Simulation of emotions of agents in virtual environments using neural networks

Aard-Jan van Kesteren
Rieks op den Akker
Mannes Poel
Anton Nijholt

Department of Computer Science
University of Twente
Enschede, the Netherlands
e-mail : anijholt@cs.utwente.nl

Abstract

A distributed architecture for a system simulating the emotional state of an agent acting in a virtual environment is presented. The system is an implementation of an event-appraisal model of emotional behaviour and uses neural networks to learn how the emotional state should be influenced by the occurrence of environmental and internal stimuli. A part of the modular system is domain-independent. The system can easily be adapted for handling different events that influence the emotional state. A first prototype and a testbed for this architecture are presented.

1 Introduction

This paper presents results of ongoing research on human-computer interaction with intelligent conversational agents in virtual environments. It reports work in a project aiming at the design and agent-oriented implementation of a multi-user, multi-agent and multi-modal interactive environment [Nijholt 1999].
Our interest in emotions and our objective to simulate emotional behaviour finds its primary motivation in the hypothesis that avatars that somehow show emotions in the way they behave are more *believable* than agents that lack these human qualities.
There is another motive for incorporating emotions in the design of synthetic agents; often mentioned by AI-researchers. A. Damasio came with neurological evidence for the importance of emotions in human decision processes [Damasio 1994]. The interaction between emotions and the rational processes that underlie the making of decisions could possibly explain why humans are so good in making decisions in a context of uncertainty. Answering the question whether machines can have emotions, M. Minsky even stated*: "The question is not whether intelligent machines can have any emotions, but whether machines can be intelligent without any emotions"* [Minsky 1986]. Hence the idea to give synthetic agents emotions in order to make it possible that they perform whatever tasks they have in a more intelligent way. Whether there is some sense in this depends on the notion of emotion one has in mind and on the analysis of the aspects involved in the process of decision-making.

To model and to simulate all of the relevant aspects of emotions in a comprehensive system is a project for years, so it shouldn't come as a surprise that every group that's conducting research on the topic of emotional agents has made it's own decisions as to what aspects are built into their system, and what is left out for (hopefully) future development. Most groups [Reilly & Bates 1992, El-Nasr & Yen 1988, Velásquez 1997] have taken a wide range of topics related to emotions (e.g. emotion, facial expressions and emotional behaviour) into account for their emotional systems. Until now we have only looked at the topic "*simulation of emotions*", as we think it's best to focus on one topic at a time.
We think of an emotional agent as an agent that has among other things (e.g. believes, desires and intentions) an *emotional state*. This emotional state can be altered by *stimuli* from the environment or by *stimuli* from internal elements of an agent (e.g. decisions, future expectations, memory, etc.). "Simulation of emotions" consists of the elements that *appraise* the stimuli and the processes that take care of the *dynamics* of the emotional state. In principle, every kind of behaviour and internal process of an agent can make use of the emotional state. For instance, the emotional state can be used to steer the facial expressions and the decision process. Consequently, our agents don't show emotions yet. A user can only get an impression of what the agent "feels", by looking at a representation of the emotional state.
For our model of emotions, we have two basic concerns. We want our model to resemble the emotional processes, as they exist within the human brain, as closely as possible and consequently we don't want to diverge too much from the leading theories of emotions. On the other hand we also want to develop a computational approach, which can be easily used to implement an emotional agent.

Our research is very much ongoing and some of the elements presented in this paper haven't been properly tested yet. This paper will give an overview of how we dealt with the problem of emotions until now and what our approach for the future will be. Furthermore an architecture for the simulation of emotions will be introduced.
First the relevant models of emotions will be discussed shortly. After that we will globally describe what our approach is, followed by a description of the environment that we used as a testbed. Subsequently our *distributed architecture* will be introduced, by firstly explaining the architecture as a whole and secondly by discussing the various subparts in more detail. A description of a first prototype will be given next to further clarify the architecture and to present some first test results. The paper will be concluded by a comparison with another model and some conclusive remarks.

2 Theories of emotion

In this section, two theories on emotion will be introduced. How we used these theories for our model will be explained in the next section.

Event appraisal models[1] have as assumption, that it's the *subjective, cognitive interpretation of events* that explains the experienced emotions.
Event appraisal models mainly focus on the question how events are appraised and in which direction (and with what velocity) the emotional state is likely to shift. We think of this as the *emotional meaning* of an event. Usually event appraisal models don't model the *dynamics of the emotional state*: if an event is appraised, what will then become the new emotional state? For an AI-model of emotion this question is as important as a natural appraisal of events.

