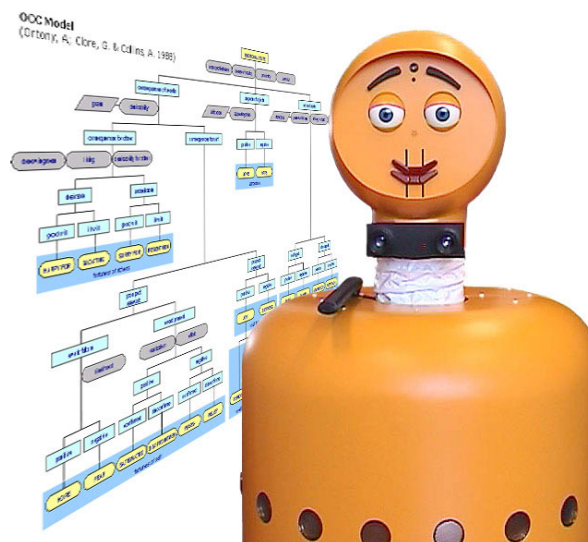


Design of an Emotion Management System for a Home Robot

Andriy Bondarev

<ISBN 90-444-0266-8>



© KONINKLIJKE PHILIPS ELECTRONICS NV 2002

All rights reserved. Reproduction or dissemination in whole or in part is prohibited without the prior written consent of the copyright holder.

Title: Design of an Emotion Management System for a Home Robot

Author(s): Andriy Bondarev a.bondarev@tue.nl
a_bondarev@ukr.net
andriy_bondarev@yahoo.com

Reviewers(s): Ko Crucq ko.crucq@philips.com
Loe L.M.G. Feijs l.m.g.feijs@tue.nl
Christoph Bartneck c.bartneck@tue.nl

Project: Mobile Intelligent (Inter)Face

Customer: Philips Research, Eindhoven, The Netherlands

Keywords: Artificial Intelligence, Emotions, OCC Model, Emotional State, Robot, Facial Expressions

Abstract: This document describes the *General Emotion Management System* (GEMS) that was created for use in the "Mobile Intelligent (Inter)Face" project. This is the software-based prototype of a general emotion management system for a robot that includes Artificial Intelligence (AI). GEMS has been created as an independent software module in the robot's architecture using the Dynamic Module Library (DML).

The GEMS module generates an emotional state of the robot each moment of actual time. The emotion generation algorithm is based on the work of O'Reilly and on the *OCC* (Ortony, Clore & Collins) model. As input the GEMS module receives information about the current robot state (*goals, actions, events, objects or agents*) from the superior robot system and detailed information from a database, which must contain all necessary variables to calculate the intensity value for each emotion.

The emotional state of the robot consists of 22 types of emotion in a multi goal environment. The GEMS module produces output in two views: the first view contains information about the *emotional state* in general and the second view contains information about the *dominant emotion*. The output of the GEMS module can be used for defining a facial expression on the mechanical face of the robot, voice output, voice modulation or body language. Also the output can be used by the superior system to influence the decision making process.

The final result is the GEMS prototype, completely integrated into the robot architecture. GEMS can be used for creating more natural, believable behaviour and interaction between the robot and a human.

Contents

1. Introduction	9
1.1. OCC model.....	9
1.2. Em – the virtual world with emotional characters	10
1.3. Facial expressions	13
1.4. Emotional Disk.....	14
1.5. Emotional machines	15
1.6. Lino – an emotional robot.....	17
1.7. Objective	19
2. GEMS overview	20
2.1. Design decisions	20
2.2. The robot's architecture and GEMS	21
2.3. GEMS architecture.....	21
2.3.1. Input information	23
2.3.2. GEMS database.....	23
2.3.3. OCC Engine	25
2.3.4. Output information	32
2.4. The INI file content	36
2.5. Expression module	37
2.6. Compatibility	37
3. User testing	38
3.1. Scenario-based user testing.....	38
3.2. "Search of the red ball"	38
3.3. "Emotional conversation"	39
3.4. "Home interaction"	39
3.5. User test procedure.....	40
3.6. Results of the demo scenario evaluation	40
4. Options for future research	42
5. Results and conclusions	45
References	47
Appendix A INI file content	49
A.1 Global constants	49

A.2 Global variables	49
A.3 Variables for log output.....	50
A.4 Information about emotions.....	51
Appendix B GEMS Database	52
A.5 Goal table (part of the file: goal.db).....	52
A.6 Object (Agent) table (part of the file: object.db)	53
A.7 Event table (part of the file: event.db)	54
A.8 Action table (part of the file: action.db)	55
Appendix C User scenarios	56
A.9 "Search of the red ball"	56
A.10 "Emotional conversation"	57
A.11 "Home interaction"	59
Appendix D Questionnaire	61



"The question is not whether intelligent machines can have any emotions, but whether machines can be intelligent without any emotions"

Marvin Minsky

1. Introduction

Sociable humanoid robots could soon be intelligent enough to enter the household as toys, cyber-pets or companions. The development of robots for domestic and health care purposes is already underway in corporate and university research. For these applications, the believability of social behaviour and ability to interact with people in a natural, intuitive, and enjoyable manner is important [1].

