De Narrator Agent zou de acties om moeten zetten in goed en mooi Nederlands. De zinnen die nu gebruikt zijn zijn veelal mechanisch. Ook worden er veel dezelfde zinnen gebruikt.
- Er zouden nuances in het verhaal kunnen worden aangebracht door te kijken naar de emoties. (een persoon loopt stampvoetend als hij boos is)
- Een verhaal zou in meerdere talen kunnen worden verteld
- Hetzelfde verhaal zou vanuit verschillende oogpunten kunnen worden verteld

De ontwikkeling van de Presentation Agent, die het verhaal vertelt staat nog in de kinderschoenen. De ontwikkeling hiervan zou zich volgens ons in twee richtingen kunnen splitsen:

- Er wordt een geanimeerde verteller gemaakt, die zelf uitdrukkingen heeft en een verhaal op een goede manier weet te vertellen. (Denk hierbij aan fluisteren als het spannend is, goede klemtonen, intonaties enz enz.)
- Het verhaal zou daadwerkelijk in zijn geheel geanimeerd kunnen worden, waarbij dus de karakters daadwerkelijk zichtbaar zijn. We zien in onze gedachten al een toneelstuk gespeeld door echte karakters in het Virtuele Muziek Centrum [Nij99]. Het emotionele model gebruikt in deze architectuur zou hier als basis kunnen dienen bij het laten zien van de emoties die de karakters ervaren.

Literatuur

[App00] Appelcline Kimberly ;2000; The Elements of Good StoryTelling;skotos tech; http://www.skotos.net/articles/ELEMENTS.shtml

[Ari88] Aristoteles; 1988; Poetica; Nederlandse vertaling van Aristoteles poetics; Athenaeum Polak & van Gennep; Amsterdam.

[Bal97] Bal, Mieke; 1997; Narratology: Introduction to the Theory of Narrative, 2nd edition; University of Toronto; Toronto, Canada.

[Bar02] Bartneck, Cristoph;2002;Integrating the OCC Model of Emotions in Embodied Characters; In Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges; Department of Industrial Design; Eindhoven University of Eindhoven

[Bat92-1] Bates, J. Loyall, A.B., &Reilly, W.S.;1992; An architecture for Action, Emotion, and Social Behaviour, In AAAI spring symposium on integrated intelligent architectures, School of Computer Science, Pittsburgh; Carnegie Mellon University, Technical Rep. CMU-CS-92-144

[Bat92-2] Bates, J. Loyall, A.B., &Reilly, W.S.;1992; Broad Agents, School of Computer Science, Pittsburgh; Carnegie Mellon University.

[Bat94] Bates J.;1994; The role of emotion in believable agents;School of Computer Science, Pittsburgh; Carnegie Mellon University, Technical Rep. CMU-CS-94-36

[Bur02] Burghouts, G. 2002. Developing an action selection mechanism for emotional agents. Master's thesis, University of Twente.

[Cai02] Caire, Giovanni; 2002; JADE Tutorial – Application-defined content languages and ontologies; TILab; Turin, Italy.

[Cal00] Callaway, Charles B.; 2000; Narrative Prose Generation; Ph.D. thesis; North Carolina State University.

[cal01] Callaway, Charles B. and Lester, James C.; 2001; Narrative prose Generation; InProceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle; North Carolina State University. http://www.csc.ncsu.edu/eos/users/l/lester/www/imedia/papers.html

[Col73] Colby, B. N.;1973; "A partial grammar of eskimo folktales," American Anthropologist, vol. 75, pp. 645-662

[Ell92] Elliot, C.D. The Affective Reasoner: A process model of emotions in a multi-agent system", Ph. D. Thesis:Nortwestern University ,Evanston,Illianois.

[Faa02] Faas, Sander; 2002; Virtual Storyteller: an approach to computational storytelling; Masters thesis University of Twente

[FIP02] FIPA; 2002; FIPA Specifications; Foundation for Intelligent Physical Agents; Concord, USA. http://www.fipa.org

[Fri97] Friedman-Hill, E. J.; 1997; JESS, The Java Expert System Shell; Sandia National Laboratories; Livermore, California. http://herzberg.ca.sandia.gov/jess/

[Fre63] Freytag, Gustav; 1863;Technique of the Drama; Classic Books.