[1] For instance [Ortony, Clore & Collins 1988] and [Roseman, Jose & Spindel 1990]

One of the leading event appraisal models is the model of Ortony, Clore and Collins (The OCC Model) [Ortony, Clore & Collins 1988]. The model only looks at *emotion types*. Each type contains a large number of emotional states, which may differ in intensity (e.g. the emotional states: worried, scared and terrified belong to the emotion type fear), but may also differ because of other reasons, such as the cause of the emotion (e.g. the emotional state heartache differs from other emotional states belonging to the emotion type distress). By focusing on emotion types instead of emotions, the whole field of emotions gets more comprehensible.
In order to appraise a particular event, there are three different aspects that can be focused on. Every aspect has a different set of emotion types attached to it. We will elaborate a bit on the different aspects by means of an example. Imagine that someone hears that his sister-in-law has just killed one of his children. If that someone focuses on the *consequences of the event*, he'll probably experience distress. If he focuses on the *action of the agent* (the sister-in-law), he'll probably experience reproach. And if he looks at the *aspects of an object* (aspects of the sister-in-law) he'll probably experience hate.
Associated with each aspect is a different kind of knowledge representation. If one focuses on the consequences of an event, *goals* are of importance. If one focuses on the action of an agent, *standards* are of importance and if one focuses on the action of an agent, *attitudes* are relevant.
A strong point of the OCC model, which makes it particularly useful for designing an AI-model of emotion, is that the model includes a complete set of variables that influence the intensity of the emotions. The variables can be global, which means that the variable influences all of the emotion types, or local, which means that the variable influences only some of the emotion types. We will shortly introduce the (in our opinion) three most important local variables. The variable *desirability* is important if one focuses on the consequences of an event. The variable *praiseworthiness* is important if one focuses on the actions of an agent and the variable *appealingness* is important if one focuses on the aspects of an object.

Besides the event appraisal models, there are more theories of emotions we found interesting and useful. One of those is Frijda's notion of emotion [Frijda 1986]. According to him, emotion is *action readiness change*, which refers to the inner disposition (and the absence) for performing actions and achieving relational change, as it exists at a particular moment. The experience of emotion largely consists of experienced action readiness or unreadiness: an impulse to flee, strike or embrace; or lack of impulse, apathy, listlessness. So, emotion is not the same as the feeling of emotion. A consequence of this is, that an explicit representation of emotions is unlikely to exist.

3 Our approach

Our research goal is to develop a human like agent that shows natural emotional behaviour. Yet, as we didn't want to start too ambitiously we are firstly focusing on simple agents in a simulated environment. The agents and the environment shall be introduced in the next section.

Our model is very much an implementation of the more theoretical OCC model. First of all, only *events* can alter the emotional state. Furthermore, the representation of emotion consists of the same *emotion types* the OCC model distinguishes, the three *aspects* of events also play a central role in our model and the same *variables* (the properties of events that are important for emotions) as in the OCC model will be used. Furthermore, the agents will contain a representation of *goals*, *standards* and *attitudes*.

The current state of research on emotions is, that a lot of work has been done on philosophical questions like: “What are emotions?”, “What influences emotions?”, “Which emotions can be distinguished?” and “What is the connection between cognition and emotion?”, but no complete quantitative models of emotions (besides some AI-models) have been developed yet. Apparently it’s very difficult to describe the emotional behaviour in a complete and precise manner. On the other hand we noticed that it’s relatively easy to imagine in practical situations what would be natural behaviour.
Therefore we chose an approach in which the system has to *learn* about emotional processes and rules from examples provided by a trainer. Most other research groups have used a completely opposite approach. In their systems the designer has to define the emotional processes and rules, from which natural emotional behaviour has to *originate*. We’ll use the common terms *top-down approach* for our approach and *bottom-up approach* for the other approach (Figure 1 gives a schematic representation of the two concepts).

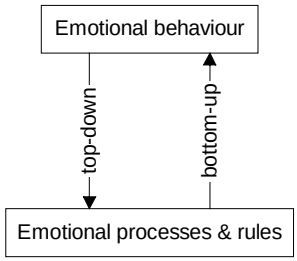

Figure 1 A top-down approach vs. a bottom up approach

For our approach trainingsdata is needed, which will be obtained through annotation. We realize that annotation is a time-consuming and tedious task and therefore we defined the architecture in such a way, that a minimal amount of trainingsdata is required.
Another advantage of a top-down approach above a bottom-up approach is, that it’s more flexible; the same system can be used for more applications, which is for instance practical if one wants to design agents with different personalities. Using a different trainingsset for each personality is an easy way to do this. Of course the same effect may be obtained in a bottom-up approach by changing some of the parameters, but we strongly doubt if parameters can be defined in such a way that it offers the same flexibility as a top-down approach does.