What are the differences between us and any, even very advanced, robots? Why do people immediately see the unnatural behaviour of any "mechanical" human? Why do people react only like "Hey, stupid machine!"? On the question "Why are you saying that?" they answered "Because this robot *does not feel* anything!" Feelings or emotions are very important for the communication process.

By emotions, people understand a particular momentary state of mind that is reflected by means of speech, gestures and, of course, facial expression. This study is largely based on the emotions defined by the model proposed by Ortony, Clore and Collins [3], commonly known as the OCC model.

1.1. OCC model

The OCC model is intended to provide a cognitive model of human emotions. The model defines three aspects of the world to which we can react emotionally: *events* of concern to us, *actions* of those we consider responsible for such events, and *objects*. These three classes of reactions lead to three classes of emotions (consequences of events, actions of agents, aspects of objects), based on evaluations in terms of different types of knowledge [3].

The OCC model provides a classification scheme (*fig. 1.1*) for 22 different emotions based on a valence reaction to *events*, *actions* and *objects* in the light of agent Goals, Standards, and Attitudes. According to the classification of emotion-eliciting situations, all types of emotion can be divided into six groups of emotions (*Table 1.1*).

A strong point of the OCC model 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.

The three most important local variables are: *desirability*, *praiseworthiness* and *appealingness*. 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 agent (*fig. 1.1*).

Because the OCC model has a strong theoretical background on explaining the generation of emotions and computing their intensities, and a structure that is simple enough to be implemented easily, it has become popular with many researchers. It converts an emotion eliciting situation into the conditions for emotion generation just by interpreting the situation at the cognitive level, thus what emotion will be generated depends on the result of the interpretation of a given situation.

Table 1.1 Basic emotions (OCC)

Positive	Negative
Well-being	
Joy	Distress
Prospect-based	
Hope	Fear
Confirmation	
Satisfaction	Fears-confirmed
Relief	Disappointment
Fortunes-of-others	
Happy-for	Sorry-for
Gloating	Resentment
Attribution	
Pride	Shame
Admiration	Reproach
Well-being / Attribution	
Gratitude	Anger
Gratification	Remorse
Attraction	
Love (Like)	Hate (Dislike)

Figure 1.1 shows the overall structure of the OCC model with all the levels of classification. On the highest level the valenced reactions are connected to the three classes of reactions which are presented as conditions (light blue rectangles). These conditions determine all requirements for each emotion to appear (yellow ovals). The emotions are connected to the six groups (blue areas). The four global variables (grey ovals) are connected to the emotional state. The rest of the variables are local and situated in places where they influence the related emotions. Some of the variables are connected to the knowledge structures (grey parallelograms).

1.2. Em – the virtual world with emotional characters

Perhaps the most comprehensive model of personality developed from the OCC model is O'Reilly's work (1996) [2]. In his emotion system, called Em, emotions are constructed, following the OCC model, from goals, the approval of others, other emotions, and hardwired attitudes (*fig. 1.2*). Happiness and sadness, for instance, are generated as a result of goal success or failure, while fear and hope are based on a belief in the goal's success or failure. Other emotions, such as pride, shame, reproach, and admiration are based on whether the action is approved or disapproved by other agents. This is determined by an agent's standards. Anger, gratitude, and remorse are composites of other emotions.

In Em, personality is expressed by manipulating the emotional parameters. The importance of specific goal attainments and the nature of the preset attitudes and standards can vary, as can the actions selected given the presence of a particular emotion. In addition, the propensity to react emotionally and in specific ways can differ. Finally, an agent's estimate of the likelihood of success in obtaining a goal can be skewed optimistically or pessimistically.

OCC Model

(Ortony A., Clore G., & Collins A., 1988)

