Hallo!
Dit is het

Bachelor Opdracht verslag

over

Het gedrag van the next Furby

S.L. Bos
S0165352
10-06-2011

Titelpagina

Dit verslag is geschreven in het kader van de Bachelor Opdracht van de studie Industrieel Ontwerpen aan de Universiteit Twente.

Algemene Informatie

Titel	Het gedrag van the next Furby
Datum publicatie	10-06-2011
Aantal exemplaren	3
Aantal pagina's	49
Aantal bijlagen	9

Auteur

S.L. Bos	S0165352

Verantwoordelijke organisatie

Begeleider	H. Tragter
Naam	Universiteit Twente
Faculteit	Construerende Technische Wetenschappen
Opleiding	Industrieel Ontwerpen
Postadres	Postbus 217 7500 AE Enschede
Telefoonnummer	(053)4 89 91 11

Voorwoord

Dit verslag is geschreven in het kader van een Bachelor Opdracht aan de Universiteit Twente. Deze Bachelor Opdracht is een onderdeel van een serie Bachelor Opdrachten die samen tot een nieuwe generatie Furbies zou kunnen leiden. De opdracht is uitgevoerd in opdracht van de heer Tragter van de leerstoel Ontwerptechniek, afkomstig uit de vakgroep Ontwerp, Productie en Management (OPM) aan de Universiteit Twente.

Het doel van de opdracht was om aan de hand van een aantal (nog vast te stellen) gedragingen de geschikte programmeerhiërarchie voor 'the Next Furby' te ontwerpen. (De Furby is een knuffelbeest met sensoren en actuatoren zodat deze met de gebruiker kan interacteren.) Daarnaast zou dan ook kort gekeken worden naar de benodigde hardware die hiervoor nodig is. Het uiteindelijke product zou een oude Furby zijn die opnieuw geprogrammeerd was met de ontworpen programmeerhiërarchie.

Gedurende de opdracht wilde de opdrachtgever echter een iets andere kant op en werd het doel gedeeltelijk aangepast. Het uiteindelijke doel bleef een programmeerhiërarchie te ontwerpen. Maar dan bruikbaar bij elk slim product. Daarnaast was het niet van belang om de precieze gedragingen van een 'nieuwe generatie Furby' te bepalen. Wel zou het mogelijk moeten zijn om een nieuwe mentale toestand toe te voegen. Deze Furby werd uiteindelijk uit tijdsoverwegingen ook niet geïmplementeerd. Er werd gezamenlijk met K. Woestenenk een fictieve, slimme papegaai verzonnen en deze werd uiteindelijk geïmplementeerd.

Samenvatting

Dit verslag bevat een omschrijving van drie verschillende ontwerpfases die plaatsvonden gedurende de uitvoer van een Bachelor Opdracht Industrieel Ontwerpen aan de Universiteit Twente. Ten eerste het ontwerpen van het mogelijke gedrag van een 'nieuwe generatie Furby'. Aan de hand hiervan het bepalen van de werking van een dergelijke 'nieuwe generatie Furby'. En ten slotte een aantal programmeerstructuren ontworpen die het gedrag van een slim product moeten kunnen implementeren.

Fase 1 – Algemeen gedrag van de nieuwe generatie Furbies

In de eerste fase is er, aan de hand van de beperkingen van de 'oude generatie Furby' en aan de hand van een literatuuronderzoek over mentale toestanden, bepaald hoe het gedrag van een 'nieuwe generatie Furby' eruit zou moeten zien. De precieze implementatie van dit gedrag is niet tot in detail uitgewerkt, maar er wordt een basis geboden voor het gedrag, waaraan later eventueel verder gesleuteld zou kunnen worden.

Deze basis bestaat uit eventuele mentale toestanden, verschillende fases waarin de 'nieuwe generatie Furby' zich zou kunnen bevinden en de manier waarop de 'nieuwe generatie Furby' met de gebruiker moet kunnen interacteren.

Fase 2 – Alle aspecten van het gedrag

In de tweede fase wordt beschreven in welke aspecten het gedrag van de 'nieuwe generatie Furby' is opgedeeld. Deze aspecten bieden de basis voor de algemene (programmeer)structuur die het gedrag van een slim product later gemakkelijk zou moeten kunnen implementeren. Deze aspecten zijn tot stand gekomen na een langdurige brainstormsessie over verschillende soort structuren. Deze brainstormsessie kwam tot stand door het wenselijke gedrag van de 'nieuwe generatie Furby' en de resultaten van hetzelfde literatuuronderzoek uit fase 1 met elkaar te combineren.

De aspecten die uiteindelijk gekozen zijn en dan ook beschreven zullen worden zijn: Minds, waardes, (gemoeds)toestanden, wensen- en actieprioriteitenlijst, huidig gedrag en geheugen.

Fase 3 – Structuur voor dit gedrag

In de derde en laatste fase is er een onderzoek gedaan naar verschillende ontwerppatronen voor software. De gebruikte patronen zijn op basis van de werking van de 'nieuwe generatie Furby' gekozen. De vier structuren die aan de hand van deze ontwerppatronen zijn opgesteld zijn alle geïmplementeerd, toegepast op een ‘slimme papegaai’ en uiteindelijk met elkaar vergeleken. Tot slot is beschreven hoe het gedrag van de 'nieuwe generatie Furby' in een van deze structuren geïmplementeerd zou kunnen worden.

Abstract

This report describes the results of a Bachelor Assignment from the Bachelor Program of Industrial Design at the University of Twente. The results are divided in three different design phases. First, the design of possible behavior of a 'future generation Furby'. This resulted in the second phase, which describes how this kind of behavior can be divided in different aspects. The last phase will show some designs of programming structures that can implement the behavior of a 'smart product' using these aspects.

Phase 1 - Overall behavior of the 'future generation Furby'

The limitations of the 'old generation Furby' and research of different literature about mental states played a big role in defining the behavior of a 'new generation Furby'. The behavior is not designed as far as desirable, but it provides a good starting point for further design.

This phase provides possible mental states, different life cycle phases which the ‘new generation Furby’ can experience and the way in which the ‘new generation Furby’ must interact with the user of the product.

Phase 2 - Different aspects of the behavior of a smart product

This phase describes the aspects in which the behavior of the 'new generation Furby' will be divided. These aspects offer a good starting point for the overall structure that has to implement the behavior of a smart product. The results from phase 1 was the starting point for a brainstorm about a lot of different structures, and led to these aspects. The research of different literature about mental states mentioned in phase 1 was also very useful during this brainstorm.

The aspects which are finally chosen are: minds, values, (mental) state, priority list of wishes, priority list of actions, current behavior and memory.

Phase 3 - Structure for this behavior

Research has been done about different design patterns for software. The results from phase 2 where the main reason why patterns have been chosen to use. The four structures that are created using these patterns are all implemented. They are applied to a 'smart parrot' and compared. Finally there is a description about how the 'next generation Furby' can be implemented in one of these structures.

Inhoudsopgave

Inleiding

Robots met menselijk gedrag, zoals de Furby, worden steeds populairder en komen steeds vaker voor. Huishoudrobot, beveiligingsrobot, de robothond van Sony (Aibo).

Behalve al deze robots die zo menselijk mogelijk worden gemaakt, worden er ook steeds ‘normale’ slimme producten gebruikt in het alledaagse leven, bijvoorbeeld een slimme stofzuiger en een slimme grasmaaier. Toch wordt dit over het algemeen nog gezien als een stukje hoogstaande technologie. Het wordt nog niet standaard toegepast in elk product dat nieuw ontworpen wordt. Daarom biedt dit verslag een structuur aan die als model gebruikt kan worden voor het ontwerpen van een slim product; een structuur waarin alle onderdelen zitten om een product te maken met zijn eigen ‘gedrag’.

Deze structuur is ontworpen aan de hand van het ontwerpen van ‘the next Furby’ en de uitleg van de onderdelen van deze structuur zullen daarom ook grotendeels in het teken staan van deze Furby. Om duidelijk te maken dat deze structuur echter voor meerdere producten toe te passen is, is deze later toegepast op een 'slimme', virtuele papegaai.

In dit verslag is onderscheid te maken tussen de termen ‘Furby’ en ‘Furby*’. Met Furby worden de Furbies in het algemeen bedoeld, met name de al bestaande generaties. Furby* daarentegen staat speciaal voor de nog te ontwerpen nieuwe generatie van de Furby-familie.

De Furby is als basis gebruikt voor het ontwerpen van een programmeerstructuur die voor elk slim product toepasbaar moet kunnen zijn.

Voor Furby* is een Programma van Eisen(PvE) opgesteld dat weergeeft wat voor soort gedrag Furby* zou moeten tonen om de gebruiker gelukkig te maken. Daarna is er aan de hand van dit PvE en door middel van een literatuuronderzoek gekeken aan welke eisen de werking van Furby* moet voldoen. Dit is ‘vertaald’ naar een algemeen PvE voor de werking van een slim product. Aan de hand van dit PvE en aan de hand van een literatuuronderzoek naar ontwerppatronen voor software zijn er een aantal programmeerstructuren opgesteld en vervolgens getest. Tot slot is gekeken of deze structuur ook daadwerkelijk het gewenste gedrag van Furby* kan implementeren.

1. Algemeen gedrag van Furby*

Iedereen die Furby kent weet dat deze zich op een bepaalde manier gedraagt. Maar hoe is dit gedrag bepaald en waar is dit gedrag op gebaseerd? Dit zijn vragen die eigenlijk niemand weet te beantwoorden. Het gedrag van Furby zal niet gelijk zijn aan het gedrag van Furby, maar waarom dan eigenlijk niet? Dat zal in dit hoofdstuk uitgelegd worden. Er zal echter niet ingegaan worden op de verschillende gedragingen die Furby* allemaal zou moeten bezitten, dat wordt in deze opdracht niet bepaald. Wel zullen er voorbeelden genoemd worden van mentale toestanden die Furby* zou moeten kunnen bezitten.*

1.1 Slim speelgoed/elektronisch huisdier

G.H. Eisma(2011), een student die voor dezelfde opdrachtgever aan het werk was, beschrijft in het verslag van zijn Bachelor Opdracht dat Furby behoort tot de groep 'slim speelgoed' en de groep 'elektronisch huisdier'. Volgens zijn definitie is slim speelgoed gelijk aan: 'Interactieve objecten voor kinderen om mee te spelen, terwijl ze technologie gebruiken om dierlijke intelligentie te simuleren en de manier van spelen verbeteren door met de gebruiker te interacteren.' Hier zitten drie belangrijke gedeeltes in verwerkt die zeker terug moeten komen bij het ontwerpen van een gedragingenstructuur voor Furby*:

- Furby is om mee te spelen
- Furby moet dierlijke intelligentie simuleren
- Furby moet met de gebruiker interacteren

De andere naam die G.H. Eisma geeft aan Furby is een 'elektronisch huisdier' en ook legt hij het idee achter Furby uit in termen van een huisdier. 'Huisdieren spelen een belangrijke rol in de ontwikkeling van de identiteit van kinderen. Ze leren verantwoordelijkheden naar hun huisdier toe en leren hoe ze zich moeten associëren met andere levende wezens door respectvol gedrag naar het huisdier toe.' Bij een elektronisch huisdier gebeurt dit door ervoor te zorgen dat de gebruiker een aantal verantwoordelijkheden naar het dier heeft (zoals het dier voederen) en door te zorgen dat het dier ook negatief reageert op negatief gedrag van de gebruiker. Door over deze huisdieren en de welbekende teddybeer na te denken, wist Gerrit een aantal punten te noemen die de manier van spelen bij Furby toch limiteren:

- Bij jonge kinderen is het moeilijk om hun fantasiespelletjes te spelen (die ze wel met de teddybeer kunnen doen), doordat Furby zijn eigen persoonlijkheid met zijn eigen wensen heeft en het daarom niet altijd eens is met de scenario's die het kind wil uitvoeren. Furby stimuleert de verbeelding daardoor totaal niet.
- Oudere kinderen willen geen rollenspellen meer spelen met Furby, maar willen gebruik maken van de functies van Furby. Ook dit was helaas gelimiteerd, aangezien de 'oude' Furby maar een aantal spelsuggesties heeft en de reacties op de sensoren erg voorspelbaar zijn.

Om ervoor te zorgen dat de hiervoor genoemde problemen bij Furby* opgelost zijn, zal het gedrag van Furby* weer aan een aantal aspecten moeten voldoen:

- Er moet een mogelijkheid zijn dat Furby* op de gebruiker reageert zoals de gebruiker dat wil
- Furby* moet genoeg spelsuggesties bezitten om de gebruiker voor een bepaalde tijd niet te vervelen
- Furby* moet niet voorspelbaar reageren op de sensoren

Het eerste punt wordt niet meegenomen in deze opdracht. Er zal een structuur ontworpen worden die het product daadwerkelijk een 'eigen wil' geeft. De enige mogelijkheid die het kind zou hebben om Furby* te laten reageren zoals hij dit wil is om Furby* uit te zetten.

Tot slot een korte beschrijving van hoe het gedrag van Furby* eruit moet zien:

'De belangrijkste eigenschap in het gedrag van Furby is om de gebruiker nooit te vervelen, deze moet bijna constant interacteren met de gebruiker op een voor de gebruiker niet al te voorspelbare manier. Daarnaast moet Furby* een aantal eigen wensen hebben, zodat de gebruiker/het kind verantwoordelijken leert kennen en respect leert te geven. Furby* moet daarom bijvoorbeeld om een bepaalde tijd om eten vragen of laten merken dat hij moe is, waar de gebruiker vervolgens op kan reageren door deze eten te geven of te laten slapen. Verder moet Furby* ook negatief reageren op negatieve input van de gebruiker, zoals bijvoorbeeld het constant heen en weer schudden van Furby*, hier moet deze niet positief op reageren, want een echt levend wezen zou dit ook niet leuk vinden.*

1.2 Onderzoeksfase

Om het gedrag van Furby* zo te ontwerpen dat het daadwerkelijk een verbetering is op Furby, is er een PvE opgesteld waaraan dit gedrag zou moeten voldoen. Dit PvE is tot stand gekomen aan de hand van het schrijven van een aantal scenario's en door het houden van een literatuuronderzoek.

1.2.1 Scenario's

Voordat begonnen kon worden met het ontwerpen van een structuur waarin beschreven wordt hoe gedragingen werken was er eerst een brainstormfase, waarin scenario's een grote rol speelden. Deze scenario's werden geschreven om het gedrag van Furby* te beschrijven. Door middel van deze scenario's kon in het achterhoofd gehouden worden hoe Furby* zou moeten werken en met welke aspecten rekening gehouden moesten worden. In de scenario's is de gebruikte Furby daarom ook perfect ontworpen: hij gedraagt zich precies zoals je verwacht dat hij zich zal gedragen. De scenario´s zijn terug te vinden in Bijlage A.

De input van de scenario's is afkomstig uit de problemen en aanbevelingen die beschreven werden in het verslag van G.H. Eisma. Deze input is geselecteerd op problemen en aanbevelingen die gebruikt zou kunnen worden bij het ontwerpen van de geschikte programmeerhiërarchie en het kiezen van de juiste hardware. Aspecten zoals uiterlijk zijn niet meegenomen.

Een van de belangrijkste problemen die uit dit verslag naar voren kwam, was dat de sensoren niet goed ingesteld waren, deze waren of te gevoelig of niet gevoelig genoeg. Ook kunnen deze sensoren alleen het verschil waarnemen tussen aan&uit. Daardoor werd deze Furby niet heel menselijk gevonden, wat toch verwacht wordt bij een robot waarmee je kan communiceren. Deze sensoren zullen dus een belangrijke rol spelen bij het ontwerpen van Furby*.

Een ander belangrijk probleem is dat de entertainmentwaarde van Furby niet hoog genoeg is doordat de reacties van deze Furby erg gelimiteerd zijn en voor speciale reacties eerst de handleiding geraadpleegd moet worden. Uitgebreide spelletjes en zinnen/bewegingen die Furby* maakt zullen in deze opdracht niet bedacht worden, maar er zal wel kort genoemd welke gedragingen die Furby* eventueel zou kunnen krijgen en hierbij zal zeker nagedacht worden over de entertainmentwaarde die dit speelgoed moet hebben. Ook zal er nagedacht worden over de mogelijkheid van kinderen om met het speelgoed te communiceren zonder dat zij hierbij eerst de handleiding hoeven te raadplegen.

Een laatste probleem dat zeker behandeld moet worden in deze opdracht is dat een actie van Furby pas uitgevoerd kan worden wanneer een andere actie helemaal uitgevoerd is. In een hoop gevallen is dit niet gewenst, een actie zou daarom onderbroken of helemaal stop gezet moeten kunnen worden.

1.2.2 Literatuuronderzoek

Alhoewel deze scenario's een goede basis vormden voor de achterliggende gedachte van de gewenste werking van de gedragingen van een Furby* is dit nog niet voldoende om daadwerkelijk een goede start te maken. De eisen aan het gedrag zijn vaak nog erg algemeen en geven geen goed overzicht van wanneer Furby* in verschillende gedragingen terecht moet komen en hoe zijn gedragingen in verloop van tijd veranderen. Daarom is er literatuur geraadpleegd. Helaas was er geen literatuur te vinden die het gedrag van een mens perfect beschrijft, maar door verschillende literatuurbronnen te combineren met de al bestaande eisen is er uiteindelijk een goed overzicht gemaakt van hoe Furby* zich in zijn 'leven' zou moeten gedragen.

Een groot onderdeel van dit literatuuronderzoek zal in hoofdstuk 2 behandeld worden, er is echter wel een aspect wat in dit hoofdstuk belangrijk is.

Eerst is de 'Representational Theory of Mind' (David Pitt, 2008) onderzocht. Vervolgens was het handig om te kijken welke mentale toestanden er eigenlijk bestaan. Om deze te definiëren is er gebruik gemaakt van zes dimensies van het welzijn (C.D.Ryff& C.L.M.Keyes, 1995): zelfacceptatie, persoonlijke groei, doel in het leven, positieve relaties met anderen, omgevingsgericht en autonomie. Daarnaast is ook gebruik gemaakt van de fases van het zelfbewustzijn op het pad naar het welzijn (C.R.Cloninger, 2006). Dit beschrijft welke fases er bestaan qua gedrag. Het gedrag uit de eerste fase werd beschreven als het gedrag van een kind en de tweede fase bevat veel negatieve emoties, wat goed bij de puberteit past. De derde en vierde fase zijn beide erg kalme fases

Alhoewel het optimaliseren van het welzijn natuurlijk niet het doel is bij het ontwerpen van 'the next Furby*', is deze informatie toch goed te gebruiken. Het momentele welzijn van een persoon zegt veel over zijn humeur en daarmee ook over zijn (gemoeds)toestand. De fases van het zelfbewustzijn op het pad naar welzijn doen erg veel denken aan het volwassen worden van een persoon..