As we want to design a system that only needs a small amount of trainingsdata, it’s especially important that the system is able to generalize well. This is one of the proven qualities of *neural networks*, and accordingly we will use those for our system. For the emotion domain, it isn’t very important that the system works very accurately in a quantitative way (the chosen numerical representation of emotions is a rough estimate anyway), as long as it performs well in a qualitative way and reasonably well in a quantitative way. Therefore relatively small neural networks can be used, which should enhance the generalization capacities of the system even more.
As Frijda’s theory of emotions clearly suggests, an explicit representation of emotions is unlikely to exist within the human brain. We do not only think that an explicit representation of emotions is unlikely from a cognitive point of view, but we also think that a wrongly chosen representation might have a strong negative effect on the capacities of the system to show a wide range of emotional phenomena. By using *recurrent neural networks*, the system can learn an optimal, implicit representation of emotions itself, which we see as a major advantage of a connectionist approach.

4 The environment and the agents

In order to develop and test a system that simulates emotions, an environment is needed, which is inhabited by agents that can have emotions. As we want to model a broad range of emotions, the environment and the agents must satisfy a number of requirements.

As we make use of the OCC model as basis for our AI-model of emotion, it's important that the agents have an explicit or implicit representation of *goals*, *standards* and *attitudes*[1]. Agents should also be able to translate observations into terms of these three concepts. In practice this last requirement has the following consequences:

- Agents should be able to see if an event satisfies a particular goal or has a positive or negative effect on the probability that a particular goal will be satisfied (important for the emotion types *joy* and *distress*).
- Agents must be able to reason about the *future* and should be able to change their *expectations* of the future as a consequence of events. Expectations about goals are important for emotion types such as *hope* and *fear*.
- Agents must have some kind of memory about previous expectations and should be able to compare new events to these previous expectations (important for the emotion types *relief*, *fears-confirmed*, *satisfaction* and *disappointment*).
- The same kind of reasoning the agents must be able to do for themselves, they must be able to do for other agents also (important for emotion types *happy-for*, *resentment*, *gloating* and *pity*).
- Agents must be able to compare actions of themselves or of other agents to their standards (important for emotion types *pride*, *shame*, *admiration* and *reproach*).
- Finally, agents must be able to compare aspects of objects or agents to their attitudes (important for emotion types *love* and *hate*).

The future must be partly *uncertain* for the agents. If it's precisely known what's going to happen in the future, emotions such as hope and fear cannot exist. A way to guarantee this is by making the environment *nondeterministic*. Another important property of the environment is, that it contains *multiple agents*, as some emotions depend on actions of other agents or on the consequences of events for other agents.
As the user has to annotate natural emotional behaviour, it's important that the user is able to put himself in the position of an agent. A user can only do this, if the behaviour of the agent is *believable* enough and if the user has the exact same knowledge as the agent. A final requirement of the environment is, that the situations are *complex* enough, such that interesting emotional behaviour can arise.
We set out to define and implement an environment, which satisfies all the previously stated constraints. For the domain we were inspired by [Inoue, Kawabata & Kobayashi 1996]. In Figure 2 a picture of the domain can be seen.

[1] We refer to [Reilly & Bates 1992] for a good example of an implementation of these concepts.

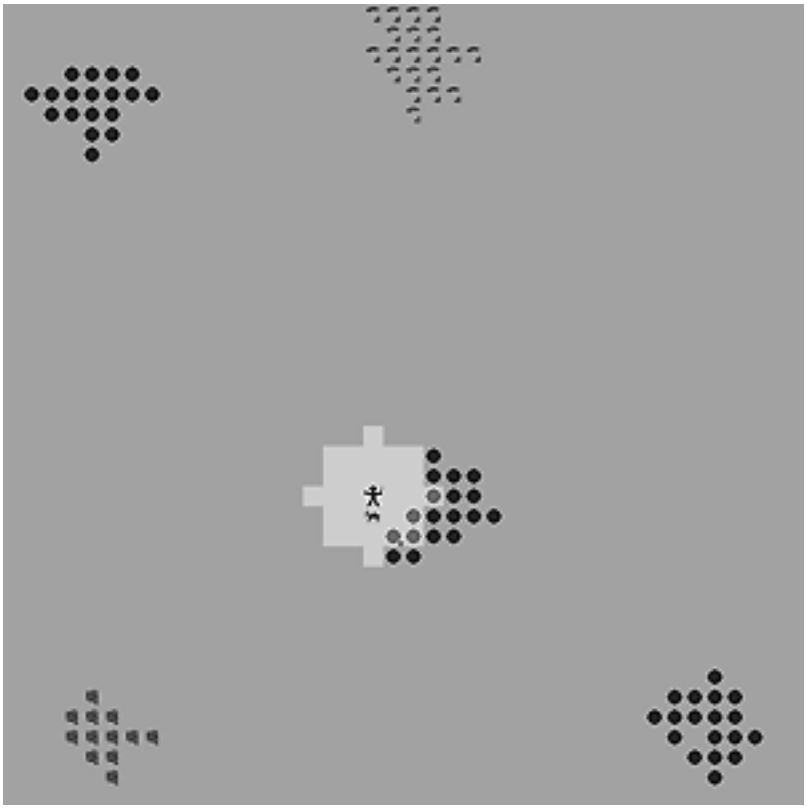

Figure 2 Picture of the environment

The domain is a gridworld containing grass, water pools that can be dry or contain water, apple trees that can have apples growing and rocks, with possibly herbs growing on it. The status of the trees, water pools and rocks constantly changes in a nondeterministic way.
Multiple agents inhabit the gridworld. One of the agents can be seen in the middle of the picture. An agent can only see a small part of the world (the light part). Currently the agent sees one predator (directly beneath the agent) and one tree with an apple (directly south-east of the predator). An agent only knows where the visible agents and predators are and he knows the location of all the trees, rocks and water pools, but the status is only known of the visible trees, rocks and water pools.