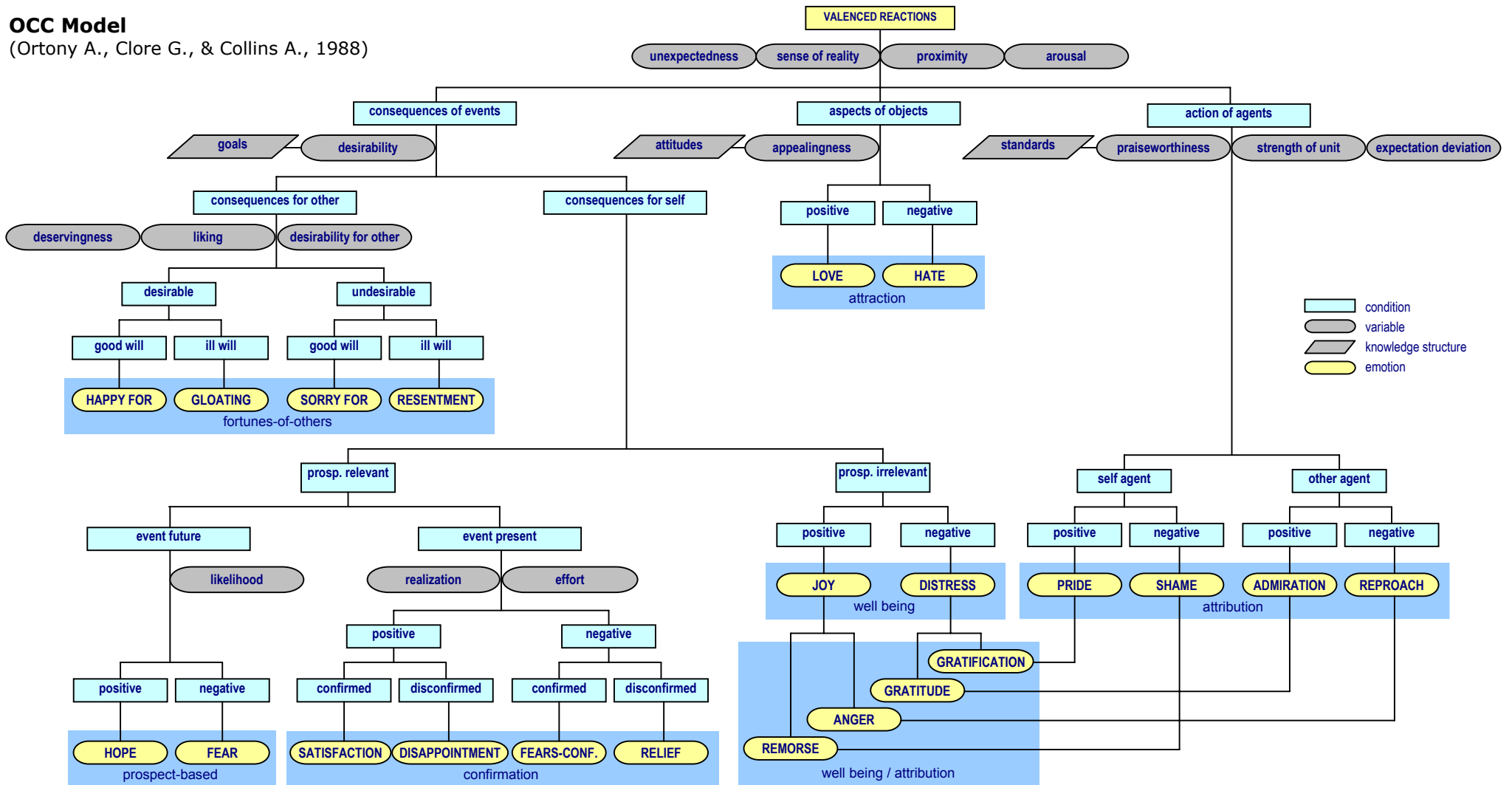


Figure 1.1 OCC Model

Em Model

(W. Scott Neal Reilly, 1996)