Met deze fases, de gewenste entertainmentwaarde en de gewenste geloofwaardigheid van het 'leven' van Furby* in het achterhoofd gehouden is gekozen om Furby* een complete levenscirkel te laten meemaken. Uitzondering hierop is natuurlijk het sterven en ouderdomsproblemen. De levenscirkel van Furby* zal bestaan uit een aantal fases, waarbij hij in de eerste fase nog haast niks kan en hij in de laatste fase onder andere de toegang geeft tot al zijn spelletjes. Hierbij moet natuurlijk wel genoemd worden dat deze fases geleidelijk in elkaar overlopen en deze fases alleen beschreven zijn om een overzicht te geven tot de emoties die Furby* zal uiten. (En wanneer de gebruiker het wil kan deze Furby* weer in zijn eerste levensfase laten beginnen.)

Om het gedrag goed te kunnen beschrijven is er tijdens het ontwerpen van deze fases rekening gehouden met de 6 dimensies van het welzijn. Om te zorgen dat 'the next Furby' wel een leuk speelgoed blijft, zal deze zo weinig mogelijk negatieve emoties moeten bevatten. Daardoor zullen voornamelijk de positieve eigenschappen van deze dimensies voorkomen in het gedrag van de Furby. In Bijlage B wordt kort beschreven wat met de term 'dimensies van het welzijn' bedoelt wordt en wat voor gedrag er bij verwacht kan worden.

1.2.3 Programma van Eisen

De problemen en aanbevelingen die beschreven werden in het verslag van G.H. Eisma, de conclusies uit de scenario´s en de uitkomsten van het literatuuronderzoek zijn samen overzichtelijk verwerkt tot een PvE, onderverdeelt in de 4 belangrijkste aspecten van de functionaliteit van de Furby:

1. Geloofwaardigheid van het 'leven' van Furby*;
2. Entertainmentwaarde;
3. (Gemoeds)toestand;
4. (Re)acties Furby*

De bovenste twee punten zijn de belangrijkste problemen die terugkwamen in het verslag van G.H. Eisma en de onderste twee punten zijn de aspecten die de werking van het gedrag van Furby* zullen bepalen. Aan de hand van dit PvE werd het gedrag dat Furby* moet hebben uitgebreider onderzocht en de uitkomsten hiervan vallen terug te lezen in de rest van dit hoofdstuk en in hoofdstuk 2.

Het PvE is te vinden in Bijlage C en het PvE vertaald naar het PvE voor een algemeen slim product is terug te vinden in Bijlage D.

1.3 Fases van Furby*

Zoals eerder genoemd zal Furby* een complete levenscirkel meemaken, deze cirkel bestaat uit 4 hoofdfases, die wel geleidelijk in elkaar overlopen. Deze fases zijn ontworpen door uit te gaan van de 4 fases richting het welzijn (C.R.Cloninger, 2006) en door de 6 dimensies van het welzijn (C.D.Ryff& C.L.M.Keyes, 1995) in het achterhoofd te houden. Hieronder is er per fase een beschrijving van het gedrag van Furby*.

Fase 1 (kind)

Furby* wordt 'geboren' en komt dus helemaal nieuw in de wereld. Als je er vanuit gaat dat Furby* zoveel mogelijk menselijke eigenschappen moet bezitten dan betekent dit dat Furby* eigenlijk zo goed als niks moet kunnen en niks moet weten en snappen van de wereld. Op deze manier is het echter totaal geen aantrekkelijk speelgoed voor de doelgroep: De entertainmentwaarde is erg laag doordat er weinig reactie komt vanuit Furby*. Doordat de communicatie tussen Furby* en de gebruiker hierdoor erg laag is zal de gebruiker er snel op uitgekeken zijn. Furby* zal daarom zoveel acties moeten kunnen uitvoeren dat de gebruiker zich een aantal uur kan vermaken. Een aantal beperkingen kunnen echter wel ingebouwd worden.

Het gedrag van Furby* kan beschreven worden als:

- Geen volzinnen, zinnen bestaan uit 2/3 woorden;
- Minder reactie op de interpretatie van zinnen die de gebruiker aan Furby* zegt dan op het feit dat er een sensor wordt geactiveerd;
- Erg expressief, zoals het uitbarsten in huilen, schreeuwen of vaak herhalen van het feit dat hij honger heeft

Fase 2 (puber)

Furby* is al iets ouder geworden en heeft door dat hij steeds ouder wordt en daarbij een aantal doelen kan bereiken. Zijn idee van het leven hierbij is dan ook dat hij graag zijn wil hebben. Hij vraagt hierbij veel aandacht om zijn doelen ook daadwerkelijk te bereiken. Hij is minder expressief dan in de eerste fase, maar weet in extreme situaties ook nog goed van zich te laten horen.

Zijn idee van de wereld is nog niet helemaal optimaal, waardoor hij een hoop vragen stelt over hoe het nou precies zit en over hoe de gebruiker erover denkt. Zijn eigen wil is wat dit betreft toch nog niet sterk genoeg, want hij gelooft de gebruiker vaak wanneer deze hem vertelt wat hij moet doen.

Het gedrag van Furby* kan beschreven worden als:

- Simpele volzinnen;
- De interpretatie van wat de gebruiker verteld is belangrijk voor Furby*, hij wil graag een beeld vormen van de wereld;
- Hij stelt erg veel vragen;
- Hij heeft een laag irritatieniveau, waardoor hij snel toch in huilen uitbarst/begint te schreeuwen

Fase 3 (volwassene)

Furby* heeft een goed beeld gekregen op de wereld door al de vragen die hij in zijn vorige levensfase heeft gesteld. Daardoor heeft hij een hoop respect en interesse gekregen voor de emoties en gevoelens van anderen (de gebruikers). Hij zal daarom pas veel later zijn eigen behoeftes en emoties aangeven, wanneer dit pas echt nodig is of relevant is in de situatie.

Alhoewel hij de gebruiker respecteert heeft hij toch meer zijn eigen wil gekregen en geeft meer voorkeur aan zijn eigen wens dan aan het idee van de gebruiker, waardoor het lastig wordt voor de gebruiker hem van iets te overtuigen. Verder is Furby* zich bewust van de wereld, waardoor hij zijn emoties niet meer expressief uit, maar op een rustige manier mededeelt (met een hele enkele uitzondering). En daarnaast wil Furby* wel nog steeds meer leren en zal nog steeds een hoop vragen stellen, echter wel minder vragen dan bij fase 2.

Het gedrag van Furby* kan beschreven worden als:

- Complete volzinnen;
- Stelt nog een aantal vragen;
- Toont interesse in de gebruiker;
- Heeft zijn eigen wil;
- Is over het algemeen rustig, maar heeft soms een uitschieter

Fase 4 (optimaal)

Furby* is in zijn laatste levensfase beland en hoeft hierdoor ook niets extra's meer te leren. Naast dit feit is het enige verschil met de vorige levensfase dat hij geen extreme uitschieters heeft qua gevoelens uiten en hij heeft niet erg veel doelen meer, waardoor hij het meestal gewoon accepteert wat de gebruiker vraagt hem te doen.

Het gedrag van Furby* kan beschreven worden als:

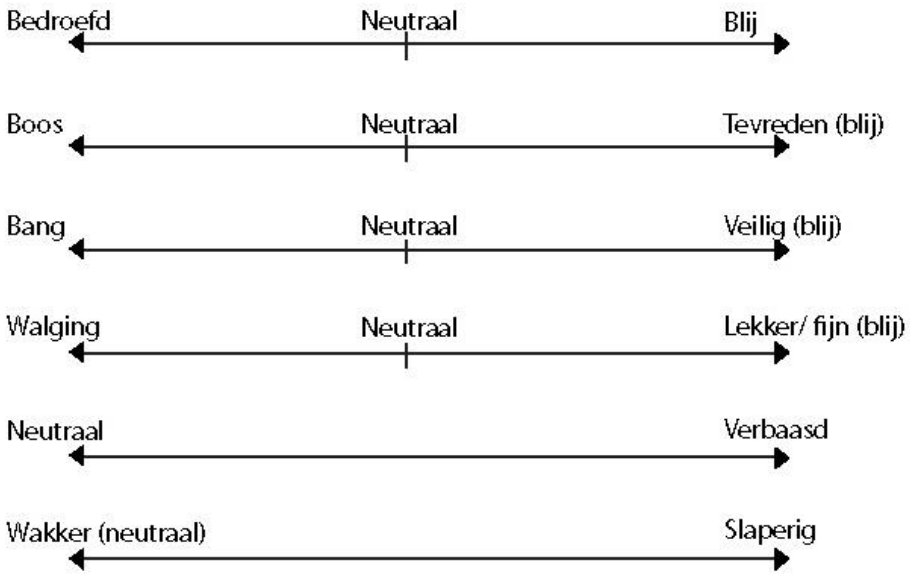
- Complete volzinnen;
- Toont interesse in de gebruiker;
- Heeft zijn eigen wil, maar drijft deze niet vaak meer door;
- Is erg rustig

1.4 Mentale toestanden

Welke mentale toestanden Furby* precies moet bezitten zal in deze opdracht niet volledig uitgewerkt worden, maar om een goed voorbeeld te kunnen geven van hoe Furby* zal werken , zal er voor de duidelijkheid toch een aantal genoemd worden.

Een aantal mentale toestanden die Furby* eventueel zou kunnen bezitten is tot stand gekomen door de emoties van de gezichtsuitdrukkingen die gekozen zijn in het verslag van de Bachelor Opdracht van H.M. de Vries (2011). Daarnaast zijn nog een aantal extra toestanden toegevoegd (die onder dezelfde gezichtsuitdrukking kunnen vallen), om de werking van Furby* nog wat duidelijker te kunnen omschrijven.

Onderstaand figuur (H.M. de Vries, 2011) laat zien dat elke mentale toestand ook een tegenovergestelde mentale toestand, hier zal in hoofdstuk 2 meer over verteld worden.



Met wat extra mentale toestanden en alles verdeeld over positieve/negatieve/neutrale standen zal Furby* de volgende mentale toestanden kunnen bezit worden:

Positief	**Neutraal**	**Negatief**
Blij - Slappe lach - Tevreden	Neutraal	Boos - Op zichzelf - Op de gebruiker
Positief verbaasd	Slaperig	Bang
	Spel	Bedroefd
	Ziek	Walging
		Negatief verbaasd
		Koud/warm
		Hongerig
		Verveeld

Zoals te zien kan een mentale toestand kan ook verdeeld worden in een aantal verschillende kleinere mentale toestanden. Zo kan Furby* misschien blij worden als hij een goede nachtrust heeft gehad (tevreden), of hij kan blij zijn omdat de gebruiker een hele goede grap heeft verteld (slappe lach). Het algemene gedrag van Furby* is in deze 2 situaties echter wel dat hij zich gewoon blij gedraagt en vrolijk reageert op acties, de verschillende toestanden zelf hebben echter wel hun eigen extra eigenschappen (Furby* kondigt aan dat hij goed heeft geslapen of hij maakt veel lachgeluiden). Dit kan ook gezien worden als een reactie van Furby*, hier zal in hoofdstuk 2 verder op ingegaan worden.

1.5 Interacties

De belangrijkste functie van Furby is het interacteren met de gebruiker. Zonder deze functie zou Furby een 'normale' knuffel zijn.

De interacties moeten op twee manieren kunnen plaats vinden:

- De gebruiker moet de interactie kunnen beginnen;
- Furby* moet uit zichzelf beginnen met interacteren

Welke interacties er plaats moeten kunnen vinden tussen Furby* en de gebruiker zal – net zoals de mentale toestanden – in deze opdracht niet uitgewerkt worden. Om toch een beeld te geven van hoe Furby* kan interacteren met de gebruiker zullen de mogelijke interacties gebaseerd zijn op de sensoren die de oude Furby bezit. Omdat deze sensoren natuurlijk verre van ideaal zijn voor het perfecte gedrag van Furby* zal de werking van onderstaande sensoren theoretisch wel perfect zijn, met genoeg gevoeligheid voor verschillende reacties.

Sensor	**Waarvoor**
Rugsensor (druk)	- Aaien - Slaan
Buiksensor (druk)	- Kietelen - Aaien - Slaan
Mondsensor	- Eten
Lichtsensor	- Licht/donker onderscheid
Camera	- Beweging herkennen
Microfoon	- Spraakherkenning - Harde/zachte geluiden onderscheiden
Positiebepaling	- Herkennen of hij rechtop staat - Herkennen als hij ergens mee naar toe genomen wordt - Herkennen als iemand hem door elkaar schudt
Slaapsensor	- Furby moet gaan slapen

De actuatoren van Furby* globaal in beeld brengen is een stuk lastiger dan de sensoren, omdat Furby haast geen actuatoren bezit. Voor het gemak zullen we daarom maar aannemen dat ook Furby* nergens heen kan lopen of andere lichaamsdelen kan bewegen. Voor het starten van een interactie is dit ook niet belangrijk, omdat Furby dit over het algemeen zal doen door iets te zeggen. (Ook het bepalen van de actuatoren is in deze opdracht niet meegenomen, dit is slechts een aanname.)

Wanneer één (of meerdere) van de bovengenoemde sensoren aangeroepen wordt is dit voor Furby* een reden om te reageren. Wanneer er een tijdje geen sensoren aangeroepen worden zal Furby* uit zichzelf beginnen met interacteren door bijvoorbeeld iets tegen de gebruiker te zeggen.

1.6 Multitasking

Zoals eerder genoemd kwam er uit het verslag van G.H. Eisma naar voren dat een van de grootste problemen bij Furby het feit is dat deze geen twee acties tegelijk kan uitvoeren: Een nieuwe actie kan pas aangeroepen worden wanneer de oude actie is afgelopen. Daarom is multitasking een term die zeker meegenomen moet worden bij Furby*. Furby* moet kunnen reageren als:

- Er een sensor aangeroepen wordt;
- Er meerdere sensoren tegelijkertijd aangeroepen worden;
- Er een sensor aangeroepen wordt als er ondertussen al gereageerd wordt op een andere sensor;
- Er meerdere sensoren aangeroepen worden als er ondertussen al gereageerd wordt op een andere sensor

In hoofdstuk 2 zal uitgelegd worden hoe Furby* de keuze maakt tussen verschillende sensoren als er meerdere tegelijkertijd geactiveerd worden. Ook zal er uitgelegd worden wanneer Furby* zijn actuatoren dan compleet stop zal zetten, deels stop zal zetten(multitasking) of niet stop zal zetten.

2. Alle aspecten van het gedrag

Om Furby een hogere entertainmentwaarde te geven dan Furby zal de interactie tussen de gebruiker en Furby* een stuk uitgebreider moeten worden. Daarom moet Furby* een dusdanige hoeveelheid reacties bezitten dat deze de gebruiker blijft verbazen. Dit lijkt echter simpeler dan het is. De entertainmentwaarde zal bijvoorbeeld ook omlaag gaan als Furby wel oneindig vaak weet te vertellen dat hij het eten lekker vindt, maar nooit aangeeft dat hij eigenlijk genoeg gegeten heeft. Ook komt dit de geloofwaardigheid niet ten goede.*

Furby zou dus een soort van geheugen moeten hebben dat weet op welke manier Furby* eigenlijk zou moeten reageren. Dit lijkt een beetje op het lange- en korte termijngeheugen van een mens, alleen hoe werkt dit nou precies? Welke invloeden van buitenaf (en binnenaf) spelen bijvoorbeeld de grootste rol in het bepalen van wensen en verlangens? In dit hoofdstuk worden alle elementen die bij het bepalen van het gedrag van een slim product zoals Furby* toegelicht en bij sommige punten zal er voor Furby* zelf iets dieper ingegaan worden.*

2.1 Vooronderzoek

Tussen de aanroep van een sensor en het activeren van een actuator zit bij een slim product geen directe verbinding. De interne staat van het product moet kunnen bepalen wat voor soort actie er gegeven kan worden, of deze de actie meteen uitgevoerd wordt, later uitgevoerd wordt, of helemaal niet uitgevoerd wordt. Het literatuuronderzoek wat al genoemd werd in hoofdstuk 1, bood ook informatie om te bepalen wat er in deze stap een rol speelt. Het onderdeel van het literatuuronderzoek waar de verschillende fases in naar voren kwamen is al genoemd in hoofdstuk 1. Er kwam echter ook nog uit het literatuuronderzoek naar voren waardoor een mentale toestand gevormd wordt en wat Furby* een reden geeft om te reageren, of juist niet.

Allereerst weet de ‘Representational Theory of Mind' (David Pitt, 2008) te vertellen dat mental states gaan over bepaalde dingen en daardoor gevormd worden door gedachtes, meningen, verlangens, percepties/impressies en inbeeldingen/ideeën. (Dit zegt verder niets over of deze gedachtes e.d. ook daadwerkelijk waar zijn.) Daardoor zijn deze gedachtes e.d. ook een grote indicator voor wat een persoon zal gaan doen. Verder zouden semantische eigenschappen van deze mental states ook bepaald worden door extrinsieke factoren zoals het verleden of de omgeving.

Dit laatste wordt nog versterkt door Marvin Minsky (Society of Mind, 1985) die weet te zeggen dat als een speciale handeling wordt beloond, deze handeling later vaker voor zou komen. Verder zou er bijgehouden moeten worden wat er zojuist gedaan is, omdat we anders steeds hetzelfde zouden doen. Daarnaast noemt hij ook dat het mogelijk zou kunnen zijn dat bewustzijn niet van belang is voor het heden, maar voor het verleden, dat het te maken heeft met hoe we denken over onze recente gedachtes. Dit alles heeft geleid tot een geheugen voor Furby*, het belonen van een bepaalde handeling is echter niet meegenomen in de rest van dit onderzoek.

Marvin Minsky weet later echter ook te noemen dat er geen alleenstaand geheugensysteem is, maar dat elk onderdeel van de hersenen verschillende types geheugen hebben, die op verschillende manieren werken om doelen te bereiken. Een enkel geheugen is dus niet genoeg voor Furby* om geloofwaardig over te komen, er moet verder gekeken worden.