A trainer can only see the things an agent can see; in the light part, a trainer can see everything, just like an agent, and in the dark part, a trainer sees only the locations of the trees, rocks and water pools, as this is also knowledge that an agent possesses. Because of this, it's easier for a trainer to imagine what an agent feels. A trainer also knows how hungry, thirsty and healthy an agent is, which events have occurred lately and what his current action is.

Agents need food and water, which can be supplied by apple trees and water pools. There are predators that can be dangerous for the agents. An attack by a predator affects the health of an agent. An agent can regain health by eating an herb.
To a large extent, the behaviour of a predator is random. The only exception is when a predator is in the neighbourhood of food, water or an herb. Then it will stay there because it knows that agents will come there sooner or later. This means that the presence of a predator is an indication that food, water or an herb may be nearby. On the other hand the presence of food, water or an herb is an indication that a predator may be nearby.

Agents are able to make decisions and have all the previously described capacities. There is a social order between agents and they can choose (dependent on their character) either to follow the leader of a group or to go their own way. Social grouping is a result of common concerns. An agent knows how thirsty, hungry and healthy the agents in his group are.

5 The system

5.1 The architecture

In this section, the architecture for our system will be introduced. First a global overview of the system will be given and after that the various concepts and subparts will be discussed in more detail. For the system we have chosen a *distributed architecture*, as depicted in Figure 3. The motivations for this particular architecture will be given after a brief introduction.