[Fre86] Freyda, N.H.; 1986; The emotions", Cambridge Unicersity, Cambridge, UK

[Gre66] Greimas A.J.;Semantique structurale; Presses universitaires de France, Parijs, 1986

[Joh81] Johnstone, Keith; 1981; Impro – improvisation and the theatre; Methuen Drama; London.

[Kau99] Kaufmann, David; 1999; What do we talk about when we talk about narrative?; George Mason University; Fairfax, Virginia. http://osf1.gmu.edu/~dkaufman/narrative.htm

[Kes01] Kesteren, A.J.; 2001; A supervised machine-learning approach to Artificial Emotions;Masters thesis University of Twente

[Kli99] C. Kline and B. Blumberg. 1999 The art and science of synthetic character design. Proceedings of the AISB 1999 Symposium on AI and Creativity in Entertainment and Visual Art.

[Lan97] Lang, R. Raymond; 1997; A formal model for simple narratives; Ph.D. thesis; Tulane University. ftp://juno.eecs.tulane.edu/pub/lang/

[Lee94] Lee, Mark G.; 1994; A model of story generation; M.Sc. thesis; University of Manchester. http://www.dcs.shef.ac.uk/~mlee/publications.html

[Loy97] B. Loyall.;1997; Believable Agents: Building Interactive Personalities. Ph.D. thesis CMUCS-97-123, Carnegie Mellon University.

[MSA99] Microsoft Corporation; 1999; Microsoft Agent 2.0; RedMond, USA. http://www.microsoft.com/msagent

[Mee81] Meehan, James; 1981; TALE-SPIN; In Schank, Roger C. & Riesbeck Christopher K.; Inside computer understanding – five programs plus miniatures; pages 197-226; Lawrence Erlbaum Associates; Hillsdale, New Jersey.

[Nij99] Nijholt,Anton; 1999; “The Twente Virtual Theatre Environment: Agents and Interactions.” Proceedings of the fiftheenth Twente Workshop on Language Technology, 147-165.

[Nim01] Nimis project,2001; Agent prototypes and documentation, report 5.2

[Ort88] Ortony, A., Clore, G.L., and Collins, A.; 1988; The cognitve Structure of emotions; Cambridge, UK, Cambridge University Press

[Pai01] Ana Paiva,Isabel Machado, Rui Prada 2001, Heroes, Villains, Magicians, …: Dramatis Personae in a Virtual Story Creation Environment, in Proceedings of the 6th international conference on Intelligent user interfaces, Santa Fe, New Mexico, USA.

[Pet98] Staller, A.,Petta, P. 1998; Towards a Tractable Appraisal-Based Architecture for Situated Cognizers, Grounding emotions in Adaptive Systems, SAB98, Workshop notes, Zurich

[Poe02] Poel, M.; op den Akker, R.; Nijholt, A.; and van Kesteren, A. 2002. Learning emotions in virtual environments. In Trappl, R., ed., *Proc of the 16th European Meeting on Cybern and System Res*, volume 2, 751–756.

[Pro68] Propp, Vladimir Jakovlevic; 1968; Morpohology of the folktale; 2nd ed.; University of Texas Press.

[Rei96] Reilly, W. Scott Neil;1996;Believable, Social and emotional agents;Masters thesis;School of Computer Science, Pittsburgh; Carnegie Mellon University

[Ros90] Roseman, I.J., Jose, P.E., & Spindel, M.S.; 1990. Appraisals of Emotion-Elicting Events: Testing a Theory of Discrete Emotions, Journal of Personality and Social Psychology 59(5), pp. 899-9915.

[Rum75] Rumelhart, D.E.; 1975; Notes on a schema for stories; In Representation and Understanding: Studies in Cognitive Science; Academic press, New York

[Rus95] Russel, Stuart J. and Norvig, Peter; 1995; Artificial Intelligence – a modern approach; Prentice-Hall International Inc.

[sch77] Schank, Robert & Abelson, Robert; 1977; Scripts, plans, doelen and understanding – An inquiry to human knowledge structures; Lawrence Erlbaum Associates; Hillsdale, New Jersey.