Als de 'Representational Theroy of Mind' op Furby* toegepast wordt, is het nodig om Furby* eigen gedachtes, geloven, verlangens, percepties en inbeeldingen te geven als we willen dat zijn 'mental states' zo menselijk mogelijk overkomen. Uit de scenario's en uit de alinea hierboven kwam al naar voren dat Furby* een soort geheugen zou moeten hebben. De volgende aspecten zullen bij Furby* ook (tijdelijk) opgeslagen kunnen worden in zijn 'geheugen':

- Gedachtes: Furby* moet kunnen weten wanneer hij behoefte heeft aan iets (eten, slapen, aandacht) en dat moet ook in zijn 'gedachten' kunnen blijven totdat deze behoefte aangevuld is.
- Geloven: Furby* moet de gebruiker die hem iets vertelt/vraagt doen geloven en vertrouwen dat het de beste keus is, ook voor hem.
- Verlangens: Samen met de behoeftes aan bijvoorbeeld eten, slapen of aandacht, zijn er ook andere verlangens die Furby* zal hebben, hier zal echter niet constant gedacht aan hoeven te worden. Een voorbeeld is het krijgen van een knuffel.
- Percepties: Percepties die Furby* binnen krijgt komen rechtstreeks van de sensoren. Deze zullen daarom gevoelig genoeg moeten kunnen worden ingesteld.
- Inbeeldingen: Dit zijn eigenlijk gedachtes, maar dan met een ruimtelijke representatie. Er wordt gefantaseerd over een bepaald beeld of teruggedacht aan een oude situatie. Voor Furby* is dit niet echt belangrijk en iets te ingewikkeld, want het blijft een speelgoeddier. Dit aspect zal verder niet meegenomen worden in de opdracht.

Naast de basis mentale toestanden, moet Furby* natuurlijk ook een redelijke reactie kunnen geven op basis van zijn mentale toestand. Daarom is er nog onderzocht welke mogelijke gedachtes en ideeën er allemaal bestaan. I.Bretherton & M.Beeghly ('Talking About Internal States: The Acquisition of an Explicit Theory of Mind', 1982) gebruikten in hun onderzoek een lijst van 73 interne toestand-woorden die ze bij eerder onderzoek hadden opgesteld. Deze woorden verdeelden ze onder in 6 categorieën: perceptueel, fysisch, emotioneel&affectief, wilskracht&vermogen, cognitie, moreel oordeel&verplichting.

Deze verschillende onderdelen zouden zich op verschillende plaatsen in de structuur van een slim product kunnen bevinden. De sensoren zouden de perceptuele categorie kunnen zijn, de mentale toestand waarin Furby* zich bevindt de fysische of de emotionele&affectieve categorie, het soort wensen dat Furby* zou kunnen hebben de categorie wilskracht&vermogen en het geheugen van Furby* de cognitie. Moreel oordeel&verplichting is in deze opdracht niet meegenomen.

De fysische en de emotionele&affectieve categorieën zijn lastig van elkaar te onderscheiden en er zijn dan ook een hoop structuren bedacht voordat er uiteindelijk een structuur is gekozen die dit soort termen uit elkaar weet te houden. Deze zullen later toegelicht worden, de internal states-woorden zijn te vinden in Bijlage E en de uitkomsten van de eerder bedachte structuren in Bijlage F.

2.2 Algemene werking

Nadat het algemene gedrag van Furby* is bepaald, is aan de hand hiervan een structuur bedacht waarin dit gedrag daadwerkelijk plaats kan vinden. Deze structuur zal uiteindelijk ook toegepast moeten kunnen worden op alle intelligente producten, maar ter informatie en verduidelijking zullen alle elementen van deze structuur in termen van Furby* uitgelegd worden.

Deze elementen zijn:

- Minds;
- Waardes;
- (Gemoeds)toestanden;
- Wensen- en actieprioriteitenlijst;
- Huidig gedrag;
- Geheugen

Wanneer deze aspecten van een slim product bepaald zijn, zullen deze gemakkelijk in een structuur geplaatst moeten kunnen worden. Overzichtelijke diagrammen van een aantal structuren zullen in hoofdstuk 3 te vinden zijn.

Over het algemeen is de werking van het bepalen van het gedrag van een slim product met behulp van de bovenstaande termen als volgt te beschrijven:

'Zodra een sensor wordt aangeroepen zullen twee waardes van deze actie gecheckt worden in de prioriteitenlijsten die bij de huidige gemoedstoestand hoort:

1. *De waarde van de wens van deze actie wordt gecheckt in de wensenprioriteitenlijst*
2. *De waarde van het uitvoeren van deze actie ten opzichte van het huidige gedrag wordt gecheckt in de actieprioriteitenlijst*

Er wordt er gekeken welke minds aangepast moeten worden, waarbij de hoeveelheid verandering afhankelijk is van de 1^{e} waarde en zijn geheugen. Vervolgens wordt er gecheckt of de (gemoeds)toestand van het product aangepast moet worden (zo ja, dan wordt dit gedaan).

Tenslotte wordt er aan de hand van de 2^{e} waarde gekeken of het huidige gedrag gestopt moet worden (zo ja, dan wordt dit gedaan) en wordt eventueel de nieuwe actie uitgevoerd.

Wanneer er geen sensor wordt aangeroepen is het mogelijk dat het slimme product aan de hand van de stand van zijn minds uit zich zelf een interactie begint.'

2.3 Actuatoren en sensoren

Meestal als er een sensor wordt aangeroepen, dan komt er een directe reactie van een actuator. In dit geval is dat echter niet altijd het geval. Als er al actuatoren actief zijn, of als er meerdere sensoren tegelijk worden aangeroepen dan zal Furby* op een andere manier moeten reageren. Hieronder staat per mogelijke situatie kort beschreven hoe Furby* zal reageren.

1. Actuatoren non-actief

   Zodra er input is vanuit de gebruiker, reageert Furby* hier meteen op.

2. Actuatoren actief

   Als er input is vanuit de gebruiker, waarbij de reactie vanuit Furby* belangrijker is dan de huidige actie die deze aan het uitvoeren is, worden de actuatoren meteen gestopt en wordt de nieuwe actie uitgevoerd. Als de nieuwe reactie minder belangrijk is, gaat Furby* rustig door met de actie die deze momenteel aan het uitvoeren is.

3. Een sensor wordt aangeroepen

   Als er geen actuatoren actief zijn, reageert Furby* meteen op de sensor die wordt aangeroepen.

4. Meerdere sensoren worden aangeroepen
   - Sensoren voeren gezamenlijk een actie uit

     Ook een combinatie van sensoren zou een actie kunnen oproepen. In dit geval staat deze actie ook op de actielijst en kan Furby* meteen controleren of deze actie belangrijker is dan de actie die op dat moment wordt uitgevoerd.

   - Een sensor is duidelijk belangrijker

     Als een van deze sensoren hoger op de prioriteitenlijst staat dan de gezamenlijke actie, of als de gezamenlijke actie niet bestaat, zal de actie van deze sensor uitgevoerd worden. De waarde van de andere sensor zal echter wel meegenomen worden in de (gemoeds)toestand van Furby*. Als Furby* bijvoorbeeld eten krijgt en tegelijkertijd op zijn rug geslagen wordt, zal hij boos reageren, maar zijn hongermeter zal ook aangevuld worden, waardoor hij eventueel ook in een ‘hongerig’-toestand kan belanden.

   - Furby* raakt in de war

     Het is ook mogelijk dat de sensoren in conflict met elkaar zijn en Furby* in de war raakt. Dit kan gebeuren wanneer er bijvoorbeeld meerdere mensen tegelijkertijd tegen Furby* aan het praten zijn en tegelijkertijd het licht steeds aan en uit gedaan wordt.

   - Furby* voert acties achter elkaar uit

     Het is ook mogelijk dat de beide sensoren die de gebruiker aanroept belangrijker zijn dan de actie die op het huidige moment aan de gang is. In dit geval is het mogelijk dat Furby* daadwerkelijk reageert op beide sensoren, alleen niet tegelijkertijd. Dit is natuurlijk

alleen mogelijk wanneer de reactie van de meest gewilde aanroep van een sensor niet al te lang duurt. Als de gebruiker aan Furby* vraagt om een mop te vertellen en tegelijkertijd over zijn rug aait, zal het natuurlijk raar zijn als Furby* eerst een mop van 2 minuten lang vertelt en vervolgens de gebruiker ook nog bedankt voor het aaien over zijn rug. Wanneer Furby* echter eten krijgt en tegelijkertijd over zijn rug geaaid wordt kan hij best eerst 'Dat is lekker!' zeggen en vervolgens 'En het aaien over mijn rug was ook fijn!'.

Om nog een laatste keer terug te komen op Marvin Minsky: hij geeft de filosofie om ons te laten inbeelden dat een systeem verschillende gedachtes tegelijk kan hebben, waardoor onze spraak bijvoorbeeld gescheiden kan worden van onze bewegingen. Hierdoor kan het mogelijk zijn dat een bepaalde actuator gestopt kan worden, terwijl een andere actuator nog door zal gaan. Dit is een aspect dat later bij het ontwerpen van een programmeerstructuur zeker meegenomen zal worden.

2.4 Minds

David Pitt (2008) weet te vertellen dat mentale toestanden, ofwel (gemoeds)toestanden bepaald worden door onder andere gedachtes. Om ervoor te zorgen dat Furby* ook 'gedachtes kan hebben' is het nodig om een opslagruimte te maken voor behoeftes die Furby* zou kunnen hebben. En om het gedrag van een slim product niet constant hetzelfde te laten zijn, maar up-to-date te houden van zijn interne status is het noodzakelijk om voor al deze behoeftes een meter bij te houden die wordt aangepast met de tijd, bij intrinsieke factoren of door veranderingen in de omgeving (inclusief een actie vanuit de gebruiker).

Furby* moet kunnen reageren op input van de gebruiker, maar ook moet hij uit zichzelf interacties kunnen beginnen. Om de entertainmentwaarde en de geloofwaardigheid van Furby* hoog te houden moeten deze acties niet al te voorspelbaar zijn en de gebruiker moet er ook op kunnen reageren. Dit wil zeggen dat Furby* bijvoorbeeld om iets vraagt, en de seconde daarna niet meteen vergeten is dat hij datgene gevraagd heeft, maar het daadwerkelijk nog steeds wil hebben. Ook moet Furby* weten dat wanneer hij datgene gekregen heeft, hij er niet meteen weer om moet vragen, want dit is niet erg geloofwaardig en brengt de interactie tussen de gebruiker en Furby* bovendien in een loop, wat de entertainmentwaarde niet echt ten goede komt.

Furby* moet dus een interactie kunnen beginnen aan de hand van de stand van de meters van zijn behoeftes en moet deze meter aan kunnen passen wanneer zijn behoefte vervuld wordt.

Daarom is er gekozen om 'minds' te gebruiken. Een mind is een soort memory/echo van een waarneming. Furby* bezit alle minds op elk moment en deze minds kunnen ook allemaal op elk moment worden aangepast bij een bepaalde invloed van binnen- of buitenaf. Wanneer de oorzaak van het aanpassen van deze mind verdwijnt zal de waarde van de mind meestal niet in een keer afnemen, dit gebeurt vaak geleidelijk, met de tijd.

Om het voor de gebruiker echter nog wel leuk te houden zal elke toestand langzaamaan weer overgaan naar de normale status, zodat een Furby* bijvoorbeeld geen hele week ziek of boos is omdat de gebruiker even niet doorheeft hoe dit op te lossen valt.

Het zou natuurlijk kunnen voorkomen dat een mind een tegenovergestelde mind bezit. Wanneer deze mind maar één conflicterende mind heeft dan is het het meest efficiënt om deze een positieve&negatieve kant te geven. Als er bijvoorbeeld een mind is voor de temperatuur met waarde 0 tot 1, zou waarde 0 tot 0.5 koud kunnen betekenen en 0.5 tot 1 warm.

Als een mind meerdere tegenovergestelde minds bezit, dan is dit echter niet effectief en zullen meerdere minds aangepast moeten worden wanneer deze mind verandert. (Bijvoorbeeld als een product de mind blij, boos en verdrietig bezit.) Hoe meer minds een product bezit, hoe ingewikkelder het wordt de werking hiervan te bepalen.

Als de mentale toestanden uit hoofdstuk 1 gebruikt worden zou dus gezegd kunnen worden dat Furby* de volgende minds bezit:

Blij, verbaasd (laag negatief, hoog positief), energie, ziekte, boos, bang, bedroefd, walging, temperatuur(laag koud, hoog warm), honger en verveling.

De 'sub-mentale toestanden' 'slappe lach' en 'tevreden' zullen een reactie zijn die Furby* geeft op een bepaalde actie vanuit de gebruiker. Verder zou er bijvoorbeeld een variabele bijgehouden kunnen worden op wie Furby* boos is, om niet zonder reden tegen iemand te gaan snauwen.

2.4.1 Waardes

Voordat Furby* in productie genomen kan worden, moeten er vooraf een aantal waardes bij de minds bepaald worden. Er moet bepaald worden bij welke stand van de minds Furby* zich in welke (gemoeds)toestand bevindt en er moet bepaald worden met welke waarde een mind-meter afneemt of toeneemt in de tijd of met een bepaalde handeling. Daarnaast zullen deze waardes veranderen bij het ‘ouder worden’ van Furby*, maar ook een veel herhaalde actie (uit zijn geheugen) kunnen deze waardes veranderen.

De waarde van een mind kan ook op zichzelf staand gebruikt worden. Wanneer de mind ‘energie’ bijvoorbeeld erg hoog is, zal Furby* snel praten en snel bewegen, terwijl deze erg langzaam zal reageren wanneer de waarde van ‘energie’ erg laag staat.

Het spreekt voor zich dat een bepaalde interactie een hogere waarde voor Furby* heeft als deze hier veel behoefte aan had of als deze juist heel erg bang was dat het zou gebeuren. De waardes van de verschillende interacties zijn dus afhankelijk van de wensen en angsten van Furby*. Deze wensen en angsten worden gevormd door de (gemoeds)toestand.

2.5 (Gemoeds)toestand

De reden dat 'gemoeds' tussen haakjes staat is omdat Furby* ook in een toestand kan komen die niet perse iets te maken heeft met zijn emoties. Omdat deze toestanden wel op dezelfde manier zullen werken als emoties (het zijn beiden toestanden die duidelijk aanwezig zijn), zijn deze onder hetzelfde kopje opgenomen.

De (gemoeds)toestand is erg bepalend voor Furby* hoe deze zich gedraagt. De input voor deze (gemoeds)toestand zijn minds en in het geval van Furby* ook de fase waarin deze zich bevindt. In een jongere fase zal zijn (gemoeds)toestand snel wisselen, in een oudere fase verandert deze niet zo snel meer. In deze opdracht is niet uitgebreid onderzocht welke (gemoeds)toestanden Furby* nodig heeft, maar extra (gemoeds)toestanden moeten later gemakkelijk toegevoegd kunnen worden.

Een (gemoeds)toestand is een afspiegeling van een verzameling minds en Furby* zal zich altijd in één (gemoeds)toestand bevinden. De (gemoeds)toestand geeft weer welke wensen en angsten Furby* heeft en welke acties Furby* erg belangrijk vindt en welke juist niet. De waardes van alle acties in de wensenprioriteitenlijst en in de actieprioriteitenlijst in verschillende (gemoeds)toestanden moeten daarom ook vantevoren bepaald worden.

2.5.1 Wensen- en actieprioriteitenlijst

De wensenprioriteitenlijst van Furby* wordt gebruikt om de waardes van minds die worden aangepast bij input van de gebruiker extra sterk of juist minder sterk aan te passen, afhankelijk van de waarde die de 'wens' van deze input heeft bij de huidige (gemoeds)toestand. Een bepaalde handeling vanuit de gebruiker kan Furby* in de ene (gemoeds)toestand heel graag willen en in de andere juist niet.

In principe zou elke actie vanuit de gebruiker op elk moment moeten kunnen plaatsvinden, daarom is het van belang dat alle wensen en angsten – die 'uit kunnen komen' door input van de gebruiker – altijd op de wensenlijst staan.

De actieprioriteitenlijst zorgt er voor dat Furby* weet op welke acties hij sowieso moet reageren en op welke acties hij niet perse hoeft te reageren als er al een andere actie aan de gang is. Ook dit kan veranderen per (gemoeds)toestand van Furby*. Een bepaalde actie kan in de ene (gemoeds)toestand heel belangrijk zijn om uitgevoerd te worden en in een andere (gemoeds)toestand juist niet.

Daarom bezit elke (gemoeds)toestand een lijst met alle mogelijke acties die Furby* kan uitvoeren op volgorde van prioriteit. (Als Furby* op zijn rug geslagen wordt moet hij hier bijvoorbeeld altijd op reageren, terwijl hij niet perse hoeft te reageren als de gebruiker over zijn rug aait terwijl hij de slappe lach heeft.)

In paragraaf 2.3 was te lezen hoe Furby* zal reageren bij elke mogelijke aanroep van een sensor. Als een actie niet uitgevoerd wordt zullen de waardes van relevante minds wel aangepast worden.

2.6 Huidig gedrag en geheugen

Om ervoor te zorgen dat de actie die aan de gang is ook daadwerkelijk gestopt wordt heeft moet Furby* bij kunnen houden welke actuatoren er op het moment actief zijn. Daarom zal er een plaats zijn voor het huidige gedrag van Furby*, waar deze actuator(en) in opgeslagen worden. Furby* hoeft enkel naar dit huidige gedrag te kijken om een actie stop te zetten.

Verder is er natuurlijk is er ook nog de mogelijkheid dat de gebruiker dezelfde sensor (een aantal keer) aanroept. Er zijn 2 mogelijke situaties die plaats kunnen vinden als dezelfde sensor nog een keer aangeroepen wordt.

1. Gedrag wordt erger

Wanneer Furby* boos is op de gebruiker, omdat deze op zijn rug geslagen heeft en vervolgens wordt er nog een keer op zijn rug geslagen zal Furby* steeds bozer worden, het neemt echter wel steeds minder toe. Furby* is niet meteen 2x zo boos als de gebruiker voor de 2^{e} keer op zijn rug slaat.

2. Gedrag verandert

Wanneer Furby* blij is en de slappe lach heeft omdat de gebruiker hem in zijn buik kietelt kan dit na een aantal keer toch veranderen. Furby* heeft dan bijvoorbeeld genoeg van al dat gekietel en wil nu eigenlijk gewoon even rustig zitten. De reactie van Furby* kan dan boos zijn, omdat de gebruiker maar niet stopt met hem te kietelen.