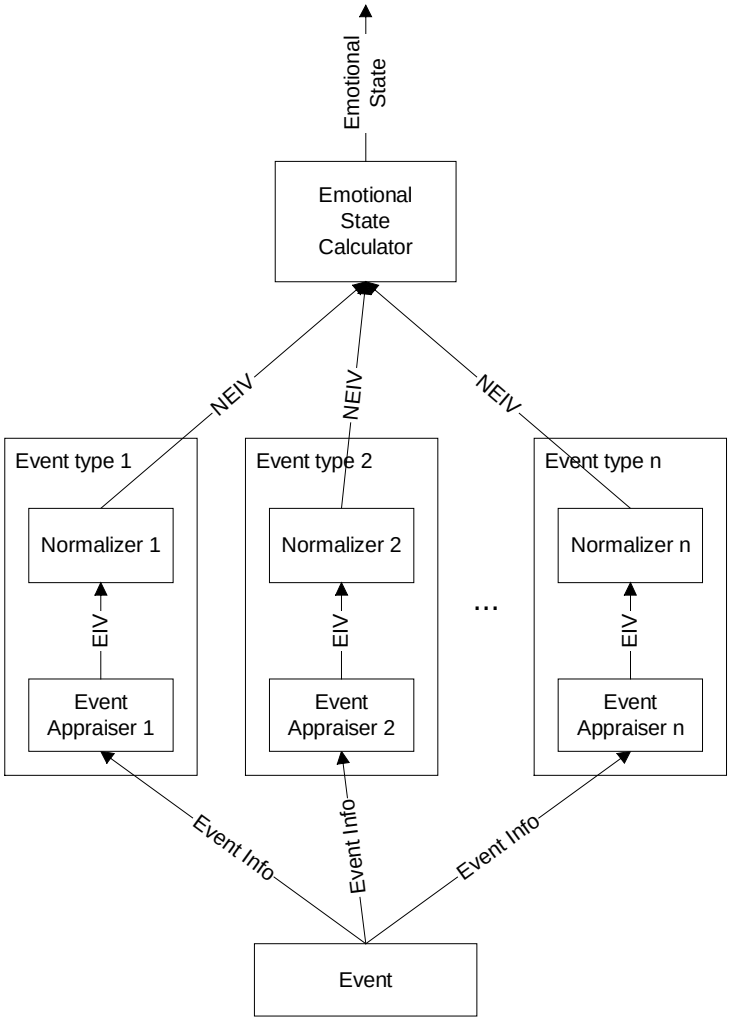

Figure 3 Schematic representation of the architecture

From a functional perspective, the task of the system is to convert an event with particular properties into a new emotional state. In order to convert an event to a new emotional state, two phases are distinguished.
The first phase is to appraise *the emotional meaning* of the event. This is the task of an *Event Appraiser*. Each event belongs to *maximally one* event type (e.g. the event: An agent sees an apple at a particular location while he has a particular desire for food belongs to the "Apple_spotted" event type). For each relevant event type, one event appraiser is defined. The content of the *event information* depends on the type of the event and can vary greatly from one event type to another. To sum up, when an event occurs, first the event type of the event has to be established. Then it's exactly known which Event Appraiser has to be used and what kind of information is needed for appraisal.
In the second phase, the *Emotional State Calculator* (the ESC) uses a numerical representation of the emotional meaning of the event (firstly stored in an *EIV* and secondly stored in an NEIV; both will be explained in detail later on) to calculate the new emotional state. A history of previous emotional states and emotional events is *implicitly* stored within the ESC, which is implemented by a recurrent neural network. The *Normalizers* form an interface between the Event Appraisers and the ESC. The task of a Normalizer is to normalize the data in an EIV, such that the ESC can treat all of the Event Appraiser in a *uniform* way. The Normalizers will also be implemented by neural networks.
This approach has a few qualities:

- **Conformity with emotion theory**. As mentioned before, emotion theorists are inclined to see the appraisal of events as a separate problem. In this architecture, the appraisal of events is explicitly a subpart of the system.
- **Conformity with human intuition**. In our first annotation attempts, we noticed that if we had to predict a new emotional state, given an event and the old emotional state, that we were inclined to first look only at the event and imagine what kind of emotional consequences this event might have and only after that we looked at the old emotional state and predicted a new emotional state. Therefore the way the system works conforms at least to our intuition and maybe also to the intuition of most people. This is an important property, as this means that the annotation task (which will be explained in the section on training) will be relatively easy.
- **Trainability**. The different subparts can be kept relatively small, as a result of which the various neural networks can be trained more easily and with a smaller amount of trainingsdata.
- **Possibility of incremental development**. This approach allows that events are added to the system one-by-one. This is a big advantage from a software engineering point of view.
- **Scalability**. If the domain gets more complicated (meaning more types of events), the amount of neural networks within the system will increase, but the complexity of the neural networks will remain the same. This means that the scalability of the system shouldn't be a problem, if the amount of types of events increases. Whether the system is scalable in other dimensions also (e.g. the amount of modelled emotions) is one of the open research questions remaining.
- **Reusability of the ESC**. The ESC is domain-independent and therefore can be reused for different agents in different domains.

5.2 Emotion Impulse Vectors

An *Emotion Impulse Vector* (an EIV) is a data structure especially suited for storing the *emotional meaning* of an event. The structure of an EIV has been chosen as follows:

$$EIV = \langle ei_1, ei_2, ..., ei_n \rangle$$

in which every emotion type e_i has a corresponding *emotion impulse* ei_i. The emotion impulse ei_i indicates how the emotion e_i is likely to change. For instance a high positive value for ei_i means that the EIV probably will have a large positive effect on the emotion e_i.
A *Normalized Emotion Impulse Vector* (an NEIV) is a normalized version of an EIV. How the normalization functions, will be explained later on.

5.3 Event Appraisal

The task of an Event Appraiser is to assess the emotional meaning of an event to construct an EIV. Every event and domain has its own properties and therefore an Event Appraiser should be constructed separately for every event type. This way *a priori knowledge* about the event and domain can be used in an optimal way. A consequence of this is, that an Event Appraiser can be implemented by a neural network, but can just as well be implemented by a rule-based system, a function or something else.

Defining or training an Event Appraiser that can immediately communicate with the ESC in such a way that the behaviour of the total system is precisely as required, would mean a full understanding of the ESC and would at least make defining or training the Event Appraiser a very difficult task. Therefore we introduced the Normalizers. Their task is to convert an EIV to an NEIV such that the behaviour of the total system works as required. As the Normalizers only have to do scaling operations, and therefore only have to learn linear or near linear functions, the Normalizers can be simple feedforward networks with only a few hidden neurons.
A nice property of using Normalizers is that the Normalizers can easily correct quantitative errors of the Event Appraisers and the ESC. Therefore it's not very important that the Event Appraisers and the ESC work well in a quantitative way, just as long as they work well in a qualitative way.

The inputs of an Event Appraiser are the variables defined in the OCC model or properties of an event that can be used to calculate the variables. The current *action* of the agent is a special kind of property, which can be important to appraise an event. For instance, an agent is bound to appraise the event of spotting a predator differently if he's attacking than if he's fleeing.