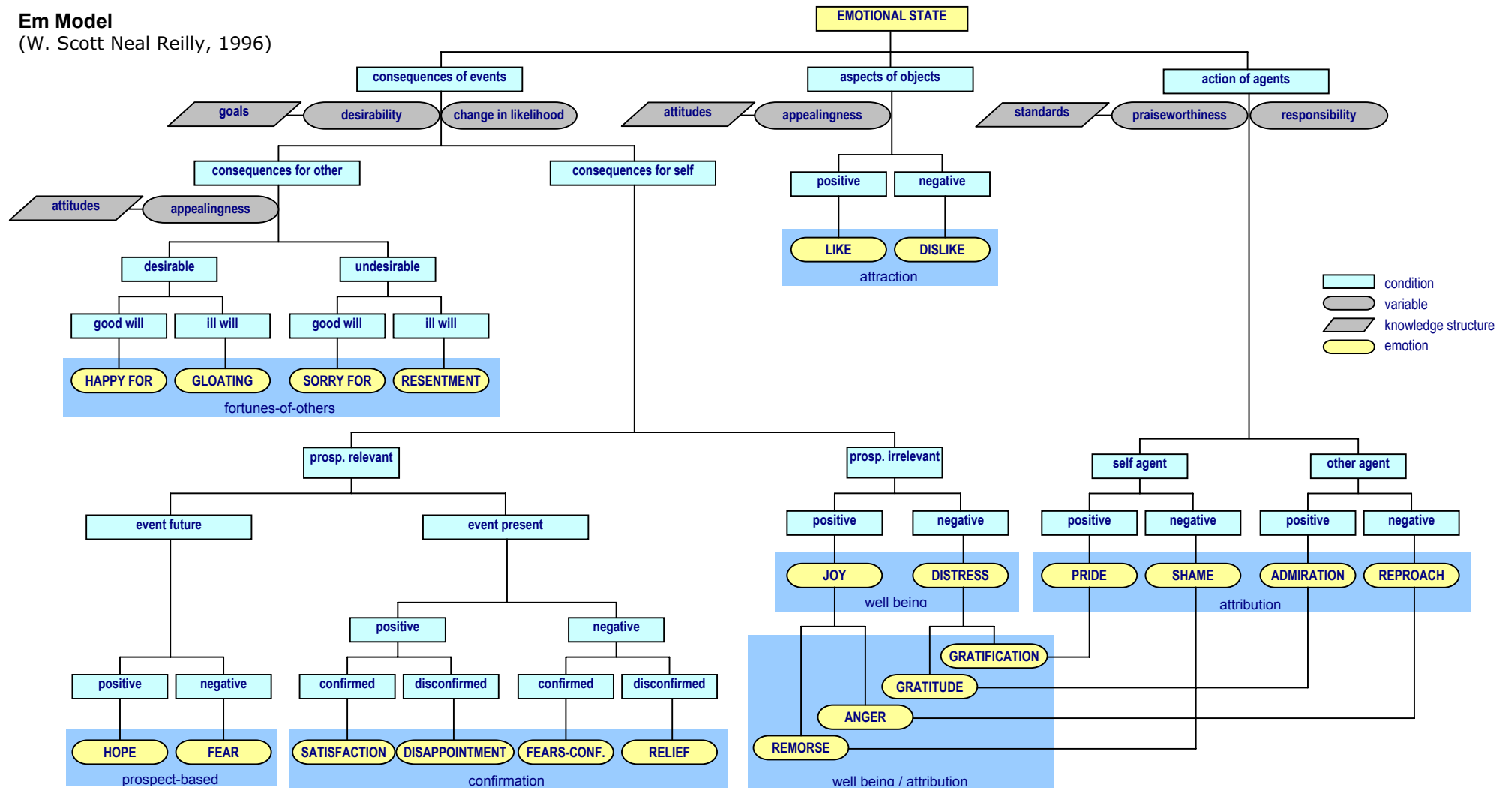


Figure 1.2 Em Model

Figure 1.2 shows the differences between the OCC model and the Em system. The Em system has the same structure for the conditions. The Em system has a more simple structure for the variables and doesn't have the global variables. The Em system has different names for the attraction emotions (Like and Dislike instead of Love and Hate).

In general, the Em system combines the mathematical approach (the Em system defines all necessary formulas to calculate the intensity of the emotions) and practical implementation of the OCC model in a virtual world with emotional characters to define an *emotional state* of the characters.

The emotional state can be used to provide a more natural interaction, provide affective feedback by expressing the emotional state in its body language, speech modulation, and facial expression.

1.3. Facial expressions

The emotional state (or emotions) is an internal characteristic of an agent. The emotions of the agent are not visible for an external spectator while the agent does not express it. The agent can express its internal emotional state with the help of its facial expressions.

There are six basic facial expressions defined by Ekman [7] recognized as universal by many facial expression and emotion researchers. These basic expressions are *joy*, *sadness*, *anger*, *surprise*, *fear*, and *disgust* (fig. 1.3). They are very useful for facial animation, and can be combined to obtain other expressions.

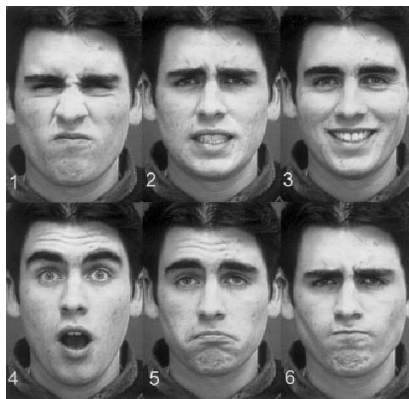


Figure 1.3 Basic facial expressions

There is a partial overlap between the expressions proposed by Ekman and the ones stated by the OCC model. Only four expressions (joy, sadness, fear and anger) are defined in the OCC model. Disgust and surprise do not find place in the OCC model, mainly because disgust does not involve much cognitive processing and surprise does not correspond to valenced reactions.

The emotions defined by the OCC model are too many in number to be directly used in the computation of the facial expressions. At the same time, they are important and necessary for making the interaction process rich with respect to the emotional expressions.

1.4. Emotional Disk

Humanoids with expressive faces have become popular in social user interfaces. There is a demand for tools with which a non-professional user can make a variety of expressive and appealing faces with little effort and resources.