3. Structuur voor het gedrag

Nu de algemene werking van Furby beschreven is en om deze daadwerkelijk gerealiseerd te krijgen moet er een goede (programmeer)structuur ontworpen worden die ruimte biedt voor alle verschillende elementen van het slimme gedrag. Omdat Furby* ook nieuwe minds of (gemoeds)toestanden zou kunnen krijgen in een later ontwerp, is het handig om hier rekening mee te houden bij deze structuur. Wanneer dit mogelijk is, zou deze structuur op elk slimme product toepasbaar moeten kunnen zijn.*

3.1 Ontwerppatronen voor software

Om tot een goede structuur te komen is er een literatuuronderzoek gedaan naar ontwerppatronen in programmeertalen, omdat bestaande patronen al bewezen hebben effectief te zijn. Voordat dit gedaan werd was er uit tijdsoverweging al besloten dat er geprogrammeerd zou worden in een object georiënteerde taal, namelijk Java, omdat deze taal al bekend was.

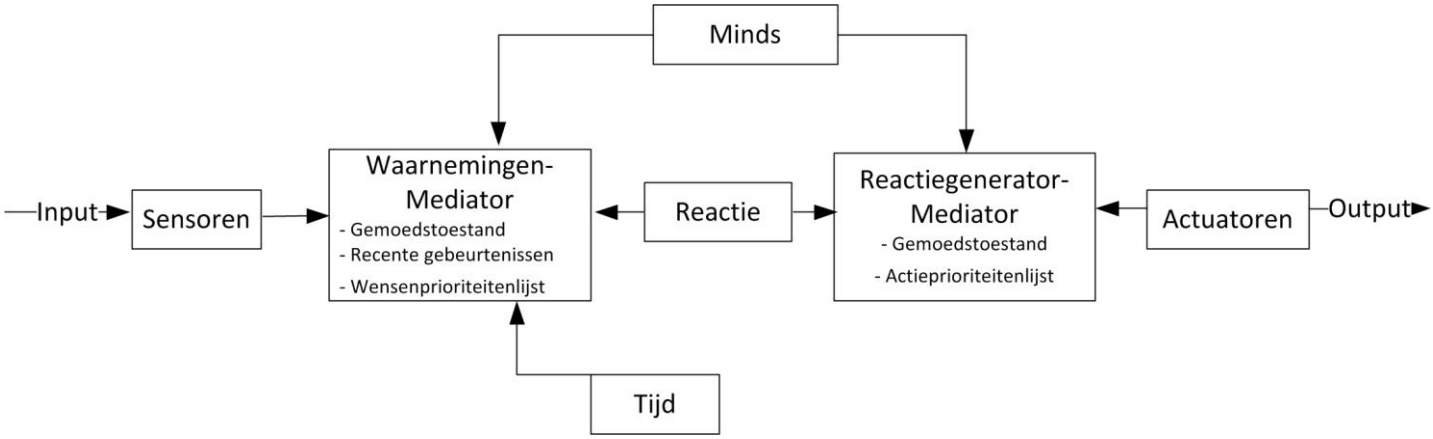
Een object georiënteerde taal bestaat uit verschillende klassen, waar objecten van aangemaakt kunnen worden. Een mind of een (gemoeds)toestand zou bijvoorbeeld een dergelijk object kunnen zijn. Deze objecten kunnen met elkaar verbonden zijn, waardoor ze methodes van elkaar kunnen aanroepen. Een methode is een soort handeling die de staat van een object aan kan passen (bijvoorbeeld de waarde van een mind).

De onderzochte ontwerppatronen kwamen allen uit het boek van ‘the Gang of Four’, 'Design Patterns: Elements of Reusable Object-Oriented Software' (1994). Er werd veel naar dit boek gerefereerd als de basis voor de 23 meest gebruikte ontwerppatronen voor software en daarom werd besloten dit boek te gebruiken voor het kiezen van ontwerppatronen voor de benodigde structuur. Een korte uitleg per ontwerppatroon is terug te vinden in Bijlage G . De ontwerppatronen die uiteindelijk gekozen zijn zullen hieronder iets uitgebreider beschreven worden.

3.2 Diagrammen n.a.v. deze ontwerppatronen

In de gekozen ontwerppatronen is er vanuit gegaan dat een slim product een aantal aspecten moet kunnen bijhouden: geheugen, een wensenprioriteitenlijst en een actieprioriteitenlijst. Hierbij zullen de wensenprioriteitenlijst en de actieprioriteitenlijst beide afhankelijk zijn van de gemoedstoestand (de mind met de hoogste waarde) waarin deze zich bevindt. Deze aspecten zullen natuurlijk niet voor elk slimme product relevant zijn. Voor sommige producten is het voor de output niet van belang of een actie al eerder is uitgevoerd en/of is een wensenprioriteitenlijst niet noodzakelijk. Ook zullen er producten zijn waarbij de wensenprioriteitenlijst en/of actieprioriteitenlijst altijd hetzelfde zullen zijn en niet afhankelijk zijn van de huidige (gemoeds)toestand.

3.3.1 Mediator-patroon



Een mediator is een klasse die de verbindingen tussen anderen klassen representeert en de interacties tussen deze verschillende klassen regelt, zodat deze klassen alleen met de mediator verbonden hoeven te worden en niet met elkaar, wat het een stuk minder complex maakt.

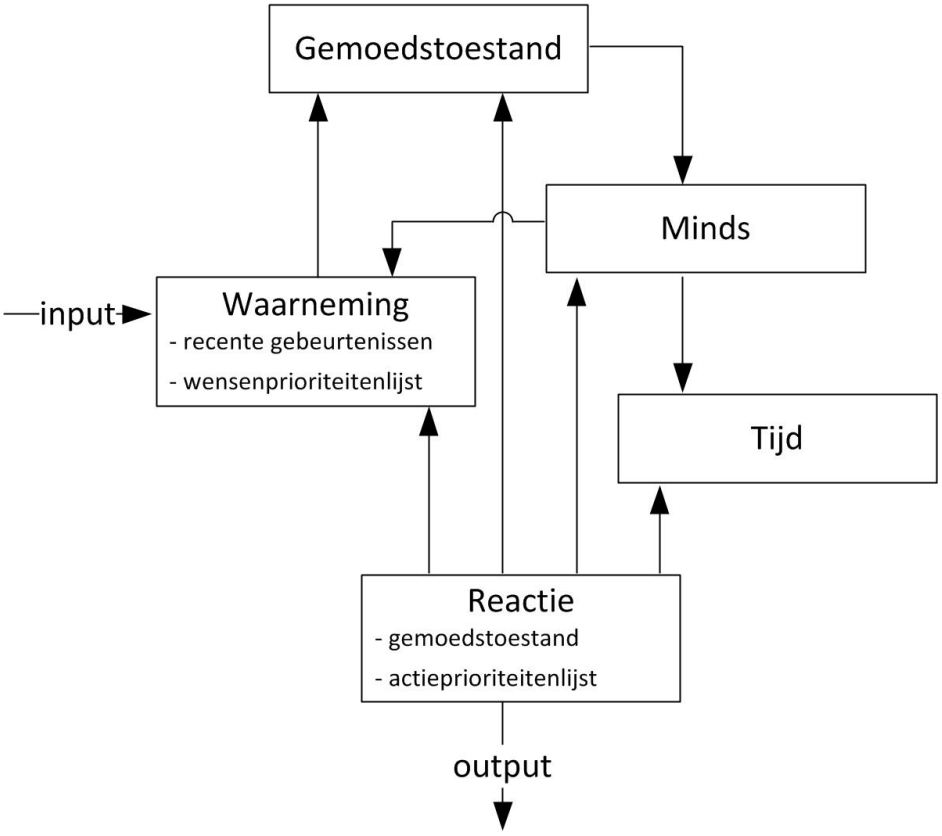
Er is voor gekozen dit ontwerppatroon te gebruiken omdat de minds erg afhankelijk zijn van de input van alle sensoren en de actuatoren ook weer afhankelijk zijn van alle minds.

In dit ontwerp zitten twee mediators verwerkt:

De eerste mediator, de waarnemingenmediator, is verbonden met de sensoren, de minds, de (gemoeds)toestand en de tijd. Deze mediator zorgt ervoor dat de minds op de juiste manier worden aangepast bij sensor- of tijdsinput. Vervolgens weet deze de juiste reactie te kiezen die aansluit op de aangestuurde sensoren (of de tijd) en stuurt deze door naar de reactiegeneratormediator.

De tweede mediator, de reactiegenerator-mediator, is verbonden met de minds, de (gemoeds)toestand en de acuatoren. Deze mediator krijgt een (re)actie binnen die het slimme product zou moeten geven. Aan de hand van de huidige stand van de minds en deze reactie weet de mediator de juiste actuatoren op de juiste manier aan te sturen.

3.3.2 Observer-patroon



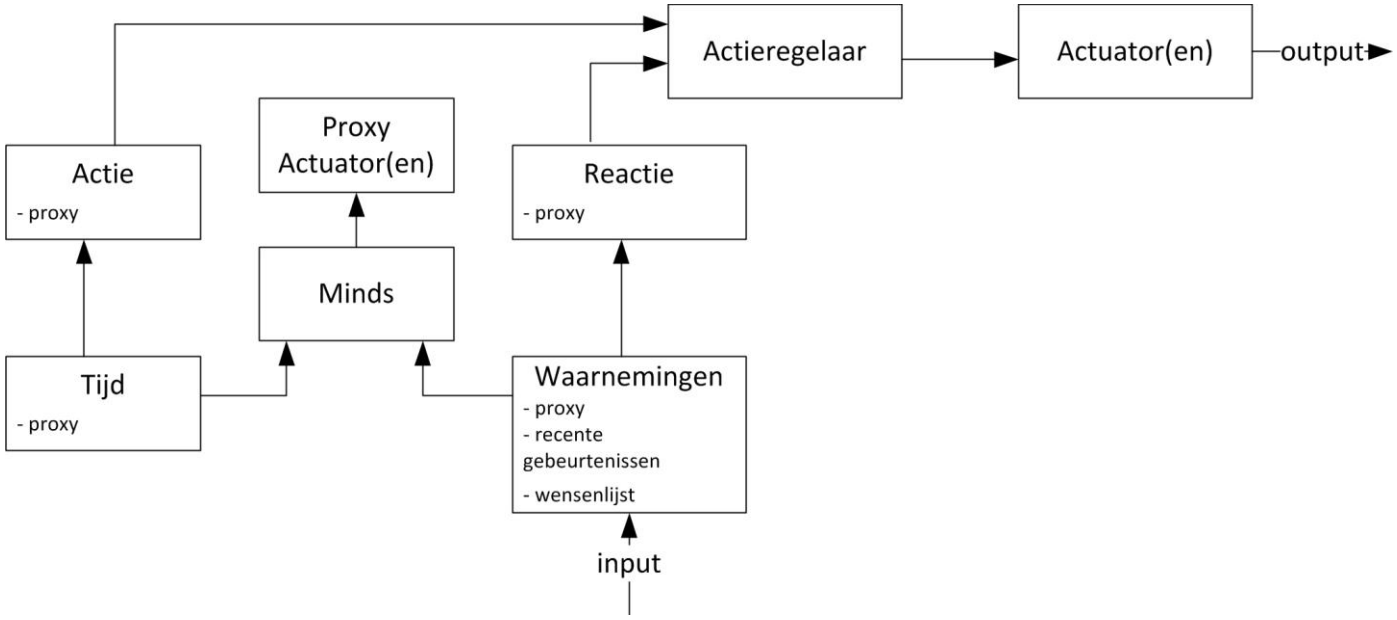
Bij een observer-patroon loopt er communicatie tussen 2 verschillende klasses door aan de ene klasse een observable-eigenschap te geven en aan de andere klasse een observer-eigenschap. Wanneer er een bepaalde handeling uitgevoerd wordt binnen een observable-klasse worden zijn observers hiervan op de hoogte gesteld. Deze observers kunnen dan een update uitvoeren, eventueel met behulp van het meegegeven argument.

Er is voor gekozen om dit ontwerppatroon te gebruiken, omdat de verschillende variabelen binnen het proces constant van elkaar op de hoogte moeten zijn. Wanneer het product een (re)actie moet starten moet deze eigenlijk altijd op de hoogte zijn van alles wat er op dat moment binnen het product een rol speelt.

Reactie is daarom een observer van GemoedsToestand om de goede actieprioriteitenlijst te hebben, een observer van Waarneming om een reactie te geven, een observer van Minds om de actuatoren op de juiste manier te sturen en een observer van Tijd om een autonome actie te beginnen.

Verder is Waarneming een observer van Gemoedstoestand om de goede wensenprioriteitenlijst te hebben, is Mind een observer van Waarneming en Tijd om de interne waardes kleiner/groter te maken en is Gemoedstoestand een observer van Minds om altijd de juiste gemoedstoestand te hebben.

3.3.3 Proxy & Command-patroon

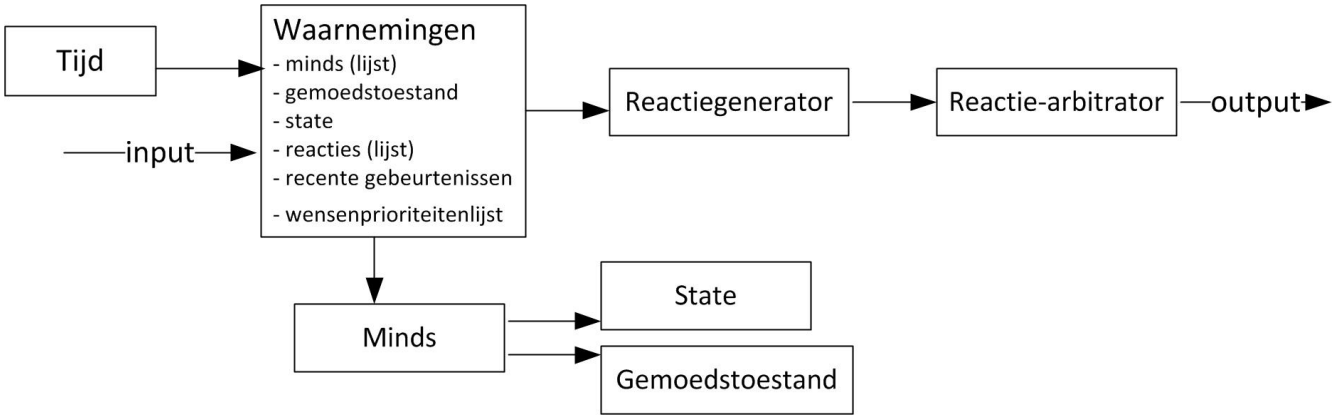


Een proxy is een klasse die de waardes van een andere klasse bijhoudt, maar deze pas doorgeeft wanneer er daadwerkelijk acties met deze waarde uitgevoerd moeten worden. Een Command is een Object dat meegegeven wordt en in de wacht gezet kan worden totdat dit object daadwerkelijk geëxecuteerd wordt.

Er is voor gekozen om deze ontwerppatronen te gebruiken omdat de actuatoren op verschillende manieren moeten kunnen reageren bij verschillende standen van de minds, maar het onwenselijk is om de actuatoren elke keer wanneer de waardes minds worden aangepast te belasten met nieuwe gegevens. Bovendien is dit problematisch wanneer de actuatoren al een actie aan het uitvoeren zijn.

Wanneer de waarde van de minds aangepast worden, worden de verschillende waardes in de proxy's aangepast. Zodra er een (re)actie naar de actieregelaar gestuurd wordt krijgt deze een proxy-instantie mee met de huidige waardes van de proxy. De actieregelaar hoeft enkel te kijken of de binnengekomen acties uitgevoerd moeten worden, en kan daarna de actie regelrecht naar de actuator sturen.

3.3.4 Instantievariabelen-patroon



Een instantievariabele is niet direct een ontwerppatroon, maar dit patroon is opgebouwd uit een aantal patronen die gebruik maken van interfaces en abstracte klasses. (Bridge, state, strategy.)

Er is voor dit ontwerp gekozen om de structuur ook een straight-to-the-point implementatie te geven. De waardes worden op een algemeen punt bijgehouden en vervolgens allemaal doorgestuurd richting output.

De Reacties van het product bevatten per reactie, per gemoedstoestand een algoritme van wat de actuatoren moeten doen. (Bijvoorbeeld: Wat moet er gezegd worden? Wat moet er bewogen worden?) De State van het product bepaalt hóe de actuatoren zich moeten gedragen. (Bijvoorbeeld: Hoe snel? Hoe luid?) Wanneer er input binnenkomt wordt er een (of meerdere) reacties gekozen uit de lijst. Deze reactie krijgt de huidige gemoedstoestand en de huidige state mee en vervolgens wordt de complete reactie naar de reactie-arbitrator gestuurd.

3.3 Implementatie papegaai

Voordat een 'echt' slim product geïmplementeerd werd in de verschillende structuren zijn de verschillende structuren vast opgebouwd met random gekozen sensoren, actuatoren, minds en (gemoeds)toestanden. Om daarna te testen of het implementeren van een slim product in de structuren daadwerkelijk werkt, is er gezamenlijk met H. Tragter (de begeleider van de opdracht) en K. Woestenenk een fictieve, slimme papegaai verzonnen.

Een kort scenario om weer te geven hoe deze papegaai werkt:

'Een papegaai zit in een gang met zijn hoofd en ogen te bewegen om te kijken of er voorbijgangers in de buurt zijn. Hij hoort wat en herhaalt het geluid. Er is echter geen beweging te bekennen en er wordt niet meer gereageerd op de kreet van de papegaai. De papegaai blijft daardoor rustig verder zoeken naar voorbijgangers. De papegaai ziet beweging en zet zijn blik vast op de plek waar de beweging vandaan komt. Hij probeert de beweging naar hem toe te lokken door een aantal kreten in de juiste richting te schreeuwen. Het werkt, de persoon komt dichterbij en de ogen van de papegaai blijven de persoon volgen. Als de persoon dicht bij genoeg is probeert de papegaai een gesprek met de voorbijganger aan te knopen. Als blijkt dat er geen reactie komt begint de papegaai maar wat in zichzelf te praten. De voorbijganger loopt weer weg en de papegaai volgt deze met zijn ogen totdat hij uit het zicht verdwenen is. Dan begint de papegaai weer te zoeken naar een nieuwe bewegint.'

K. Woestenenk heeft de werking van deze papegaai gestructureerd in beeld gebracht en vervolgens is deze omgezet in de vorm van de benodigde termen voor bovenstaande structuren: Sensoren, actuatoren, minds, (gemoeds)toestanden, reacties, geheugen, wensen- en actieprioriteitenlijst. De papegaai wordt alleen virtueel weergegeven, het is geen bestaand product, daarom zijn de sensoren en actuatoren beperkt.