5.4 The Emotional State Calculator

The task of ESC is to calculate a new emotional state given a NEIV. Although the ESC is one neural network, conceptually it can be seen as two different parts. The first part, the core of the ESC, is a recurrent network and models all of the emotional behaviour. The state of the recurrent network can be seen as an *implicit emotional state*. We will implement the implicit emotional state with an Elman network [Elman 1990].
The second part of the ESC is a feed forward network, which has the implicit emotional state as input. This part, the Emotional state monitor, has as task to translate the implicit emotional state, which is very hard to understand and use to influence the behaviour, into an *explicit emotional state*. The structure of the explicit emotional state has been chosen as follows:

Explicit emotional state = $<e_1, e_2, \ldots, e_n>$

in which e_i denotes the intensity of an emotion type. Every e_i refers to a different emotion type. A schematic representation of the ESC is depicted in Figure 4.

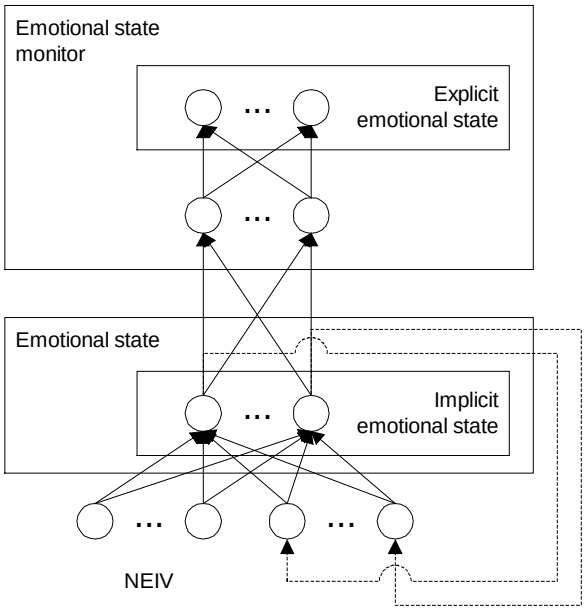

Figure 4 Schematic representation of the Emotional State Calculator. A dashed line represents a time delay of 1.

5.5 Decay

The *decay* of an emotional state is a topic that has received special attention in most other AI-models of emotion and is also a topic mentioned in [Ortony, Clore & Collins 1988] as being a specific point of concern for a computational model of emotion. In our system the decay is modeled as an event type with a corresponding Event Appraiser and Normalizer. An event of this type occurs by definition after a fixed interval of time and sees to it that the emotional state eventually returns to a state of rest. A property that determines the appraisal of this event is among other thing the current (explicit or implicit) emotional state.

5.6 Training the system

The system is built out of three types of building blocks. For every type of building block a different training strategy has to be used. The order in which the building blocks have to be trained or defined is depicted in Figure 5.

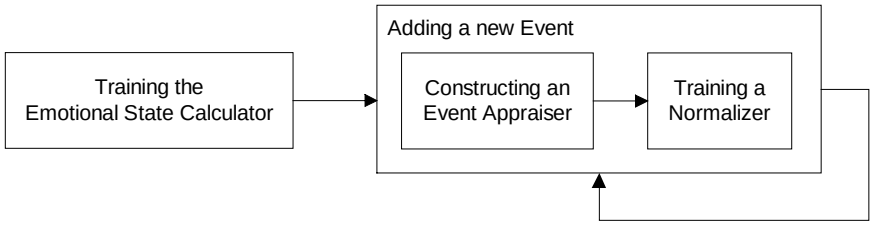

Figure 5 Schematic overview of the trainingsprocess

First the Emotional State Calculator has to be trained. This is the most difficult part of the system to train, as gathering trainingsdata is difficult. A *temporal sequence* of (NEIV, explicit emotional state) pairs is needed to train the Elman network. The NEIVs can be generated randomly, but the emotional states have to be annotated by hand.
There are two problems with this approach. First of all a reasonable amount of trainingsdata is needed and secondly the needed data is of an abstract nature, as a result of which annotating even one sample isn't an easy task. But we feel that the benefits of this tedious work are large enough to be profitable. If the Emotional State Calculator has been trained, adding events to the system is easy and an Emotional State Calculator is domain-independent and therefore only has to be trained once. The trainingsset doesn't have to be perfect, as the learning algorithm should be able to filter out the errors. Therefore the trainer can annotate quickly without worrying too much about errors, which makes the annotation task a bit easier. Also, it shouldn't be overlooked that a significant part of the difficulties of training the ESC are inherent to the complexity of the problem of emotions. A nice property of our system is that a major part of the complexity of emotions is modelled within a domain-independent part of the architecture, so that the same problems don't have to be solved over and over again for every domain and agent.

After the ESC works satisfactory, event types can be added one-by-one to the system. For a new event type, an Event Appraiser has to be trained or defined firstly. Again it's only important that an Event Appraiser works well in a qualitative way, which makes defining or training an Event Appraiser relatively easy.