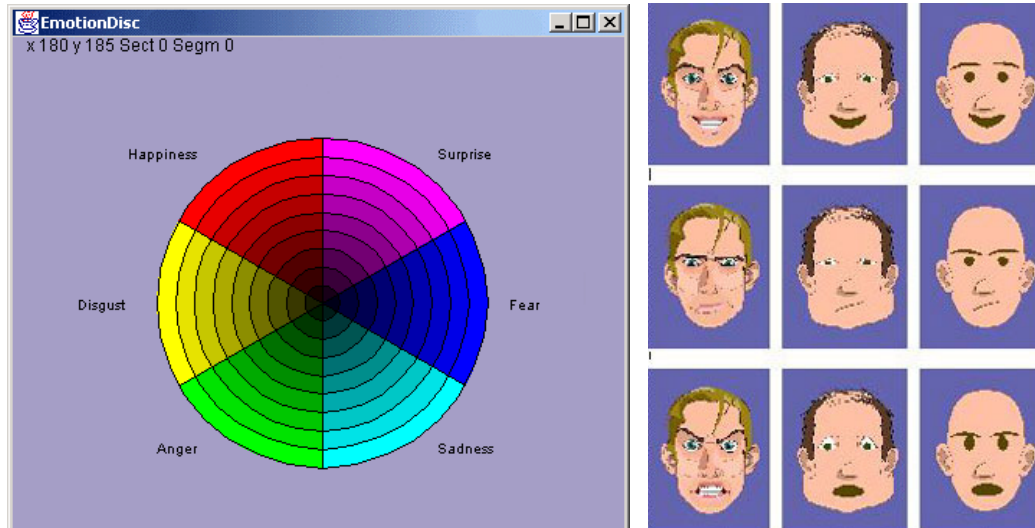


Figure 1.4 Emotion Disk with the six basic expressions. Three faces made from facial repertoire elements, showing identical expressions in each row.

CharToon was developed by Epictoid company [15] in 2001. CharToon is a vector-graphics based facial animation tool written in Java, with which one can construct faces which can be animated [5]. The CharToon system dedicated to design and animate non-photorealistic, cartoon faces for web and other applications. Novel features are based on the concept of a facial repertoire, containing ready-to-use facial expressions. With constraint-based animation editing facilities, a user can define expressions on a high level and design the dynamics and

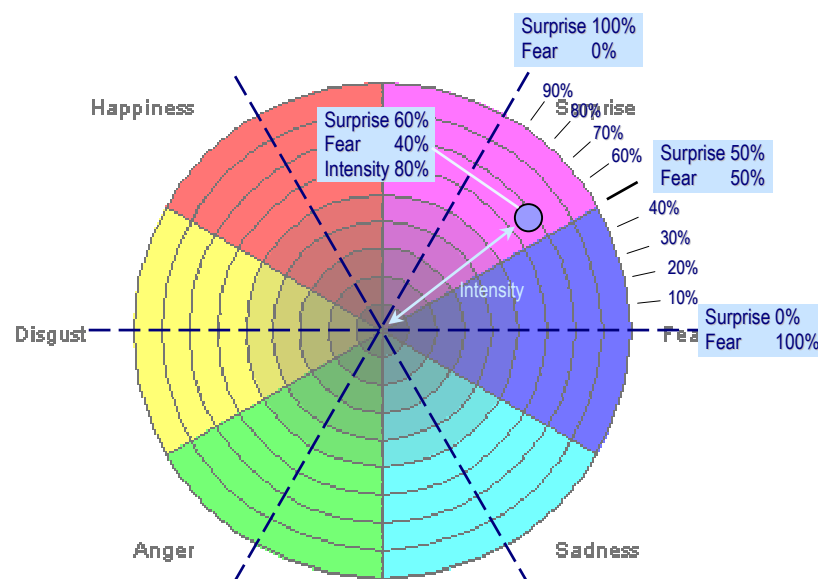


Figure 1.5 Two-dimensional emotional space

behaviour of a face.

The extension to CharToon is the Emotion Disc (*fig. 1.4*). The Emotion Disc is a graphical user interface, which enables the generation and exploration of facial expressions. The snapshots of the initial six basic expressions – joy, surprise, fear, sadness, anger and disgust – have to be designed. The further elements of the emotion space are generated by blending two of the given emotions in a certain way. The design of the Emotion Disc is based on an almost 50-year old observation by Schlosberg [9], stating that the six basic emotional expressions are perceptually related in such a way that they can be arranged in a two-dimensional space along a circle. The continuous space of Emotion Disc is, accordingly, a round disc showing a neutral face in the centre and maximal expressions on the perimeter (*fig. 1.5*). The Emotion Disc can be used as a direct controller of the expression of an avatar's face.

1.5. Emotional machines

Many companies and researchers are working on the synthesis of emotional expressions. The application areas of the research are from computer games, to software agents, toys and robots [25].

Many studies have been performed to integrate emotions into machines [17]. Considerable amounts of research on autonomous robots have been carried out.

Toy and pet robots (*fig. 1.6*) have physical bodies and behave actively while generating motivations by themselves. They interact with human beings physically.



Figure 1.6 Toy and pet robots

An embodied home character, such as eMuu (*fig. 1.7*), could be the social entity necessary to provide natural dialogues [19]. This character would have to be able to utilize the full range of communication channels, including the emotional expressions, to give intuitive feedback to the user [18].

Eventually sociable robots will assist us in our daily lives, as collaborators and companions. During the last years different humanoid robots have been developed (*fig. 1.8*). They can communicate in a manner that supports the natural communication modalities of humans. Examples include facial expression, body posture, gesture, gaze direction, and voice. The ability for people to naturally communicate with these machines is important.