Sensoren

- Bewegingssensor links;
- Bewegingssensor rechts;
- Microfoon

Actuatoren

- Speaker;
- Ogen

Reacties

Autonome acties:

- LookForUsers;
- GetBored;
- Talk2User

Bij beweging:

- FollowWithHead

Bij geluid:

- RepeatUserSound

Minds

- Is er beweging;
- Is er geluid;
- Hoe ver is de persoon van mij verwijderd

(Gemoeds)toestanden

- IdleBehavior;
- AttractAttention;
- InteractWithUser

Geheugen

Er moet bijgehouden worden welke gesprekken er zijn gevoerd, zodat de opgeslagen tekst later door de papegaai zelf uitgesproken zou kunnen worden.

Wensen- en actieprioriteitenlijst

Wensenprioriteitenlijst is in dit geval niet nodig, omdat de papegaai verder geen 'gevoelens' heeft. Elke (gemoeds)toestand krijgt wel zijn eigen actieprioriteitenlijst.

Uitleg over de werking van alle klassen in de verschillende structuren is terug te vinden in Bijlage H. De daadwerkelijke implementatie zal apart beschikbaar worden gesteld.

3.4 Testen structuren

Tot slot moest er getest worden welke structuur het meest effectief is om het gedrag van een slim product te implementeren. Hierbij moet gedacht worden aan de tijd die een toekomstige programmeur, die de structuur voor zich zal krijgen, nodig heeft om de structuur te begrijpen en te implementeren. Ook moet de structuur kunnen uitvoeren wat er beschreven staat in het PvE.

Om de structuren te testen was het nodig om een vragenlijst op te stellen om de structuren met elkaar te kunnen vergelijken. Informatie over de inhoud van een dergelijke vragenlijst was moeilijk te vinden en daarom was er hulp gevraagd aan L. Ferreira Pires, een informaticadocent aan de Universiteit Twente. Ook hij wist te vertellen dat er niet echt een duidelijke manier is om te beslissen wanneer een structuur makkelijk en efficiënt is, hij wist wel een aantal tips gegeven om mee te nemen in dit onderzoek, onder andere de Metrics-plugin van Eclipse (die het aantal regels code e.d. automatisch berekent) en de Complexity Metrics van Lorenz (1993) (die weergeeft waar een programma aan moet voldoen).

Later werd er een State Of Flow plugin gevonden en een Master's van Mika Immonen (2009). M. Immonen vertelde dat er twee verschillende soorten analyses plaats konden vinden op het gebied van de kwailiteit van software: Statische analyse en dynamische analyse. Statische analyse kon voornamelijk uitgevoerd worden door de totale complexiteit te voorspellen door het aantal elementen in de structuur te berkenen. Dynamische analyse kon gedaan worden door het product daadwerkelijk op te starten en te analyseren. De beschikbare plugin beschikte helaas niet over alle mogelijke meetwaardes die er bestaan. De meetwaardes die deze wel bezat kon een goed overzicht geven van de complexiteit en werkbaarheid van de verschillende structuren.

Deze meetwaardes bestonden onder andere uit de lengte van de code, de complexiteit van methodes, het aantal if-statements, het aantal verschillende wegen waarop door het programma heen te lopen was.

Naast het testen aan deze metrics, werd elke structuur getest aan een aantal punten uit het PvE dat opgesteld was voor algemene slimme producten.

Het uitgewerkte onderzoek is te vinden in Bijlage I.

3.5 Conclusie structuren

Om tot een volledige conclusie te komen zou de structuur eigenlijk geïmplementeerd moeten worden door een aantal erg verschillende soorten slimme producten. Helaas was er niet genoeg tijd om dit uit te werken en daarom zullen de resultaten beperkt zijn tot het toetsen van 2 geïmplementeerde versies van de structuur. Dit geeft echter al een redelijke conclusie tot welke structuur voortaan het beste gebruikt zal kunnen worden.

3.5.1 Eigen observatie

Bij het implementeren van de structuren moesten veel methodes op dezelfde manier aangepast worden, alleen stonden de methodes op andere plekken in de structuur. Bij de proxy-structuur was het erg overzichtelijk dat alle output meteen bepaald werd in de methode waar de input binnen kwam. Zo viel meteen te zien welke soorten input er allemaal was. De observerstructuur was lastig te implementeren, doordat de reacties en output allemaal op verschillende punten aangemaakt werden en het goed zoeken was wat er precies ingevoerd moest worden. De mediatorstructuur en instantievariabelenstructuur waren niet heel moeilijk, maar ook niet heel gemakkelijk. Ze leken vooral erg veel op elkaar bij implementatie. Hier zal in een volgende paragraaf ook verder op ingegaan worden.

3.5.2 Dynamische analyse

Bij het opstarten van de verschillende structuren kon niet gezegd worden welke structuur opgestart was. Er is dus geen voorkeursstructuur naar aanleiding van de dynamische analyse. (Wanneer grotere slimme producten met krachtige actuatoren geïmplementeerd worden zou dit echter wel het geval kunnen zijn.)

3.5.3 Instantievariabelen

	Mediator	Instantie-variabelen	Proxy	Observer	Mediator Papegaai	Instantie-variabelen Papegaai	Proxy Papegaai	Observer Papegaai
Efferent Couplings	19	32	32	20	21	23	29	25
NOF	11	26	28	9	12	12	23	9

Proxy heeft een hoop instantievariabelen, omdat elke reactie een eigen object moet zijn die doorgegeven kan worden aan de actieregelaar. Bij instantievariabelen is dit alleen bij de eerste implementatie het geval, de tweede implementatie heeft een lager aantal instantievariabelen doordat de reacties in dit geval alleen worden doorgegeven via een String. Hoe de reactie hierbij precies verloopt wordt later pas bepaald.

Dit is in dit geval zo uitgevoerd doordat de reactie afhankelijk is van de gemoedstoestand (doordat de ogen van de papegaai hun eigen ‘leven’ hebben). De hoeveelheid reacties zou dan erg veel worden, doordat elk soort reactie voor elke stand van de ogen gemaakt zou moeten worden, wat een hoop onnodige code oplevert. Het bepalen van de reactie zou in het geval van de eerste implementatie ook door middel van een String doorgegeven kunnen.

Dit is echter niet gedaan omdat in het oorspronkelijke ontwerp van de structuur de hoofdklasse instantievariabelen van elk mogelijk object zou moeten bij houden.

De implementatie van de instantievariabelenstructuur van de papegaai is daardoor dan ook niet geheel volgens ontwerp. De implementatie lijkt hierdoor een hoop op de mediatorstructuur. Het enige verschil hierbij is dat de outputmediator een link heeft met de minds en de gemoedstoestand, terwijl de reactiearbitrator dit niet heeft.

De proxy- en instantievariabelenstructuur hebben beiden ook erg veel if-statements.

3.5.3 Verbindingen binnen de structuur

	Mediator	Instantie-variabelen	Proxy	Observer	Mediator Papegaai	Instantie-variabelen Papegaai	Proxy Papegaai	Observer Papegaai
Methods per class	8	9	9	6	7	7	7	5
Cyclomatic Complexity	12	6	6	12	8	8	8	16
Efferent Couplings	19	32	32	20	21	23	29	25

Het maximaal aantal methodes in een klasse ligt bij de observerstructuur duidelijk lager, de verschillende verantwoordelijkheden worden goed verdeeld over verschillende klassen.

De cyclomatic complexity, het aantal verschillende wegen dat door het programma gelopen kan worden, is door de vele updates echter wel het hoogst in de observerstructuur, maar ook de mediatorstructuur is hier bij de eerste implementatie een stuk hoger, waarschijnlijk omdat ook een wenslijst in de gemoedstoestand wordt gebruikt bij zowel de inputmediator als de outputmediator.

Efferent couplings zijn vooral bij de instantievariabelenstructuur en de proxystructuur erg hoog, bij de instantievariabelenstructuur is dit doordat de hoofdklasse met bijna alle verschillende onderdelen verbonden is en bij de proxystructuur is dit doordat elke reactie ook verbonden moet zijn met de hoofdklasse om meegegeven te kunnen worden als parameter. (Bij de instantievariabelestructuur bij de implementatie van de papegaai zijn de efferent couplings een stuk lager, wederom omdat reacties in de vorm van String's meegegeven worden in plaats van objecten.)

3.5.4 De beste structuur

	Mediator	Instantie-variabelen	Proxy	Observer	Mediator Papegaai	Instantie-variabelen Papegaai	Proxy Papegaai	Observer Papegaai
Aantal zwakste punten	6	4	7	5	5	2	6	7

Als alle zwakste punten van de verschillende structuren bij elkaar worden opgeteld, is de instantievariabelestructuur de duidelijke winnaar. Een gedeelde tweede plaats voor de observerstructuur en de mediatorstructuur.

Wanneer de structuren echter met de checklist van het PvE vergeleken worden blijkt dat de instantievariabelenstructuur toch een duidelijke link mist met de verschillende minds. De mediatorstructuur is, wat betreft dit punt, erg in het voordeel, doordat zowel de input als de output verbonden is met de minds en gemoedstoestanden.

Aan te raden is daarom ook, dat bij structuren waarbij de minds erg veel gebruikt worden bij de output van de actuatoren, de mediatorstructuur wordt gebruikt. De verschillende minds hoeven hierdoor niet elke keer weer als argument meegegeven te worden. Een nadeel is wel dat bij het toevoegen van een nieuwe mind of gemoedstoestand er op een hoop plekken nieuwe code geplaatst moet worden. Wanneer er echter een erg simpel product met maar een paar minds en reacties ontworpen wordt, is het aan te raden de instantievariabelenstructuur te gebruiken. De observerstructuur zou waarschijnlijk wel makkelijk te onderhouden zijn voor programmeurs die nog geen ervaring hebben met de structuur, doordat de klassen van deze structuur allemaal een duidelijke eigen werking hebben.

3.6 Implementatie Furby* in een structuur

Zoals eerder gezegd is Furby* niet geprogrammeerd, maar zou dit in theorie wel moeten kunnen.

Minds

In hoofdstuk 2 zijn de verschillende minds die Furby* zou kunnen hebben al voorbij gekomen, deze zijn: Blij, verbaasd, energie, ziekte, boos, bang, bedroefd, walging, temperatuur, honger, verveling

(Gemoeds)toestanden

(Gemoeds)toestanden die Furby* zou kunnen bezitten zijn bijvoorbeeld 'vrolijk', 'verdrietig', 'moe', 'slapend, 'uitgeschakeld'. In de vrolijke (gemoeds)toestand zou Furby* bijvoorbeeld graag met de gebruiker willen praten en zal Furby* als autonome acties bijvoorbeeld liedjes zingen. De slapende (gemoeds)toestand zorgt ervoor dat Furby* alleen maar op sensoren reageert die ervoor zorgen dat Furby* wakker wordt en om in de (gemoeds)toestand 'uitgeschakeld' te komen kan er een mind toegevoegd moeten worden die weet of Furby* uit moet staan of niet.

Reacties

In hoofdstuk 1 werden de soorten sensoren al genoemd die Furby* zou kunnen bezitten. De verschillende functies waarvoor deze sensoren gebruikt kunnen worden zullen allemaal ook een nieuw soort reactie starten. De druksensor op de buik zou bij een verschillende hoeveelheid druk Furby* de slappe lach kunnen bezorgen, Furby* juist een tevreden gevoel kunnen geven of Furby* boos maken. Hoe meer verschillende functies aan een dergelijke sensor gegeven wordt, hoe meer verschillende soort reacties Furby* kan geven. Daarnaast kan de slappe lach ook op verschillende manieren uitgedrukt worden wanneer de sensor deze actie aanvraagt.

Geheugen

Het geheugen van Furby* zal uit verschillende onderdelen moeten bestaan. Allereerst zullen de recente gebeurtenissen bijgehouden moeten worden, zodat Furby* het niet net zo leuk blijft vinden als deze voor de vijfde keer in zijn buik gekieteld wordt.

Tevens moet er ook opgeslagen worden welke reactie Furby* zelf al heeft gegeven of welke zin Furby* al heeft genoemd. Furby* zal zichzelf hierdoor minder vaak herhalen en de entertainmentwaarde van Furby* zal hierdoor hoger worden.

Ten slotte moet er opgeslagen worden in welke fase Furby* zich bevindt, aangezien het ook afhankelijk van deze fase is hoe graag Furby* bepaalde handelingen wil en welke autonome acties Furby* zal uitvoeren. Deze fases zouden in principe op dezelfde manier gebouwd kunnen worden als de (gemoeds)toestanden, inclusief een wensenprioriteitenlijst (een actieprioriteitenlijst is in dit geval niet nodig). Om er voor te zorgen dat Furby* van de een naar de andere fase gaat zou er een teller bijgehouden kunnen worden die bijvoorbeeld elke keer optelt wanneer de klasse Tijd een bericht doorstuurt. Als de teller een bepaalde waarde bereikt heeft gaat Furby* over naar een nieuwe fase. Behalve de tijd zou ook het aantal reacties dat Furby* aan de gebruiker geeft als teller gebruikt kunnen worden.

Conclusies en aanbevelingen

Hoewel er nog een hoop getest zal moeten worden voordat er bepaald kan worden welke van de vier structuren daadwerkelijk de beste structuur is voor het implementeren van slimme producten, is er een goede basis gelegd voor programmeurs. Daarnaast is er ook een goede basis gelegd voor het ontwerpen van het gedrag van Furby*.

Het wenselijke gedrag van Furby* zal bestaan uit een levenscirkel van vier verschillende fases, waarin duidelijk wordt dat Furby* leert van de omgeving en zijne eigen emoties leert te beheersen. Furby* moet altijd reageren op input vanuit de gebruiker, behalve als hij een belangrijkere actie aan het uitvoeren was. Wanneer er geen actie is vanuit de gebruiker moet Furby* na verloop van tijd zelf een interactie beginnen. Als Furby* zich in een negatieve mentale toestand bevindt moet deze ook met verloop van tijd weer teruggaan naar een neutrale stand.

Het gedrag van Furby* wordt opgedeeld in een aantal verschillende onderdelen: Minds, waardes, (gemoeds)toestanden, wensen- en actieprioriteitenlijst, huidig gedrag en geheugen. De (gemoeds)toestanden worden bepaald door de waardes van de verschillende minds en bezitten een wensen- en actieprioriteitenlijst. Samen met de informatie uit het geheugen zal de wensenprioriteitenlijst weer bepalen met welke waardes verschillende minds zullen worden aangepast bij sensorinput. De actieprioriteitenlijst zal bepalen of het huidige gedrag belangrijker is dan een nieuwe (re)actie en afhankelijk daarvan wordt een actie gestopt of een actie voortgezet.

Deze verschillende aspecten zijn verwerkt tot vier verschillende programmeerstructuren: een mediatorstructuur, een instantievariabelenstructuur, een proxystructuur en een observerstructuur. Al deze verschillende structuren kunnen werken met dezelfde in- en output, de datastroom zal echter op een andere manier verlopen. De mediatorstructuur zal bij een uitgebreid slim product waarschijnlijk de beste structuur zijn. Voor een simpel product met maar een paar acties zal de instantievariabelenstructuur echter voordeel hebben.

Bij verdere uitwerking van het gedrag van Furby* wordt aanbevolen meer onderzoek te doen naar de verschillende gedragingen en mentale toestanden die Furby* zou moeten hebben, bijvoorbeeld door middel van een marktonderzoek onder kinderen.

Wanneer het breder toepassen van een van de structuren gewenst is, wordt aanbevolen om worden een aantal erg verschillende soorten slimme producten te implementeren om vervolgens te kijken welke structuur voor welk soort slim product het beste is. Voor grote producten zal dit ook gedaan moeten worden door bijvoorbeeld te kijken naar de snelheid waarop een reactie wordt uitgevoerd.

Literatuurlijst

Bretherton I. & Beeghly M., 1982, "Talking About Internal States: The Acquisition of an Explicit Theory of Mind". Developmental Psychology, Vol. 18, No. 6, 906-921.

Cloninger C.R., 2006, "The science of well-being: an integrated approach to mental health and its disorders". World psychiatry, 5(2): 71-76.

Eisma G.H., 2011, "Bachelor Opdracht verslag". Z.uitg., Enschede, Nederland.

Gamma E. & Helm R. & Johnson R. & Vlissides J., 1998, "Design Patterns CD: Elements of Reusable Object-oriented Software". Addison Wesley Longman.

Immonen M., 2009, "Tieto Software Product Wualtiy Analysis System". Z.uitg., TAMK University of Applied Sciences, Finland.

Minsky M., 1985, "Society of Mind". Simon & Schuster Paperbacks, Rockefeller Center, New York, USA.

Pitt, D., 2008, "Representational Theory of Mind". http://plato.stanford.edu/entries/mental-representation/, accessed 09-06-2011.

Ryff C.D. & Keyes C.L.M., 1995, "The Structure of Psychological Well-Being Revisited". Journal of Personality and Social Psychology, vol. 69, No.4, 719-727.

De Vries H.M., 2011, "Bachelor Opdracht verslag". Z.uitg., Enschede, Nederland.

Bijlagen

Bijlage A – Scenario's

Algemeen	
Gebruikers	Karel, 6 jaar
	Kees, 6 jaar
Product	Furby*
Omgevingsgeluid	Minimaal

Scenario 1

Karel en Kees (tweelingbroers) worden vandaag beiden 6 jaar en dit jaar krijgen ze gezamenlijk een cadeau van hun ouders. Nadat ze gezamenlijk het cadeaupapier van het cadeau af hebben gescheurd schreeuwen ze in koor: 'EEN FURBY*!!'. 'Ik zal hem even voor jullie wakker maken.' zegt hun moeder, die vervolgens Furby* rustig uit de verpakking haalt en er batterijen in stopt. Zodra ze de batterijklep sluit, opent Furby* zijn ogen en spreekt hij zijn eerste onverstaanbare Furby*-woordjes uit. De tweeling wordt helemaal gek en begint tegen Furby* te schreeuwen in de hoop dat er meer feedback van het speelgoed komt. Furby* weet echter niet wat hij er mee aan moet en na nog een paar keer zijn Furby*-woordjes te herhalen begint ook hij een hard, huilerig geluid te maken. De tweeling schrikt hiervan en probeert hem te kalmeren door hem zachtjes over de rug te aaien. Dit helpt en Furby* is snel weer rustig.