From a procedural perspective, training a Normalizer is a bit less straightforward than training the ESC or an Event Appraiser. (EIV, NEIV) pairs are needed to train a Normalizer. The user cannot be expected to annotate NEIVs, because this would mean among other things that the user has to know exactly how the ESC works, which is not a realistic requirement. Therefore we propose the following approach.
Instead of (EIV, NEIV) pairs, (Event Information, Explicit emotional state, State of the ESC) triples have to be gathered. This can be done within the real domain. Every time an event of the relevant event type occurs, the user has to annotate the new explicit emotional state. The event information can be extracted from the event and the current state of the ESC can be copied easily. After the triples have been gathered, the triples have to be converted to the (EIV, NEIV) pairs. It is easy to convert the event information to an EIV, as the Event Appraiser is already trained or defined. An NEIV can be found by using the inverse of the ESC. If this inverse is known, the explicit emotional state (the output of the ESC) and the state of the ESC are enough information to calculate an NEIV. Of course the inverse is not known, but using an *iterative improvement algorithm* (e.g. a simulated annealer), can solve this final problem. This algorithm has to find the NEIV that explains the explicit emotional state the best. If needed, a second criterion can be that the NEIV has to resemble the EIV as much as possible.

Using this approach makes training a Normalizer easy for a user. Annotating an explicit emotional state is easy and as the Normalizers are usually small networks, only a small amount of trainingsdata is needed.

6 A first prototype

A first prototype of the system has been implemented. In this prototype the various subparts were kept as simple as possible, such that we could examine whether the subparts are able to *cooperate* and whether the system as a whole is capable of simulating the required behaviour. One of the consequences of this is, that the ESC has not been implemented by the proposed Elman network yet.

In the first version only two emotion types are modelled, joy and distress, and seven event types. The two emotion types joy and distress were considered as one emotion type, the joy/distress emotion type. If the intensity of this emotion type is negative, the agent experiences distress and if the intensity is positive, the agent experiences joy. So, an EIV, an NEIV and the explicit emotional state, all consist of only one value.

The ESC has been implemented as a simple linear function (and therefore the inverse is known). The previous emotional state is one of the variables of the function. Although this simple function suits us well in this phase of the research, we are convinced that such a simple approach won't be sufficient in the future and that the described approach with an Elman network is a more powerful and a cognitively more credible approach.

The Event Appraisers were kept simple too. We will clarify this part of the architecture a bit more, by focusing on the "New_predator_spotted" event type in detail. The events belonging to this event type occur when the agent sees a predator for the first time.
As mentioned before, the variables from the OCC model are the only factors that influence the intensity of an emotion type. Although more than one variable is defined in the OCC model, that influence the emotion types joy and distress, only the variable *desirability* determines the appraisal value in this version. The event information of the "New_predator_spotted" event type consists of two properties: The current health of the agent (*health*) and the amount of predators the agent has seen previously this turn (*#seen_previously)*. If the health of the agent is low, the desirability of the event should be lower, than if the health is high, as being near a predator while health is low is very dangerous. The property #seen_previously should have a negative effect on the desirability of the event, as for instance seeing a third predator is less desirable than seeing a second predator. Taking all of this into account, the EIV for this event was defined as follows:

$$EIV = \langle desirability(health, \#seen\_previously) \rangle$$
$$= \langle 1 * health - 20 * \#seen\_previously - 100 \rangle$$

Only qualitative and no quantitative arguments and no were taken into account while drawing up the formula and as a consequence the 1, 20 and 100 in the formula could for instance just as well have been 1.5, 30 and 0.

The Normalizer is a simple feed forward network, with one hidden layer, consisting of only 2 neurons. Only 20 trainingsamples were used to train the Normalizer. The other Event Appraisers and Normalizers were constructed in a similar way.

How to test the quality of the system is a hard question and a research project by itself. This version was tested in an informal way. We checked whether the emotional behaviour shown by the system corresponds to our ideas of emotions, which we also used for annotation. This was the case and therefore we can conclude that the architecture looks promising, but a lot more experiments need to be conducted before definite conclusions can be made.

7 Comparison with other models

Before defining our own models we studied other models. [Pfeifer 1988] gives a nice overview of all of the AI-models of emotion until 1988. Since then, more interesting models were put forward. [Reilly & Bates 1992] also based their model on the OCC model. In [El-Nasr & Yen 1988] fuzzy logic was used, a very useful technique for the emotion domain we think. A system which we want to discuss more thoroughly in this section is Cathexis [Velásquez 1997], as it resembles our model in a couple of ways; it also has a distributed architecture and special care has been taken to allow for maximum flexibility.