Figure 1.7 eMuu – An Embodied Emotional Character for the Ambient Intelligent Home

The Sociable Machines Project (MIT) develops an expressive anthropomorphic robot called Kismet (fig. 1.9) that engages people in natural and expressive face-to-face interaction [20]. This work integrates theories and concepts of social development and psychology to enable Kismet to enter into natural and intuitive social interaction with a human. To do this, Kismet perceives a variety of natural social cues from visual and auditory channels, and delivers social signals to the human through gaze direction, facial expression, body posture, and vocal babbles. The robot has been designed to support several social cues and skills that could ultimately play an important role in socially situated learning with a human instructor.

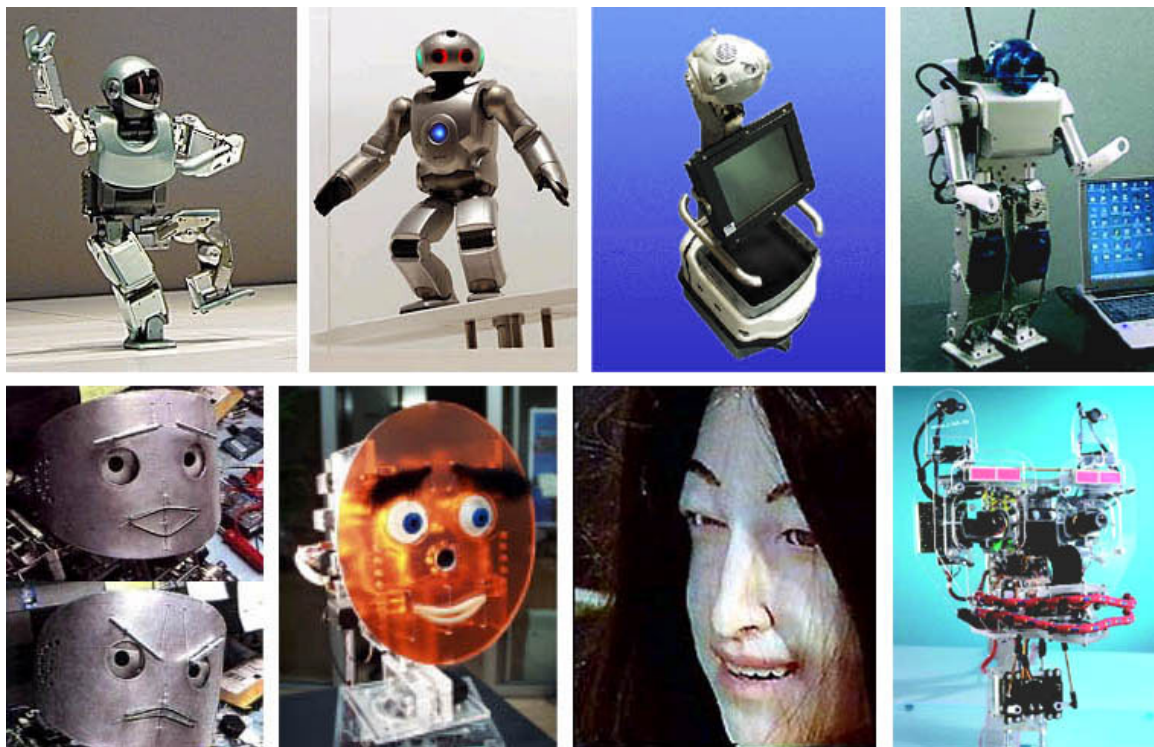


Figure 1.8 Humanoid robots

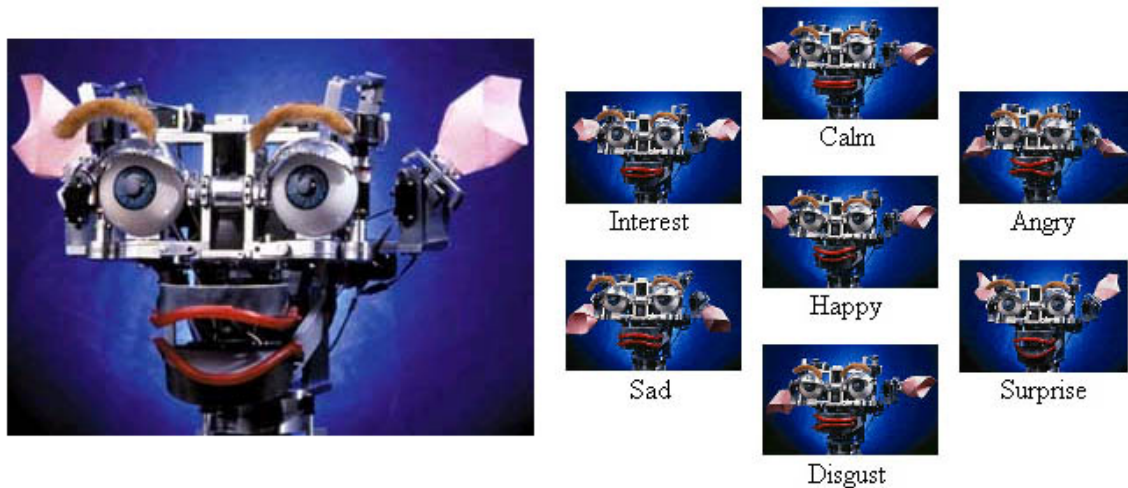


Figure 1.9 Kismet is an expressive robotic creature

1.6. Lino – an emotional robot

Lino – the domestic robot (*fig. 1.10*) is being developed by the department 'Mechanics, Heat and Particle Optics' (Philips Research, Eindhoven, The Netherlands). The robot is not meant to be industrial, but rather a helpful home appliance with which a user can interact.