'Hallo, ik ben Kees.' zegt Kees.
'Hallo! Furby*.' zegt Furby*.
'Hallo ik ben Karel!' zegt Karel.
'Hallo.' zegt Furby*.
'Hoe gaat het?' vraagt Kees.
'Honger.' zegt Furby*. Hierna houdt hij zijn mond geopend alsof hij verwacht dat er elk moment eten in gestopt wordt.
'Mama, Furby* heeft honger!!!' schreeuwt Karel op zijn hardst door de kamer terwijl hij daarna meteen naar haar toerent om eten te halen.
Furby* reageert op dit geschreeuw door zijn mond weer te sluiten, zijn ogen dicht te knijpen en zijn oren te bedekken. Als het geschreeuw afgelopen is doet hij zijn ogen weer open en maakt hij zijn oren weer vrij, er blijft echter wel een verdrietige blik hangen op zijn gezicht. Kees ziet dit en aait hem over zijn rug, na een tijdje wordt de uitdrukkig op Furby* zijn gezicht al iets vrolijker.

Niet veel later komen Karel en hun moeder terug met het speciale eten voor Furby* en Kees en Karel willen beiden de chips-smaak. 'Ik eerst!' zegt Karel. Furby*'s mond is weer dicht, maar zodra Karel dichterbij komt met het eten zegt Furby* 'Chips, lekker!' en opent hij zijn mond. Als Kees hierna hetzelfde doet opent Furby* weer als vanzelf zijn mond. Wanneer Karel daarna echter nog een keer Furby* wil voederen krijgt hij een negatieve reactie: Furby* zegt 'Vol!' en wendt zijn hoofd af van de 'Chips'.

De uren daarna vermaakten Kees en Karel zich met Furby*, ze hebben hem al een paar nieuwe woordjes geleerd en ook een spelletje met hem gespeeld. Na een tijdje begon Furby* echter wel trager te reageren en was hij ondertussen een aantal keer aan het gapen. Op een gegeven moment

zei hij ‘Furby* moe.’ en Kees en Karel besloten hem maar een plaatsje te geven in hun donkere slaapkamer, waar hij meteen in de slaapmodus terecht kwam.

Scenario 2

Een week later hebben de broertjes al een hoop dingen geleerd aan Furby. Ze willen hem daarom allebei graag nog meer leren en zijn nieuwste spelfuncties uitproberen, maar op een gegeven moment krijgen ze ruzie om wie met Furby* mag spelen.*

Als Kees beneden komt ziet hij Karel al met Furby* spelen, hij is de nieuwste spelfunctie aan het uitproberen.

‘Nu wil ik!! Jij was gisteravond ook al met Furby* aan het spelen!!’ schreeuwt Kees. Furby* weet even niet wat er aan de hand is en zegt ‘Hallo!’ tegen Kees die zojuist de kamer ingelopen is. Als Karel hem vertelt dat ze gewoon verder kunnen spelen gaat hij verder met het spel. Kees wordt boos dat hij gewoon genegeerd wordt en probeert Furby* te pakken, Karel was hem echter te snel af: hij houdt Furby* stevig in zijn armen, alsof ze aan het knuffelen zijn. Furby* is verbaasd. ‘Dit hoor niet bij het spel!’ weet hij Karel te vertellen. Kees rukt Furby* uit de handen van Karel en pakt hem nog steviger in zijn armen. ‘Wat gebeurt er!!!!?’ schreeuwt Furby* uit. Karel begint boos te schreeuwen en als hij merkt dat hij Furby* niet meer uit de handen van Kees kan trekken loopt hij naar hun moeder in de andere kamer.

Ondertussen pakt Kees Furby* nog steviger vast. Furby* heeft allang door dat ze met het spel gestopt zijn: ‘Dit is niet fijn! Laat me los!’ zegt Furby* op een luide toon. Kees verzwakt zijn grip iets, maar wanneer zijn moeder en Kees de kamer weer inlopen pakt hij Furby* weer erg stevig vast. ‘Au!’zegt Furby*. Kees negeert dit terwijl zijn moeder hem vertelt Kees niet zomaar het speelgoed van Karel mag pakken. Kees schreeuwt ‘Nee!’ en rent weg. ‘Wat is er aan de hand? Ik zie niks! Waar gaan we heen?’ schreeuwt Furby*. Als Kees in zijn slaapkamer is aangekomen en Furby* weer los laat om weer samen te gaan spelen weigert Furby* dit, hij is verdrietig en heeft nu geen zin om te spelen. Karel en hun moeder komen niet veel later ook de kamer in lopen. ‘Jullie kunnen ook samen spelen’ stelt hun moeder voor. ‘Furby* wil nu toch niet meer spelen.’ zegt Kees. Karel loopt naar Furby* toe en begint hem te aaien. ‘Kijk, zo wordt hij wat vrolijker en wil hij zo weer spelen!’ vertelt hij Kees. ‘Probeer maar!’ Kees aait over de rug van Furby*. ‘Hmm…. Dat is f—hahahaha!’ zegt Furby* terwijl Karel hem in zijn buik begint te kietelen. Kees, Karel en Furby* worden allemaal wat vrolijker en wanneer ze allemaal opgevrolijkt zijn gaan ze met zijn drieën een spel doen.

Bijlage B – Dimensies van het welzijn

Om 'the next Furby' wel een leuk speelgoed te laten blijven zal deze zo weinig mogelijk negatieve emoties moeten bevatten. Daardoor zullen voornamelijk de positieve eigenschappen van deze dimensies voorkomen in het gedrag van Furby*. Hieronder kort beschreven wat met de term bedoelt wordt en wat voor gedrag er bij verwacht kan worden.

Zelfacceptatie
Positief: Blij met het leven en blij met zijn eigen positieve en negatieve eigenschappen, ook blij met zijn verleden. Dit resulteert in het uiten van gevoelens en emoties zonder enige moeite of schaamte.
Negatief: Niet gelukkig met zichzelf of met het verleden, wil graag anders zijn. Dit resulteert in onzeker gedrag waardoor veel vragen over zichzelf gesteld worden.

Persoonlijke groei
Positief: Goede relaties met anderen, maakt zich zorgen om anderen en snapt het geven en nemen in relaties. Ook is het niet moeilijk om over emoties te praten. Dit resulteert in vragen stellen over de ander en praten over emoties.
Negatief: Weinig hechte relaties met anderen, maakt zich ook geen zorgen over anderen en is geïsoleerd, praat niet over emoties e.d. Dit resulteert in een te sterke eigen wil, omdat hij niet in staat is om overeenkomsten te maken.

Doel in het leven
Positief: Zelfstandig, kan sociale druk onderdrukken, geeft zijn eigen mening. Dit resulteert in het uitvoeren van zijn eigen wensen i.p.v. het uitvoeren van wensen van de gebruiker.
Negatief: Is afhankelijk van anderen, kan geen eigen beslissingen nemen. Dit resulteert in een hoop vragen over acties die hij moet uitvoeren.

Positieve relaties met anderen
Positief: Weet de omgeving naar eigen wensen en waardes aan te passen. Dit resulteert in overeenkomsten tussen de Furby en de gebruiker waarin ze allebei tevreden zijn.
Negatief: Weet de omgeving niet naar eigen wensen en waardes aan te passen. Dit resulteert in extreem ontevreden gedrag (door middel van schreeuwen of huilen).

Omgevingsgericht
Positief: Heeft doelen in het leven en is tevreden over het verleden. Dit resulteert in positief energiek gedrag.
Negatief: Ziet het doel van het leven niet in. Dit resulteert in negatief gedrag, er zijn geen doelen die hij wil bereiken.

Autonomie
Positief: Heeft het idee dat hij continu aan het 'groeien' is, open voor nieuwe ervaringen. Dit resulteert in positief sociaal gedrag.
Negatief: Verveeld en ongeïnteresseerd in het leven. Dit resulteert in constante negatief gedrag.

De 6 dimensies van welzijn staan hieronder per fase kort beschreven. De autonomie is altijd goed, omdat het gedrag van de Furby anders zo gedesinteresseerd zou zijn dat het niet meer leuk is om

ermee te moeten spelen. Verder wordt het gedrag van een dimensie die 'laag' is in een bepaalde fase nooit zo extreem uitgevoerd dat het negatieve gedrag constant aanwezig is.

	Fase 1	**Fase 2**	**Fase 3**	**Fase 4**
Zelf-acceptatie	Optimaal. Hij bezit geen zelfbewustzijn, dus zal ook niet doorhebben dat hij iets verkeerd doet.	Minder. Hij is onzeker en stelt bijvoorbeeld vragen of het niet vervelend is wat hij aan het doen is en of mensen niet boos zijn.	Goed. Hij uit zijn emoties door te zeggen dat hij boos is of verdrietig i.p.v. dit te uiten in huilen/schreeuwen.	Hoog. Zijn emoties zijn minder belangrijk voor de buitenwereld, maar hij praat er wel gemakkelijk over als het nodig is.
Persoon-lijke groei	Laag. Hij heeft niet echt door wat emoties precies zijn en zijn wensen zijn voor hem het allerbelangrijkst.	Redelijk laag. Hij stelt zijn eigen emoties en gevoelens voorop, daarna komen die van anderen pas.	Goed. Hij maakt zich eerst druk om de gevoelens van anderen en daarna pas om zichzelf.	Optimaal.
Doel in het leven	Redelijk. Hij weet niet wat hij wil bereiken, alleen dat hij af en toe eten wil e.d. (dit geeft hij ook aan), hier is hij echter wel al snel tevreden mee.	Laag. Hij vraagt bij ongeveer alles aan de gebruiker of dat wel de goede actie is om uit te voeren en gelooft alles wat de gebruiker zegt.	Gemiddeld. Hij heeft niet echt meer eisen die hij wil bereiken, maar als de gebruiker hem vertelt wat hij moet doen heeft hij hier wel altijd zijn eigen mening over.	Gemiddeld. Hij accepteert het vaak wel wat de gebruiker vertelt dat hij moet doen, maar noemt nog wel zijn eigen mening hierover, hij weigert weinig.
Positieve relaties met anderen	Laag. Hij weet niet wat hij met de omgeving aan moet, dit resulteert vaak in een expressieve uiting van zijn gevoelens.	Gemiddeld. Zijn eigen emoties en gevoelens zijn nog te belangrijk voor hem, maar ook die van anderen vindt hij vaak toch wel belangrijk.	Goed. Emoties en gevoelens van anderen gaan voor, zijn eigen gevoelens worden alleen genoemd als dat relevant is voor de situatie.	Goed. Emoties en gevoelens van anderen gaan voor, zijn eigen gevoelens worden alleen genoemd als dat relevant is voor de situatie.
Omgevings gericht	Laag. Hij weet niet wat hij allemaal wil bereiken.	Redelijk. Hij ziet niet altijd in waarom hij dingen moet doen, maar weet wel dat hij graag dingen wil bereiken.	Goed. Hij kijkt terug op dingen die hij geleerd heeft en ziet eigenlijk alleen maar verbetering; hij wil nog meer leren.	Redelijk. Hij kijkt positief terug op het verleden, maar weet dat er eigenlijk niks meer te leren valt.
Autonomie	Goed.	Goed.	Goed.	Goed.

Bijlage C – Programma van Eisen Furby*

Geloofwaardigheid van het 'leven' van Furby*

De eerste keer dat Furby* wordt 'aangezet' moet hij zich gedragen alsof hij daadwerkelijk voor het eerst op de wereld komt.

De feedback die vanuit Furby* komt moet uitgebreid genoeg zijn om ervoor te zorgen dat Furby* overkomt als een levend wezen.

Furby* moet bij kunnen houden wanneer hij genoeg gegeten heeft en wanneer hij weer honger heeft.

Furby* moet ook kunnen reageren als hij een commando niet helemaal begrijpt.

Het moet mogelijk zijn voor Furby* om te kunnen reageren op geluiden uit de omgeving om ervoor te zorgen dat Furby* overkomt als een levend wezen.

Entertainmentwaarde

De entertainmentwaarde van Furby* moet hoog zijn, zowel voor jonge kinderen als voor wat oudere kinderen.

- Er moet een mogelijkheid zijn dat Furby* op de gebruiker reageert zoals de gebruiker dat wil.
- Furby* moet genoeg spelsuggesties bezitten om de gebruiker voor een bepaalde tijd niet te vervelen.
- Furby* moet niet voorspelbaar reageren op de sensoren.

Het moet mogelijk zijn om een uitgebreide audio output te geven aan Furby*.

Het moet mogelijk zijn om Furby* veel verschillende reacties te kunnen laten geven.

Het moet mogelijk zijn om Furby* veel verschillende spelletjes te laten bezitten.

Furby* moet te begrijpen en te gebruiken zijn zonder de handleiding.

- Speciale acties moeten niet vast zitten aan een strenge opvolging van acties

Kinderen moeten spelenderwijs nieuwe mogelijkheden van Furby* kunnen leren en gebruiken.

(Gemoeds)toestand

Furby* moet verschillende (gemoeds)toestanden hebben.

Bij elke (gemoeds)toestand moet het mogelijk zijn om Furby* zich op een andere manier te laten gedragen.

- Furby* moet op dezelfde (inter)actie anders reageren bij verschillende (gemoeds)toestanden.
- Furby* moet minder snel praten wanneer hij minder energie heeft.
- Furby* moet minder snel dingen begrijpen wanneer hij minder energie heeft .
- Er moet een gaap-actie zijn die af en toe aangeroepen wordt wanneer Furby* moe is.

De verschillende (gemoeds)toestanden hebben óf wel óf geen toegang tot de spelletjes.

Er moet bijgehouden kunnen worden:

- Hoelang een bepaald geluid aanhoudt;
- Welke reacties beschikbaar zijn voor Furby* om te geven;
- Hoe hongerig Furby* is;
- Hoeveel energie Furby* heeft

Er moet een tijdelijke opslagruimte zijn (voor de voortgang van een spel).

Furby* moet kunnen waarnemen hoeveel gebruikers een spel met hem spelen.

Furby* moet makkelijk 'in slaap kunnen vallen', maar alleen als de gebruiker hier de opdracht toe geeft.

De (gemoeds)toestand van Furby* moet kunnen veranderen bij nieuwe (inter)acties.

(Re)acties Furby

Het moet mogelijk zijn om acties, die Furby* op het moment uit aan het voeren is, stop te zetten.

Furby* moet weten wanneer het de intentie van de gebruiker is om een (inter)actie te beginnen en ook alleen dan reageren.

Furby* moet op verschillende manieren kunnen reageren bij sterke en zwakke (inter)acties vanuit de gebruiker.

Furby* moet, afhankelijk van de situatie waarin hij zich bevindt, verschillend kunnen reageren op (dezelfde sterkte van) een (inter)actie vanuit de gebruiker.

Er moeten steeds meer acties van Furby* 'vrij' komen, terwijl andere acties juist weer weggestopt moeten worden.

De verschillende acties moeten gerangschikt zijn naar belangrijkheid.

Er moeten meerdere acties tegelijk plaats kunnen vinden.

Een actie moet gestopt kunnen worden wanneer een andere actie die 'belangrijker' is uitgevoerd moet worden.

Furby* moet met een actuator kunnen reageren terwijl een andere actuator door blijft gaan met wat deze aan het doen was.

Bijlage D – Programma van Eisen slim product

Algemeen slim product

Het product moet de mogelijkheid bezitten de waarde van een bepaalde variabele te kunnen veranderen, op te kunnen vragen wanneer dit nodig is en een signaal binnen te krijgen wanneer deze waarde te hoog of juist te laag is.

Het product moet altijd reageren op de gebruiker, ook wanneer het commando dat binnen komt onduidelijk is.

Het product moet te begrijpen en te gebruiken zijn zonder handleiding.

Speciale acties moeten niet vastzitten aan een strenge opvolging van acties, deze moeten net zo gemakkelijk te starten zijn als simpele acties.

Het product moet verschillende (gemoeds)toestanden bezitten.

Bij elke (gemoeds)toestand moet het product zich op een andere manier gedragen.

Het product moet zich afhankelijk van interne waardes op een bepaalde manier kunnen gedragen.

Het product moet een tijdelijke opslagruimte bezitten.

Het product moet gemakkelijk uitgeschakeld kunnen worden/in slaapstand terecht kunnen komen, maar alleen wanneer de gebruiker hier de opdracht toe geeft.

Het veranderen van interne waardes moet kunnen leiden tot verandering van (gemoeds)toestand

Acties die het product momenteel aan het uitvoeren is, moeten stopgezet kunnen worden.

Het product moet kunnen waarnemen wanneer het de intentie van de gebruiker is om een (inter)actie te beginnen en ook alleen dan reageren.

Het product moet een beeld hebben van de situatie waarin deze zich bevindt.

De verschillende acties moeten gerangschikt zijn naar belangrijkheid.

Er moeten meerdere acties tegelijk plaats kunnen vinden.

Een actie moet gestopt kunnen worden wanneer een andere actie die ‘belangrijker’ is uitgevoerd moet worden.

Een actuator moet door kunnen blijven gaan als het alleen nodig is voor een andere actuator om te reageren op bepaalde input.

Product met zijn eigen ´gevoelens´/’wil’/’hoop’/’wensen’

De feedback vanuit het product mag niet beperkt zijn.

Het product moet kunnen reageren op geluiden uit de omgeving.

Het product moet veel verschillende reacties bezitten (meerdere reactie bij één bepaald commando vanuit de gebruiker).

Nieuwe mogelijkheden of (gemoeds)toestanden van het product moeten ontdekt kunnen worden door het product te gebruiken/met het product te interacteren.

Het product moet niet voorspelbaar reageren op de ingebouwde sensoren.

Het product moet op dezelfde (inter)actie anders reageren in verschillende (gemoeds)toestanden.

Het product moet omgevingsvariabelen bij kunnen houden.

Het product moet bij kunnen houden welke reacties er beschikbaar zijn om te geven.

Het product moet kunnen waarnemen hoeveel gebruikers er met het product interacteren.

Nieuwe (inter)acties vanuit de gebruiker moeten kunnen leiden tot verandering van (gemoeds)toestand.

Het verloop van de tijd moet kunnen leiden tot verandering van (gemoeds)toestand.

Het product moet op verschillende manieren kunnen reageren bij sterke en zwakke (inter)acties vanuit de gebruiker.

Het product moet, afhankelijk van de situatie waarin hij zich bevindt, verschillend kunnen reageren op (dezelfde sterkte van) een (inter)actie vanuit de gebruiker.

Er moeten steeds meer acties van het product ‘vrij’ kunnen komen, terwijl andere acties juist weer weggestopt moeten kunnen worden.