Cathexis is composed of a number of *"proto-specialist" agents* [Minsky 1986], each representing a different emotion type. Within each proto-specialist, different *sensors* are monitoring both *external and internal stimuli* for the existence of the appropriate conditions that would elicit the emotion type represented by that particular proto-specialist. Input from these sensors either increases or decreases the *intensity* of the emotion type. Parallel to this process, there's always a *decay process* going on, which sees to it that the intensity of the emotion type eventually returns to a state of rest. The way the sensors function and how the sensors and the decay function affect the intensity can be completely defined by a user. Not only the sensors can influence the intensity of an emotion, but also other proto-specialists agents can, by providing either *inhibitory* or e*xcitatory* input. A schematic overview of the system is depicted in Figure 6.
Both Cathexis and our approach have distributed architectures. The difference is, that in Cathexis the emotions are scalable and in our architecture the complexity of the domain is scalable. The advantage of making the emotions scalable is, that it makes it easier to implement a system within a simple domain. But, as the domain gets more complex, the approach of Cathexis will experience scalability problems we think, as all the proto-specialists have to perform increasingly complex task then, while no provisions have been taken to deal with this complexity.

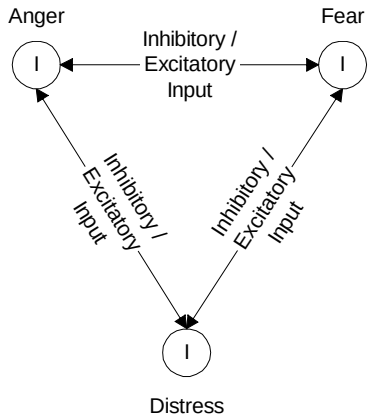

Figure 6 The "proto-specialist" agents for anger, fear and distress. (I: Intensity)

In our approach it's difficult to train the ESC for a system with multiple emotion types. But after that part has been done, adding more event types is easy.
The Cathexis system allows for a large degree of flexibility. Nevertheless we don't think the framework is powerful enough to model complex temporal emotional phenomena and phenomena in which more than two emotions participate. Our approach does have these possibilities and therefore is more powerful than Cathexis.
For Cathexis a bottom-up approach is used to make a system suitable for a particular domain. We think that it will be very difficult to precisely define all the functions and parameters in such a way that the system performs well in the case of a complex environment in which a lot of types of events can occur. In our architecture the generalization capabilities of neural networks should make it easier to handle complex environments and as a top-down approach is used, a full understanding of emotional processes and the environment is not needed; the trainer only has to know in practical cases, what the right emotional behaviour is.

8 Conclusion

An emotion theory based architecture for simulating emotions has been presented in this paper, which can be used to develop emotional agents acting within complex environments. A first prototype indicated that the various subparts of the distributed architecture are able to cooperate well and that the system is able to show natural emotional behaviour.

In this first implementation, the Emotional State Calculator (the ESC) was kept very simple. For the eventual system we have proposed a recurrent network of which the state of the network can be seen as an implicit emotional state. A foreseen problem of our approach is training the recurrent network, as it's difficult to obtain sufficient trainingsdata. But we feel that this problem is largely inherent to the complexity of emotions and that a nice property of our architecture is that a major part of this complexity is concentrated within a domain-independent part of the architecture.

In the future we want to expand our prototype with more emotions and event types. Furthermore we think that more research is needed on the ESC.

References

Damasio, A.R. (1994). *Descartes' Error: Emotion, Reason, and the Human Brain*. New York: G.P. Putnam.

Elman, J.L. (1990). Finding structure in time. *Cognitive Science*, 14, pp. 179-211.

El-Nasr, M.S., & Yen, J. (1988). Agents, Emotional Intelligence and Fuzzy Logic. *Proceedings of the IEEE NAFIPS '98*, Fl.

Frijda, N.H. (1986). *The Emotions*. New York: Cambridge University Press.

Inoue, K., Kawabata, K., & Kobayashi, H. (1996). On a Decision Making System with Emotion. *IEEE International Workshop on Robot and Human Communication*, Tokyo, Japan, pp. 461-465.

Minsky, M. (1986). *The Society of Mind*. New York: Simon and Schuster.

Nijholt, A. (1999). The Twente Virtual Theatre Environment: Agents and Interactions. *Proceedings of the fifteenth Twente Workshop on Language Technology*, pp. 147-165.

Ortony, A., Clore, G.L., & Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge, UK: Cambridge University Press.

Pfeifer, R. (1988). Artifical Intelligence Models of Emotion. In V. Hamilton, G.H. Bower, & N.H. Frijda (Eds.), *Cognitive Perspectives on Emotion and Motivation*, Netherlands: Kluwer, pp. 287-320.

Reilly, W., & Bates, J. (1992). *Building emotional Agents*, Pittsburgh, PA: Carnegie Mellon University, Technical Rep. CMU-CS-92.143.

Roseman, I.J., Jose, P.E., & Spindel, M.S. (1990). Appraisal of Emotion-Eliciting Events: Testing a Theory of Discrete Emotions. *Journal of Personality and Social Psychology*, 59(5), pp. 899-915.

Velásquez, J. (1997). Modeling Emotions and Other Motivations in Synthetic Agents. In: *Proceedings of the AAAI Conference 1997*, Providence, RI, pp. 10-15.