Using several microphones Lino can hear (within 360 degrees) where someone is talking (i.e. speaker localisation) and with a directional microphone it can pick up voice-commands. As an appliance it can interact with all other appliances in the house. Using sonar and a 3D camera it will learn its surroundings and know how to manoeuvre in the house. A camera is also used to recognize the faces of people and objects in the room. Lino can move itself by using the wheels of a pioneer platform.



Figure 1.10 Lino – domestic robot



Figure 1.11 Lino's mechanical face

As a domestic robot, Lino is intended to interact with people. To naturally interact with people Lino has its head equipped with a mechanical face (*fig. 1.11*). Lino's face has several moving mechanical parts, which are simulating a human face (eyes, eyelids, eyebrows and lips) and can be controlled to show emotions. The head is mounted on the mechanical neck, which gives the possibility to move the head left - right, up - down and forward - backward.

To be socially acceptable during the interaction process, Lino can express six types of facial expressions on his mechanical face: happiness, surprise, fear, sadness, anger and disgust (*fig. 1.12*). Also Lino has the possibility to blend any two of these facial expressions.

The facial expression on Lino's face can be produced manually using the Emotional Disk from the CharToon (*fig. 1.12*). The disk gives the visual interpretation of the facial expression space

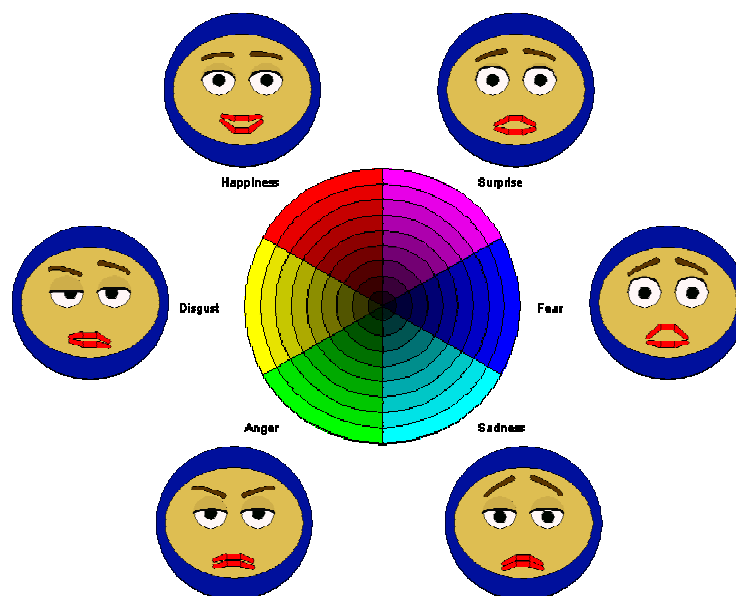


Figure 1.12 Six Lino's facial expressions

as a circle with six sectors. Each sector represents one of the facial expressions. Each dot on the disk represents the particular parameters of the facial expression as shown on *Figure 1.5* [5].

The robot has a distributed software system that provides easy communication between all processes. Dynamic Module Library (DML) [4] is a framework that provides a dynamic communication between the software processes (modules) of the robot. DML has been created on top of the Parallel Virtual Machine library [6], which provides the distributed and OS-independent functionality. It allows for the modules to be started, stopped, paused and resumed from its own internal code or from the outside by another module. The modules will run user-defined code at specific times. This can be when data is received or at specific time intervals. The modules communicate via ports that can be dynamically created and represent software-equivalents of hardware-ports.

These ports communicate via direct connections, that can be created and removed from its own internal code, automatically by the DML-framework or from outside by another module. There are input, output and bi-directional ports. The output and bi-directional ports can send arrays of integers, doubles or strings. The ports can contain the last received/sent value, the first received/sent value or a buffer filled with all unread values. Another feature of the ports is that they can be push or pull. This dictates how the data is sent between an input and output port. DML handles any errors that are concerned with the communication between the running processes, by automatically reconnecting broken connections between ports.