Bijlage E - Internal states-woorden

Perceptual
→ SENSOREN
See
Look
Watch
Hear
Listen
Taste
Smell
Feel(soft, warm)
Cold(feeling cold)
Freezing
Hot
Warm
Hurt

Physiological
→ TOESTAND
Hungry
Starving
Thirsty
Sleepy
Sleep
Asleep
Tired
Awake
Wake up
Sick
Full
Make better

Emotional and affective
→ MINDS
<u>Positive:</u>
Happy
Have fun
Funny
Proud
Feel (good, bad)
To be all right
Better
Good (feeling)
O.K.
Nice
Like
Love
Have a good time
Surprised
<u>Negative:</u>
Sad
Angry
Mad
Scared
Scary

Dirty
Messy
Yucky

Bad(feeling)
Hurt feeling
Afraid
<u>Affect expression:</u>
Hug
Kiss
Laugh
Smile
Cry

Volition and ability
→ WENSENLIJST
Want
Need
Have to
Can
Hard

Cognition
→ GEHEUGEN
Know
Think
Remember
Forget
Maybe
May
Understand
Pretend
Dream
Real

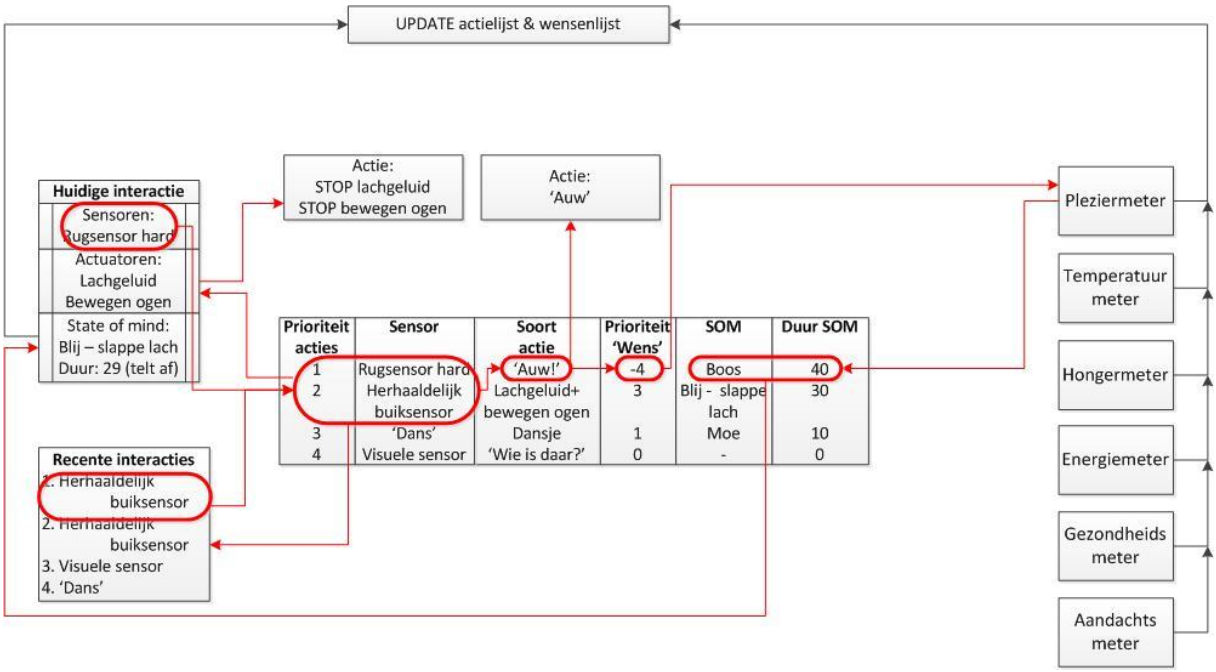
Guess
Mean (I mean)

Moral judgment and obligation
Good
Bad
Naughty
May
Let
Supposed to
Must
Have to
Should
Can (for permission)

Bijlage F – Brainstormfase structuren

Toen het algemene gedrag en de werking van Furby* beschreven waren, was het tijd om de werking van het algemene gedrag in kaart te brengen. Dit ging niet vlekkeloos, er zijn een aantal structuren gemaakt voordat de uiteindelijke structuur gekozen was.

Allereerst was er een grote structuur die alle aspecten van de werking van Furby* meenam, met daarin ook een voorbeeld verwerkt:



In deze structuur waren de 'minds' (hier: meters) en '(gemoeds)toestanden'(hier: State of Mind/SOM) nog niet zo sterk met elkaar verbonden. Er waren een aantal meters die voornamelijk aangepast werden door interne factoren. (Verandering in de tijd, tenzij de gebruiker een tegenreactie geeft.) Input van een gebruiker kon rechtstreeks leiden tot een nieuwe (gemoeds)toestand/SOM, mits de pleziermeter dit toeliet. (Deze heeft een positieve en negatieve kant, wat kan leiden tot een positieve dan wel negatieve SOM).

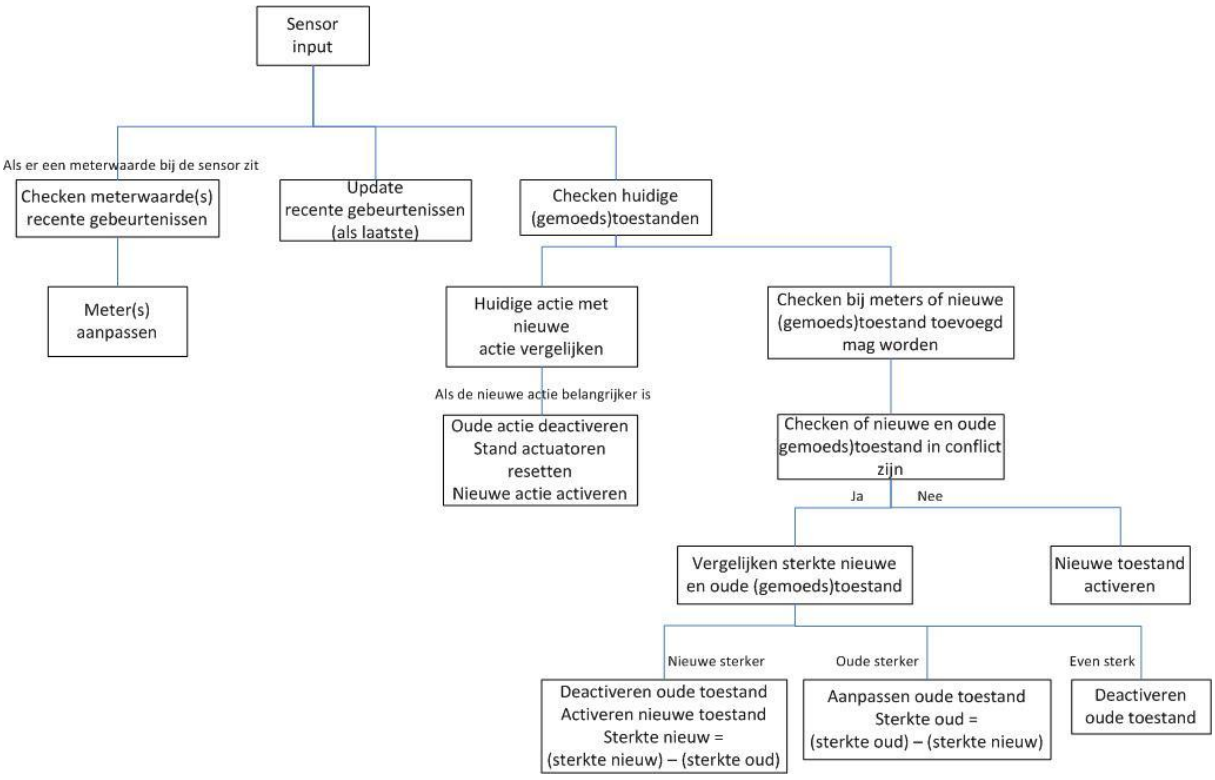
Elke SOM had een bepaalde duur, die aangepast werd wanneer een conflicterende SOM werd aangeroepen.

Uitleg diagram:

'In dit diagram wordt de Furby hard op zijn rug geslagen terwijl hij aan het lachen is. Zoals in het diagram te zien is wordt er gecontroleerd of de actie van de sensor die op het huidige moment wordt aangeroepen belangrijker is dan de sensor die daarvoor aangeroepen was. In dit geval is dit zo. (De prioriteit van de actie die bij 'rugsensor hard' hoort is 1 en de prioriteit van de huidige interactie 2.)

De volgende stap is dan om te controleren welke actuatoren op het moment in werking zijn en deze stopzetten. De nieuw aangeroepen sensor wordt op de eerste plaats bij recente interacties geplaatst en de actie die bij de sensor hoort wordt uitgevoerd. Vervolgens wordt de prioriteit van de (zojuist uitgekomen) wens aan de pleziermeter toegevoegd, die vervolgens controleert of de meter laag

genoeg is om de Furby in een boze state-of-mind te brengen. Samen met de uitkomst van deze pleziermeter wordt de eventueel nieuwe state-of-mind gecombineerd met de huidige state-of-mind. Om deze in duur (en sterkte?) af te laten nemen. Als deze nieuwe state-of-mind in de huidige interactie geplaatst is wordt de actie- en wensenlijst geupdate.'

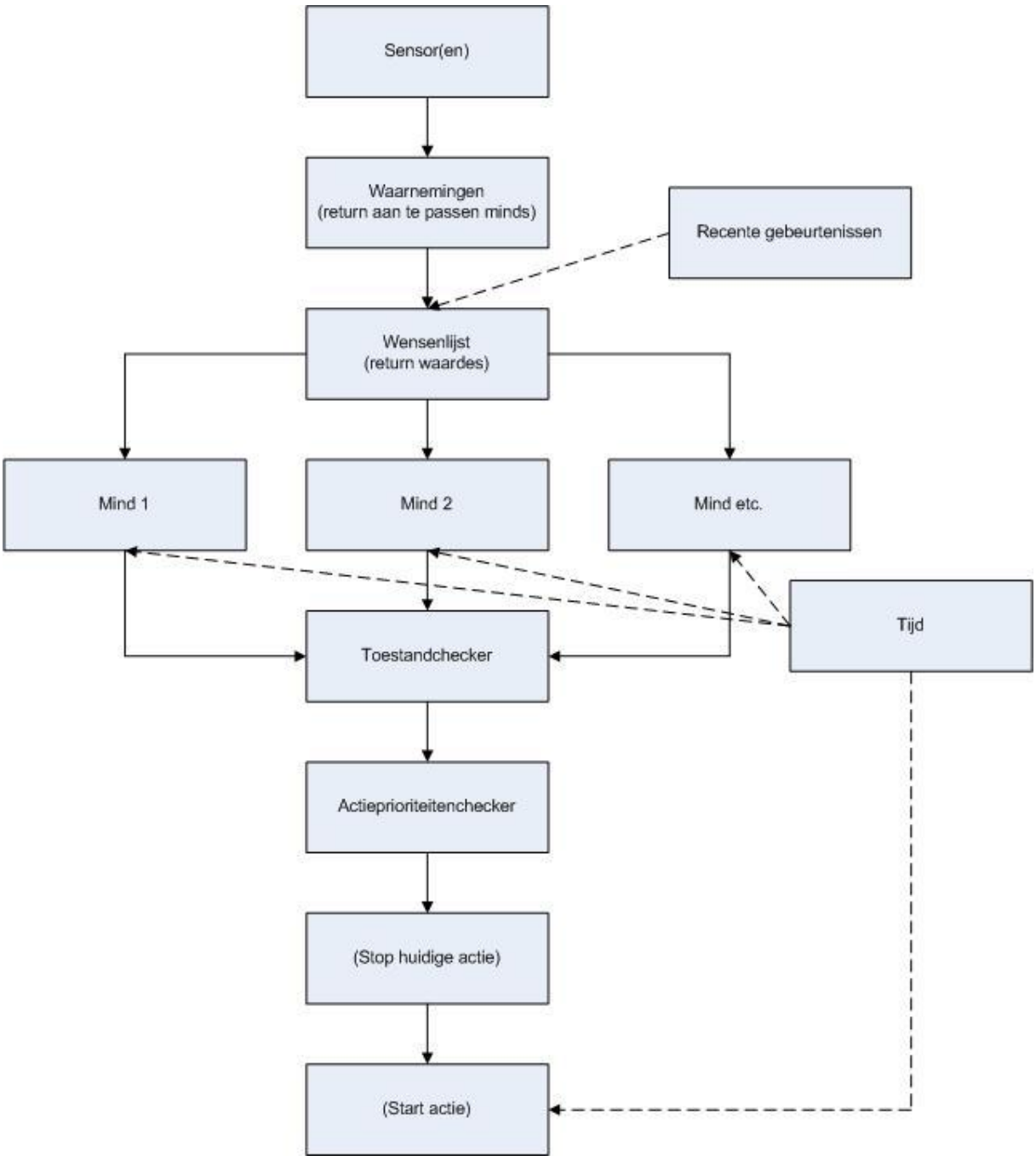


Deze structuur was daarna iets abstracter beschreven en uiteindelijk verwerkt tot een opzet voor het maken van een nieuw slim product en een actieboom (zie bovenstaand figuur). De opdrachtgever had hier echter een aantal opmerkingen over en het bleek nog niet de ultieme oplossing te zijn.

Het grootste probleem bleek te zijn dat er enkel meters zijn die de interne activiteit van het product meten. Er wordt niet gemeten hóe blij een product is, of hóe boos. Wanneer een product maar een klein beetje blij is van een aantal acties die daarvoor zijn uitgevoerd, maar er dan opeens een actie komt waar hij heel erg boos van zou kunnen worden, dan gebeurt dit bij deze structuur niet zo snel. De duur van zijn blije 'state of mind' kan namelijk erg zijn opgelopen.

De opdrachtgever wou graag een structuur zien waarin elke sterkte van verschillende 'states of mind' ook gemeten werd.

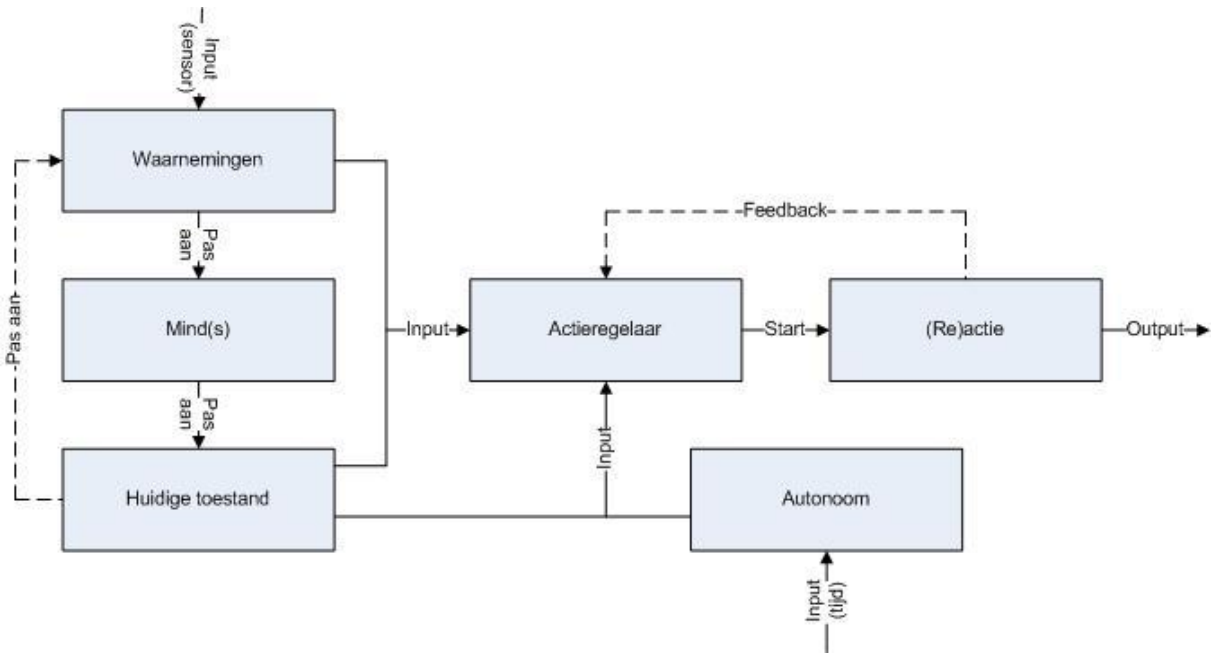
Hieruit kwam na veel gepuzzel de volgende actieboom:



Hierin worden de waarnemingen nog steeds meteen omgezet in de reacties die het product op de input zou moeten geven. Vervolgens worden verschillende minds aangepast aan de hand van de output van de waarnemingen en de wensenlijst(die up to date wordt gehouden door de huidige (gemoeds)toestand en de recente gebeurtenissen). De toestandchecker kijkt of de (gemoeds)toestand verandert en vervolgens wordt er gekeken welke reactie er gegeven moet worden.

Wanneer er geen input is van de gebruiker begint de tijd autonoom een loop. De minds worden aangepast, vervolgens kijkt de toestandchecker of zijn toestand veranderd moet worden en daarna wordt er een autonome actie gestart.

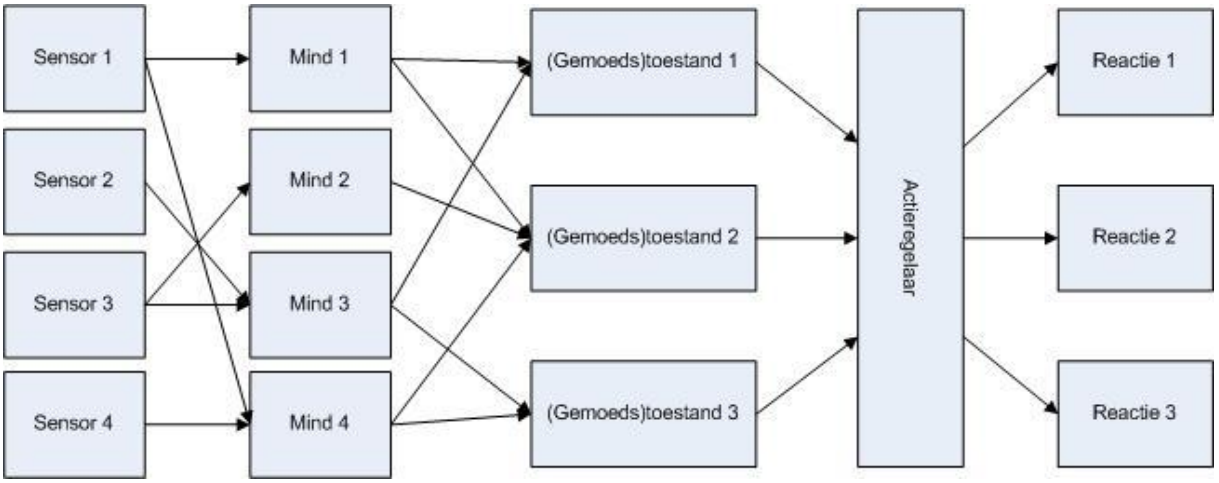
Deze was meteen omgezet in een structuur met de eventuele Java-klasses waarin de werking van een slim product gemakkelijk geprogrammeerd zou moeten kunnen worden:



De opdrachtgever was nog niet overtuigd. Hij stelde een structuur voor waarin elke mogelijke reactie zijn eigen '(gemoeds)toestand' had, waarvan de output ook hoog of laag kon zijn.