[seg88] Segre, Cesar; 1988; Introduction to the Analysis of the Literary Text; Indiana University Press; Bloomington.

[Tho81] Thomas, F. and Johnston, O.;1981; Disney Animation: The Illusion of Life; Abbeville Press, New York

[Too88] Toolan, Michael J 1988, Narrative: a critical linguistic introduction, Routledge, New York, USA

[Tur92] Turner, Scott R.; 1992; MINSTREL: a computer model of creativity and storytelling; Ph.D. thesis; Technical Report UCLA-AI-92-04; University of California.

Bijlage 1: Propps Functies

- *initial situation*; it is not a function (alpha)

I. **absentation**: a family member absents him/herself from home (beta)

II. **interdiction**: an interdiction is addressed to hero (gamma)

III. **violation**: interdiction is violated (delta); a paired element

IV. **reconnaissance**: villain makes an attempt at reconnaissance (epsilon)

V. **delivery**: villain receives information about his victim (zeta); a paired function

VI. **trickery**: villain attempts to deceive his victim in order to take possession of him or his belongings (eta) [at this point the villain may assume a disguise]

VII. **complicity**: victim submits to deception and thereby unwittingly helps his enemy (theta); subfunction: *preliminary misfortune* (lamda) wherein villain deliberately causes the difficult situation

VIII. **villainy**: villain causes harm or injury to a family member (A); crucial function by means of which the actual movement of the tale is created; *complication* is begun by an act of villainy

VIIIa. **lack**: one member of a family either lacks something or desires to have something (a)

IX. **mediation**, **the connective incident**: misfortune or lack is made known; hero is approached with a request or command; he is allowed to go or he is dispatched (B)

X. **beginning counteraction**: seeker agrees to or decides upon counteraction (C)

XI. **departure**: hero leaves home (arrow/up)

XII. **the first function of the donor**: hero is tested, interrogated, attacked, etc., which prepares the way for his/her receiving either a magical agent or helper (D)

XIII. **the hero's reaction**: hero reacts to the actions of future donor (E)

XIV. **provision or receipt of a magical agent**: hero acquires the use of magical agent (F)

XV. **spatial transference between two kingdoms, guidance**: hero is transferred, delivered, or led to the whereabouts of an object of search (G)

XVI. **struggle**: hero and villain join in direct combat (H)

XVII. **branding**, **marking**: hero is branded (J)

XVIII. **victory**: villain is defeated (I)

XIX. **liquidation of misfortune or lack**: the initial misfortune or lack is liquidated (K); this function, together with **villainy** (A), constitutes a pair: the narrative reaches its peak here

XX. **return**: hero returns (arrow/down)

XXI. **pursuit**, **chase**: hero is pursued (Pr)

XXII. **rescue**: rescue of hero from pursuit (Rs)

XXIII. **unrecognized arrival**: hero, unrecognized, arrives home, or in another country (o)

XXIV. **unfounded claims**: a false hero presents unfounded claims (L)

XXV. **difficult task**: a difficult task is proposed to hero (M); one of the tale's favorite elements

XXVI. **solution**: the task is resolved (N)

XXVII. **recognition**: hero is recognized (Q); complements function XVII

XXVIII. **exposure**: false hero or villain is exposed (Ex)

XXIX. **transfiguration**: hero is given a new appearance (T)

XXX. **punishment**: villain is punished (U)

XXXI. **wedding**: hero is married and ascends the throne (W)

Bijlage 2: Episodefuncties

Deze bijlage bevat de file met functies die gebruikt kunnen worden voor het maken van een file met informatie voor een episode. De functienaam vindt u na elke deffunction. Een functie moet worden aangeroepen m.b.v parameters. Deze parameters vindt u in tussen de haakjes na de functienamen.

```
;-------------------------------------------
; This file contains the functions that are used
; to read the episode scipts
;-------------------------------------------
(import vs.storyworldontology.*)
(import vs.storyagentontology.*)
(deffunction bactor (?class ?name ?sex)
      (bind ?actor (new ?class))
      (call ?actor setName ?name)
      (call ?actor setSex ?sex)
      (call ?*my-module* createActor ?actor)
      (return ?actor)
)

(deffunction bobject (?class ?strenght)
      (bind ?object (new ?class))
      (call ?*my-module* createObject ?object)
      (return ?object)
)

(deffunction nrOfLocs(?nr)
      (call ?*my-module* setNrOfLocs ?nr)
      (return ?nr)
)

(deffunction blocation(?class ?name)
      (bind ?location (new ?class))
      (call ?location setNamenl ?name)
      (call ?location setNameen ?name)
      (call ?*my-module* createLocation ?location)
      (return ?location)
)

(deffunction adjacent(?loc1 ?loc2)
      (bind ?adj (new Adjacent))
      (call ?adj setAgens ?loc1)
      (call ?adj setPatiens ?loc2)
      (call ?*my-module* createAdjacent ?adj)
      (return ?adj)
```

```
)

(deffunction located (?object ?location)
      (bind ?located (new Located))
      (call ?located setAgens ?object)
      (call ?located setPatiens ?location)
      (call ?*my-module* createLocated ?located)
      (return ?located)
)

(deffunction episodicgoal (?actor ?goal)
      (call ?*my-module* createEpisodicGoal ?actor ?goal)
)

(deffunction episodicgoalX (?actor1 ?goal ?actor2)
      (call ?*my-module* createEpisodicGoalX ?actor1 ?goal ?actor2)
)

(deffunction constraintX (?class ?agens ?patiens)
      (bind ?action (new ?class))
      (call ?action setAgens ?agens)
      (call ?action setPatiens ?patiens)
      (call ?*my-module* createConstraint ?action)
)
```

Bijlage 3: Een Action Tendency Script

In deze bijlage wordt een Action Tendency Script weergegeven, zoals gebruikt in de Virtuele Verhalenverteller.

```
;-----------------------------------------
; This file contains the dedault ats
; Watch it, whenever you have to call a function with another actor ?X;
; as argument,; the function gets a X extra !!
;-----------------------------------------
(deffunction Friendly ()
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (+ (JoDi)(HoFe))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction FriendlyX (?X)
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (+ (AdReX ?X)(HoFeX ?X)(LoHaX ?X))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction Unfriendly ()
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (- 0 (JoDi)(HoFe))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction UnfriendlyX (?X)
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (- 0 (AdReX ?X)(HoFeX ?X)(LoHaX ?X))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction Passive()
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (- 60 (HoFe) )
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction PassiveX (?X)
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             (- 0 (AdReX ?X)(HoFeX ?X)(LoHaX ?X))
```

```
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction OffensiveX (?X)
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (+ (HoFeX ?X))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)
(deffunction Superior ()
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (+ (PrSh) 10)
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)
(deffunction SuperiorX (?X)
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (if (< (LoHaX ?X) (- 0 30))
             then
             2
             else
             0
             )
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)
(deffunction Panic ()
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (- 0 (JoDi) (HoFe))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction Agressive ()
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (- 0 (JoDi) (HoFe))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)

(deffunction Cheerful ()
      (return
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (+ (JoDi) 20)
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      )
)
```

Bijlage 4: 5 verhalen van de Virtuele Verhalenverteller

In deze bijlage tonen we u 5 verhalen die door de Virtuele Verhalenverteller direct achter elkaar, met dezelfde episode-informatie zijn gegenereerd.

Verhaal 1:

Er was eens een prinses. Ze heette Amalia. Ze bevond zich in Het kleine bos.
Er was eens een schurk. Zijn naam was Brutus. De schurk bevond zich in De woestijn2.
Er ligt een Zwaard. in De bergen.
Er ligt een Zwaard. in Het grote bos.
Amalia loopt naar de woestijn1.
Brutus loopt naar de woestijn1.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia loopt naar het kleine bos.
Brutus loopt naar het kleine bos.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia slaat de mens.
Brutus pakt de mens op.
Amalia schreeuwt.
Brutus ervaart hoop ten opzichte van Amalia vanwege de volgende actie :
Brutus pakt de mens op.
Brutus loopt naar de woestijn1.
Amalia schreeuwt.
Brutus ervaart hoop ten opzichte van Amalia vanwege de volgende actie :
Brutus pakt de mens op.
Brutus gaat het kasteel binnen.
Amalia schreeuwt.
Brutus neemt in het kasteel de mens gevangen.
en de mensen spraken jaren later nog over deze treurnis