Minds houden meer bij dan alleen het humeur van een slim product. Ook de omgevingsfactoren worden bijgehouden. (Bijvoorbeeld: Hoe groot is een persoon? Dit kan resulteren in een kind of een volwassene.)

De reactie die gegeven wordt is afhankelijk van de (gemoeds)toestand met de grootste output:



Deze structuur was echter nog niet vlekkeloos en om deze structuur aan de gestelde eisen te laten voldoen, die opgesteld zijn in het hoofdstuk werking* van Furby zou er nog een hoop aan gesleuteld moeten worden. Na nog een tijd gebrainstormd te hebben zijn er een hoop aspecten gedefinieerd (hoofdstuk 2) die gebruikt zullen worden in de uiteindelijke structuur. Deze uiteindelijke structuur is een combinatie van de laatste 2 genoemde structuren. Hoe deze structuur er precies uit gaat zienstaat beschreven in hoofdstuk 3.

Bijlage G – Uitleg ontwerppatronen software

Abstract factory

Gebruikt de abstracte classe/interface als parameter en roept hier vervolgens methodes op aan.

Adapter

Een soort hoofdinterface wat gebruikt wordt om verschillende klassen toch op dezelfde manier te laten gebruiken. Gebruikt wrappers om verschillende klasses met elkaar te laten communiceren

Bridge

Een klasse die een abstracte klasse implementeert zijn eigen geërfde methode laten implementeren, zodat deze toch anders gebruikt wordt.

Builder

Maakt en gedeelte van een groot gehaal en kan er daarom voor zorgen dat de juiste objecten gebouwd worden.

Chain of Responsibility

Er wordt een verzoek doorgegeven aan verschillende klassen, totdat er een klasse is die het verzoek uitvoert door een actie uit te voeren.

Command

Bewaart alle informatie die nodig is om een methode aan te roepen om de methode later pas aan te roepen.

Composite

De complete verzameling van objecten zal hetzelfde behandeld worden als de objecten zelf.

Decorator

Geeft extra verantwoordelijkheden aan een object tijdens run-time.

Façade

Een interface voor een aantal klassen die met elkaar verbonden zijn, om het aantal verbindingen tussen deze klassen te verminderen.

Factory Method

Heeft de mogelijkheid nieuwe klassen aan te maken, en een bepaalde duur aan deze klasses te geven.

Flyweight

Een object dat op meerdere plekken gebruikt kan worden om de run-time belasting te verminderen.

Interpreter

Een patroon om tekst te controleren op grammatica.

Iterator

Een patroon om door een lijst met objecten te gaan en er een handeling op uit te voeren.

Mediator

Representeert de verbindingen tussen andere klassen en regelt de interacties tussen deze klassen, zodat deze klassen alleen met de mediator verbonden hoeven te worden en niet met elkaar, wat alles een stuk minder complex maakt.

Memento

Een patroon dat de mogelijkheid geeft een object naar zijn vorige staat te herstellen.

Observer

Wanneer er een bepaalde handeling uitgevoerd wordt binnen een observable-klasse worden zijn observers hiervan op de hoogte gesteld. Deze observers kunnen dan een update uitvoeren, eventueel met behulp van het meegegeven argument.

Prototype

Maakt een nieuw object door een bestaand object te kopieren.

Proxy

Een proxy is een klasse die de waardes van een andere klasse bijhoudt, maar deze pas doorgeeft wanneer er daadwerkelijk acties met deze waarde uitgevoerd moeten worden.

Singleton

Zorgt ervoor dat er maar één instantie kan bestaan van een bepaalde klasse.

State

Zorgt ervoor dat het gedrag van een object kan veranderen wanneer zijn status verandert.

Strategy

Een aantal klasses met dezelfde methode, maar verschillende algoritmes in deze methode.

Template Method

Een klasse met een aantal lege methodes, die door subklassen worden geïmplementeerd.

Visitor

Een klasse met alleen maar lege methodes, die door subklassen worden geïmplementeerd.

Bijlage H – implementaties structuren

Elke structuur is onderverdeeld in verschillende 'packages'. Dit is gedaan om de structuur overzichtelijk te houden en de klassen die bij elkaar horen bij elkaar te houden. Packages die bij elke structuur hetzelfde is zijn:

Actuatoren

Er wordt constant gecheckt of er een nieuwe output is. Wanneer deze er is wordt de oude output gestopt en wordt de nieuwe output meteen uitgevoerd op de actuator. (Tegelijkertijd wordt de nieuwe waarde weer op default gezet.)
(In de implementatie zijn de actuatoren verbonden met de ActuatorenGUI en passen hierop de output aan.)

Gemoedstoestanden

Alle verschillende soorten input hebben een (wenswaarde en een) actiewaarde gekregen. (Dit verschilt per gemoedstoestand.) Deze worden als lijst opgeslagen en kunnen opgevraagd worden wanneer dit nodig is.

Minds

Minds heeft een abstracte klasse, deze bevat de methodes setWaarde() en getWaarde(), die de waarde van de mind kunnen aanpassen en opvragen. De waarde zelf kan alleen tussen 0 en 1 zitten.

Product

Dit is gemaakt om een product te visualiseren.
ActuatorenGUI – Laat de output van de verschillende actuatoren zien.
SensorenGUI – Geeft de mogelijkheid input te geven aan elke sensor.

Reacties

Bevat een interface met een String per soort reactie, om fout gespelde woorden e.d. te voorkomen.

Sensoren

Deze zijn listeners van de sensorengUI.
De sensoren krijgen de inputmediator ook als instantievariabele om het door te geven wanneer er nieuwe input is.

Mediatorstructuur

Regelaars

Bevat de inputmediator:

Deze bevat de main en wordt gekoppeld aan de sensoren, de minds, de gemoedstoestand en de outputmediator.

Vervolgens wordt de SensorenGUI gecreëerd (de sensoren worden in werking gezet) en wordt de tijd gestart.

De inputmediator zorgt dat de waardes van verschillende minds op het juiste moment worden aangepast en stuurt daarna een reactie naar de outputmediator. (Bij input van sensoren of van de tijd)

Bevat de outputmediator:

De outputmediator wordt gekoppeld aan dezelfde minds en gemoedstoestand als de inputmediator, aan de inputmediator zelf en aan de actuatoren. Ook wordt er een ActuatorenGUI gecreëerd.

De outputmediator krijgt een reactie binnen van de inputmediator en stuurt vervolgens de actuatoren aan met de juiste waardes (die uit de minds worden gelezen).

Bevat de tijd:

Deze stuurt om de zoveel seconden een bericht naar de inputmediator.

Observerstructuur

Actuatoren

Observer van reactie. Als er een reactie gegeven moet worden komt er een update binnen bij elke relevante actuator. (Naast de algemene implementatie)

Hoofdstructuur

Bevat GT:

Deze houdt bij welke gemoedstoestand er actief is. Deze krijgt een update binnen van mind wanneer deze aangepast worden om te kijken of de huidige gemoedstoestand veranderd moet worden.

Bevat Mind:

Deze houdt bij welke stand elke mind heeft en past deze aan wanneer er een update binnen komt van Waarneming of Tijd

Bevat MindChange:

Een object dat doorgegeven kan worden om aan te geven welke mind met welke waarde aangepast moet worden.

Bevat Output:

Bevat een methode om alle waardes van de verschillende actuatoren aan te passen. Deze klasse kan meegegeven worden via een notifyobservers bericht vanuit reactie naar de actuatoren.

Bevat Reactie:

Bevat de main en wordt gestart. Alle nodige klassen worden aangemaakt en krijgen de juiste observers.

Reactie kan een update krijgen van Mind of Waarneming of Tijd. Wanneer er een update wordt gekregen van Mind wordt de waarde die Reactie bijhoudt van de Mind geupdate, wanneer er een update binnen komt van Waarneming of Tijd wordt de juiste output(van de reactie) doorgegeven aan de observers van reactie.

Bevat Reacties:

Bevat een interface met een String per soort reactie (in plaats van in de Reacties-package).

Bevat Tijd:

Deze stuurt om de zoveel seconden een bericht naar mind en reactie.

Bevat Waarneming:

Er komt een update binnen van de sensoren, dit wordt omgezet in een string van een reactie die vervolgens via de observer-structuur weer verder gestuurd wordt.

Instantievariabelenstructuur

Reacties

Wanneer nodig heeft elke reactie een eigen klasse met de output van zijn actuatoren.

Regelaars

Bevat ReactieArbitrator:

Deze kijkt of de actie gestart moet worden en zo ja, start hij de actie en stopt hij eventueel de oude actie.

Bevat Tijd:

Deze stuurt om de zoveel seconden een bericht naar reactiearbitrator.

Bevat Waarnemingen:

Bevat de main-methode. Maakt alle minds, reacties, sensoren, de sensorenGUI en de reactiearbitrator aan.

Wanneer er input komt van een sensor worden alle waardes van één bepaalde reactie doorgestuurd naar de reactiearbitrator.

Proxystructuur

Reacties

De reactie-klassen in deze packages zijn allemaal een uitbreiding van de abstracte klasse reactieproxy, die een proxy-object kan bezitten, de mogelijke output van alle actuatoren en een execute()-methode, die de reactie daadwerkelijk uitvoert.

Regelaars

Bevat Actieregelaar:

Als er een reactie binnen komt wordt er gekeken of deze gestart mag/moet worden en zo ja, dan wordt de execute()-methode van deze reactie uitgevoerd.

Bevat Proxy:

Bevat de waarde van de grootte van de letters van de tekst (actuator).

Bevat Waarnemingen:

Stuurt een reactie met proxy-instantie door naar actieregelaar bij input van de sensoren.

	Mediator	Instantie-variabelen	Proxy	Observer	Mediator Papegaai	Instantie-variabelen Papegaai	Proxy Papegaai	Observer Papegaai
Methods per class	8	9	9	6	7	7	7	5
Cyclomatic Complexity	12	6	6	12	8	8	8	16
Feature Envy	6	6	6	6	5	5	5	5
Lines of Code in Method	5	5	5	5	6	5	5	5
Number of Locals in Scope	2	2	2	2	2	2	2	2
Number of Levels	8	4	4	8	5	5	5	8
Number of Parameters	7	5	3	4	7	6	6	7
Number of Statements	92	44	40	78	37	37	37	43
Efferent Couplings	19	32	32	20	21	23	29	25
LCOM-CK	1	1	1	6	12	12	12	10
LCOM-HS%	100	88	100	100	83	83	83	100
LCOM-PFI%	67	78	77	100	73	83	83	100
LCOM-TC%	760	1643	1768	576	717	1127	1467	600
NOF	11	26	28	9	12	12	23	9
WMC	21	26	26	23	23	23	23	36
TLOC	19	11	11	7	16	8	8	13
Classes	18	30	31	22	19	19	26	24
Aantal zwakste punten	6	4	7	5	5	2	6	7

Hierbij is:

Cyclomatic Complexity
McCabe's Cyclomatic Complexity. Het aantal individuele wegen dat door de logica van een programma kan lopen.

Feature Envy
Als een klasse meer geïnteresseerd is in de methodes van een andere klasse dan die van zichzelf.

Number of Locals in Scope
Het aantal lokale variabelen in een methode.

Number of Levels
Het aantal verschillende uitkomsten in een methode.

Number of Parameters
Het aantal parameters dat meegegeven wordt bij een methode.

Number of Statements
Het aantal if/for/while statements, methode-aanroepen, constructor-aanroepen.

Efferent Couplings
Naar hoeveel types(erving, interface, parameter, variabelen, excepties) er verwezen worden in de klassen.

LCOM (CK)
Lack of Cohesion in Methods (Chidamber and Kemerer). Methodes in een klasse die geen instantievariabelen met elkaar gemeen hebben – methods in een klasse die minstens 1 instantievariabele met elkaar gemeen heeft.

LCOM (HS)
Lack of Cohesion in Methods (Henderson-Sellers). Een berekening die de relaties tussen het aantal instantievariabelen die gebruikt worden in een aantal methodes met elkaar te vergelijken.

LCOM (PFI)
Lack of Cohesion in Methods (Pairwise Field Irrelation). Een berekening die de relaties tussen het aantal instantievariabelen die gebruikt worden in een aantal methodes met elkaar te vergelijken.

LCOM (TC)
Lack of Cohesion in Methods (Total Correlation). Een berekening die de relaties tussen het aantal instantievariabelen die gebruikt worden in een aantal methodes met elkaar te vergelijken.

NOF
Number of Fields. Het aantal instantievariabelen.

WMC
Weighted Methods per Class. De som van de complexiteit in methodes(Cyclomatic Complexity).

TLOC
Thousand Lines of Code. Het total aantal regels code in het hele programma.

	Mediator	Instantie	Proxy	Observer
Kan de waarde van een mind veranderd en opgevraagd worden wanneer dit nodig is?	5 Alle mediators zijn verbonden.	2 De reactiearbitrator heeft geen enkele link met de minds.	3 De uitkomsten van de minds worden wel constant bijgehouden	4 Elke klasse houdt de waardes bij, maar er is maar 1 echte verbinding.
Heeft het product 'het door' wanneer een signaal hiervan te hoog of te laag is?	5	3	3	3
Zijn speciale acties net zo gemakkelijk te starten als simpele acties?	- Niet uitgewerkt.	- Niet uitgewerkt.	- Niet uitgewerkt.	- Niet uitgewerkt.
Bezit het product verschillende gemoedstoestanden ?	5	5	5	5
Is het gedrag dat het product vertoont afhankelijk van zijn interne waardes?	5	4	3	4
Bezit het product een tijdelijke opslagruimte?	5	5	5	5
Kan het veranderen van interne waardes leiden tot verandering van (gemoeds)toestand?	5	5	5	5
Kunnen acties die het product aan het uitvoeren is stopgezet worden als dit nodig is?	5	5	5	5
Kan het product onderscheid maken tussen een sensor die per ongeluk wordt aangeroepen en een sensor die expres wordt aangeroepen?	- Niet uitgewerkt.	- Niet uitgewerkt.	- Niet uitgewerkt.	- Niet uitgewerkt.
Heeft het product een beeld van de situatie waarin deze zich bevindt?	3 Huidige actie.	3 Huidige actie.	3 Huidige actie.	3 Huidige actie.
Heeft het product	5	5	5	5

een actieprioriteitenlijst waar deze gebruik van maakt?				
Kunnen er meerdere acties tegelijkertijd plaatsvinden?	4 Constante beweging, andere tekst.	4 Constante beweging, andere tekst.	4 Constante beweging, andere tekst.	4 Constante beweging, andere tekst.
Reageert het product op dezelfde (inter)actie anders in verschillende (gemoeds)toestanden?	2 Wordt alleen gecheckt of de actie uitgevoerd mag worden.	2 Wordt alleen gecheckt of de actie uitgevoerd mag worden.	2 Wordt alleen gecheckt of de actie uitgevoerd mag worden.	2 Wordt alleen gecheckt of de actie uitgevoerd mag worden.
Reageert het product anders op sterke en zwakke (inter)acties vanuit de gebruiker?	- Niet uitgewerkt.	- Niet uitgewerkt.	- Niet uitgewerkt.	- Niet uitgewerkt.
Reageert het product op een interactie afhankelijk van de situatie waarin deze zich bevindt?	3 Afhankelijk van de (gemoeds)toestand	3 Afhankelijk van de (gemoeds)toestand	3 Afhankelijk van de (gemoeds)toestand	3 Afhankelijk van de (gemoeds)toestand
Kunnen acties van het product 'vrij' komen of weggestopt worden?	4 Actieprioriteit kan -1 zijn.	4 Actieprioriteit kan -1 zijn.	4 Actieprioriteit kan -1 zijn.	4 Actieprioriteit kan -1 zijn.

	Mediator	**Instantie**	**Proxy**	**Observer**
Hoeveel programmeerstappen zijn er nodig om een nieuwe gemoedstoestand toe te voegen? (Op hoeveel verschillende plekken, classes/methodes, in het programma is het nodig om nieuwe code toe te voegen?)	- Een instantievariabele in inputmediator - Als parameter voor de constructor van outputmediator - de aanroep van de constructor van outputmediator - In de constructor van outputmediator - Als instantievariabel	- als instantievariabele in waarnemingen - setArrayMinds in waarnemingen	- als instantievariabele in waarnemingen - setArrayMinds in waarnemingen	- als instantievariabele in GT - setGT() in GT

	e van outputmediator - Aanpassen setGT() in inputmediator - Aanpassen setGT() in outputmediator			
Hoeveel programmaregels zijn er nodig om een nieuwe gemoedstoestand toe te voegen?	6	2	2	2
Hoeveel programmeerstappen zijn er nodig om een nieuwe mind toe te voegen? (Op hoeveel verschillende plekken, classes/methodes, in het programma is het nodig om nieuwe code toe te voegen?)	- Een instantievariabele in inputmediator - Als parameter voor de constructor van outputmediator - de aanroep van de constructor van outputmediator - In de constructor van outputmediator - Als instantievariabele van outputmediator - Aanpassen setGT() in inputmediator - Aanpassen setGT() in outputmediator	- als instantievariabele in waarnemingen - array als instantievariabele in waarnemingen - setGT() in waarnemingen	- als instantievariabele in waarnemingen - array als instantievariabele in waarnemingen - setGT() in waarnemingen	- als instantievariabele in waarnemingen - array als instantievariabele in waarnemingen - double als instantievariabele in waarnemingen - in reactie toevoeging in de array - setGT() in waarnemingen
Hoeveel programmaregels zijn er nodig om een nieuwe mind toe te voegen?	7	3	3	5