Verhaal 2:

Er was eens een prinses. Ze heette Amalia. Ze bevond zich in Het kleine bos.
Er was eens een schurk. Zijn naam was Brutus. De schurk bevond zich in De woestijn2.
Er ligt een Zwaard. in De bergen.
Er ligt een Zwaard. in Het grote bos.
Amalia loopt naar de woestijn1.
Brutus loopt naar de woestijn1.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia gaat het kasteel binnen.
Brutus gaat het kasteel binnen.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia slaat de mens.
Brutus schopt de mens.

Amalia schreeuwt.
Brutus slaat de mens.
Amalia schreeuwt.
Brutus pakt de mens op.
Amalia schreeuwt.
Brutus ervaart hoop ten opzichte van Amalia vanwege de volgende actie :
Brutus pakt de mens op.
Brutus neemt in het kasteel de mens gevangen.
en de mensen spraken jaren later nog over deze treurnis

Verhaal 3:

Er was eens een prinses. Ze heette Amalia. Ze bevond zich in Het kleine bos.
Er was eens een schurk. Zijn naam was Brutus. De schurk bevond zich in De woestijn2.
Er ligt een Zwaard. in De bergen.
Er ligt een Zwaard. in Het grote bos.
Amalia loopt naar de woestijn1.
Brutus loopt naar de woestijn1.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia loopt naar het kleine bos.
Brutus loopt naar het kleine bos.
Amalia schreeuwt.
Brutus slaat de mens.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia slaat de mens.
Brutus pakt de mens op.
Amalia schreeuwt.
Brutus ervaart hoop ten opzichte van Amalia vanwege de volgende actie :
Brutus pakt de mens op.
Brutus schopt de mens.
Amalia schreeuwt.
Brutus ervaart hoop ten opzichte van Amalia vanwege de volgende actie :
Brutus pakt de mens op.
Brutus schopt de mens.
Amalia schreeuwt.
Brutus loopt naar de woestijn1.
Amalia zingt.
Brutus slaat de mens.
Amalia schreeuwt.
Brutus slaat de mens.
Amalia schreeuwt.
Brutus gaat het kasteel binnen.
Amalia schreeuwt.
Brutus neemt in het kasteel de mens gevangen.
en de mensen spraken jaren later nog over deze treurnis

Verhaal 4:

Er was eens een prinses. Ze heette Amalia. Ze bevond zich in Het kleine bos.
Er was eens een schurk. Zijn naam was Brutus. De schurk bevond zich in De woestijn2.
Er ligt een Zwaard. in De bergen.
Er ligt een Zwaard. in Het grote bos.
Amalia loopt naar de woestijn1.
Brutus loopt naar de woestijn1.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia loopt naar de kale vlakte.
Brutus loopt naar de kale vlakte.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia loopt naar de bergen.
Brutus loopt naar de bergen.
Amalia pakt zwaard op.
Brutus ervaart angst ten opzichte van Amalia vanwege de volgende actie :
Amalia pakt zwaard op.
Brutus schopt de mens.
Amalia steekt de mens neer.
en ze leefde nog lang en gelukkig!!!

Verhaal 5:

Er was eens een prinses. Ze heette Amalia. Ze bevond zich in Het kleine bos.
Er was eens een schurk. Zijn naam was Brutus. De schurk bevond zich in De woestijn2.
Er ligt een Zwaard. in De bergen.
Er ligt een Zwaard. in Het grote bos.
Amalia loopt naar de woestijn1.
Brutus zingt.
Amalia zingt.
Brutus loopt naar de woestijn1.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia gaat het kasteel binnen.
Brutus gaat het kasteel binnen.
Amalia ervaart angst ten opzichte van Brutus vanwege de volgende actie :
Amalia ziet Brutus
Amalia slaat de mens.
Brutus pakt de mens op.
Amalia schreeuwt.
Brutus ervaart hoop ten opzichte van Amalia vanwege de volgende actie :
Brutus pakt de mens op.
Brutus neemt in het kasteel de mens gevangen.
en de mensen spraken jaren later nog over deze treurnis