1.7. Objective

This research is about a problem of the natural interaction between the robot and a human. It is a trial to solve the problem by explicitly modelling the robot's affective state, and then using this dynamic component to create a believable behaviour of the robot with the help of dynamic facial expressions during the interaction. This process can be achieved by the following two steps:

- Development of an emotion module (GEMS) for the dynamic modelling of the robot's affective state with the required inference methods for the current emotional state and the dominant emotion. The reasoning process appraises events, actions, and objects in the light of the robot's goals, standards and attitudes. The result will be used to provide a more natural conversation style and provide affective feedback by expressing the emotional state in its facial expression.
- Evaluation of the effectiveness of the robot's emotional models in achieving the research objective with explicit user scenarios.

This document provides detailed information about the features and possible usages of the GEMS module. The GEMS architecture, connection protocols, tuning functions and possible adjustments are presented. All information is presented from a technical point of view and is intended for developers who will continue research in this direction or will use this module for increasing the believability of the robot emotional behaviour with the ability of emotion-based reactions.

2. GEMS overview

General Emotion Management System (GEMS) was created as the software-based prototype of an emotion system for a robot that includes Artificial Intelligence (AI). The prototype is integrated in the overall software architecture of the robot. It can process multiple events as input and produce a multidimensional emotional state as output.

The GEMS module generates an emotional state of the robot each moment over time. The emotion generation algorithm is based on the work of O'Reilly [2] and on the OCC (Ortony, Clore & Collins) model [3].

As input the GEMS module receives information about the current robot state (*goals, actions, events, objects or agents*) from the superior robot system and detailed information from the database, which must contain all necessary variables to calculate the intensity value for each emotion.

The emotional state of the robot consists of 22 types of emotion in a multi goal environment. As output the GEMS module defines information about the *emotional state* in general and information about the *dominant emotion*. The output of the GEMS module is used for defining a facial expression on the mechanical face of the robot.

2.1. Design decisions

During the project the following design decisions were made.

1. The theory of the Em system was chosen as the basis of the GEMS module. There are several reasons for this choice. The Em system has its architecture based on the OCC model. But at the same time the Em system represents a mathematical formulation of the OCC model with a complete set of rules (or formulas) and variables, which are necessary for the calculation of the emotion's intensity. Each variable has a complete description including the list of the factors that influence this variable.
2. Values of the variables that are necessary to calculate the intensity of each emotion should be stored in a database. The database includes four tables with the variables which characterize goals, events, actions and objects (agents). The tables of the database are interconnected with each other and represent the structure of the robot's knowledge about its surroundings (the external world).
3. The emotional state of the robot is the result of the work of the GEMS module. The emotional state should contain the information about all current goals of the robot and all emotions with non-zero intensity that exist "inside" the robot at any particular moment. The intensity of the emotions decays in time. Also the emotional state should keep during some time the information about the recent emotions with the intensity decayed to zero. It is necessary to make the corrections in the intensity value of new emotions that have the same type and the same "reason" of appearance as an existing one with zero intensity value. For example, the robot should "feel" the emotion with type "Like" towards the red ball (when it was detected successfully) with a lower intensity value if the robot saw the red ball ten times during the last 20 minutes.

4. The current emotional state could be represented by the facial expression on the mechanical face of the robot. It could be done by defining a dominant emotion of the robot's emotional state, which could be mapped to the particular facial expression. The dominant emotion could be converted to the facial expression with the help of the mapping rules, which make a connection between the 22 types of emotion and the six facial expressions.

2.2. The robot's architecture and GEMS

GEMS has been created as an independent software module in the robot's architecture. The GEMS classes have been created using the Dynamic Module Library (DML). DML is a framework that provides dynamic communication between processes (modules) across UNIX and Windows platforms [4]. DML is built on top of the Parallel Virtual Machine (PVM) library version 3.4.4 [6]. This library provides the interaction between software modules across different operating systems and platforms (*fig. 2.1*).

The communication with other modules is done using ports, which is supported by DML.

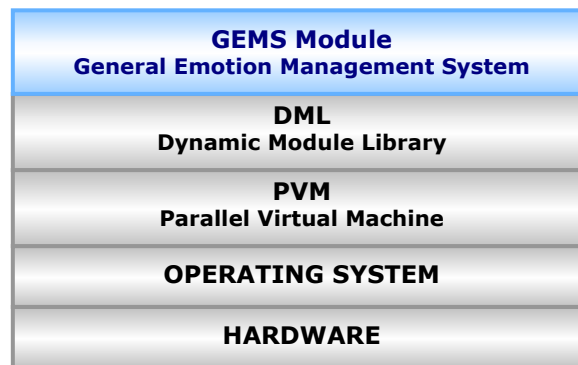


Figure 2.1 Position of the GEMS module in the overall robot architecture

2.3. GEMS architecture

Figure 2.2 shows the software architecture of the GEMS module and its connections to other modules. The connections are made using the ports. On the diagram the connections are shown as triangles connected by means of arrows.

The architecture is presented with the help of structure elements – blocks or units. The purpose of using this kind of elements is to show the logical structure of the GEMS module. The big white arrows show the data flows between the units.

The GEMS major components are: *Information unit, GEMS Database, OCC Engine, Initialisation unit, Mood unit and Output unit* (*fig. 2.2*).

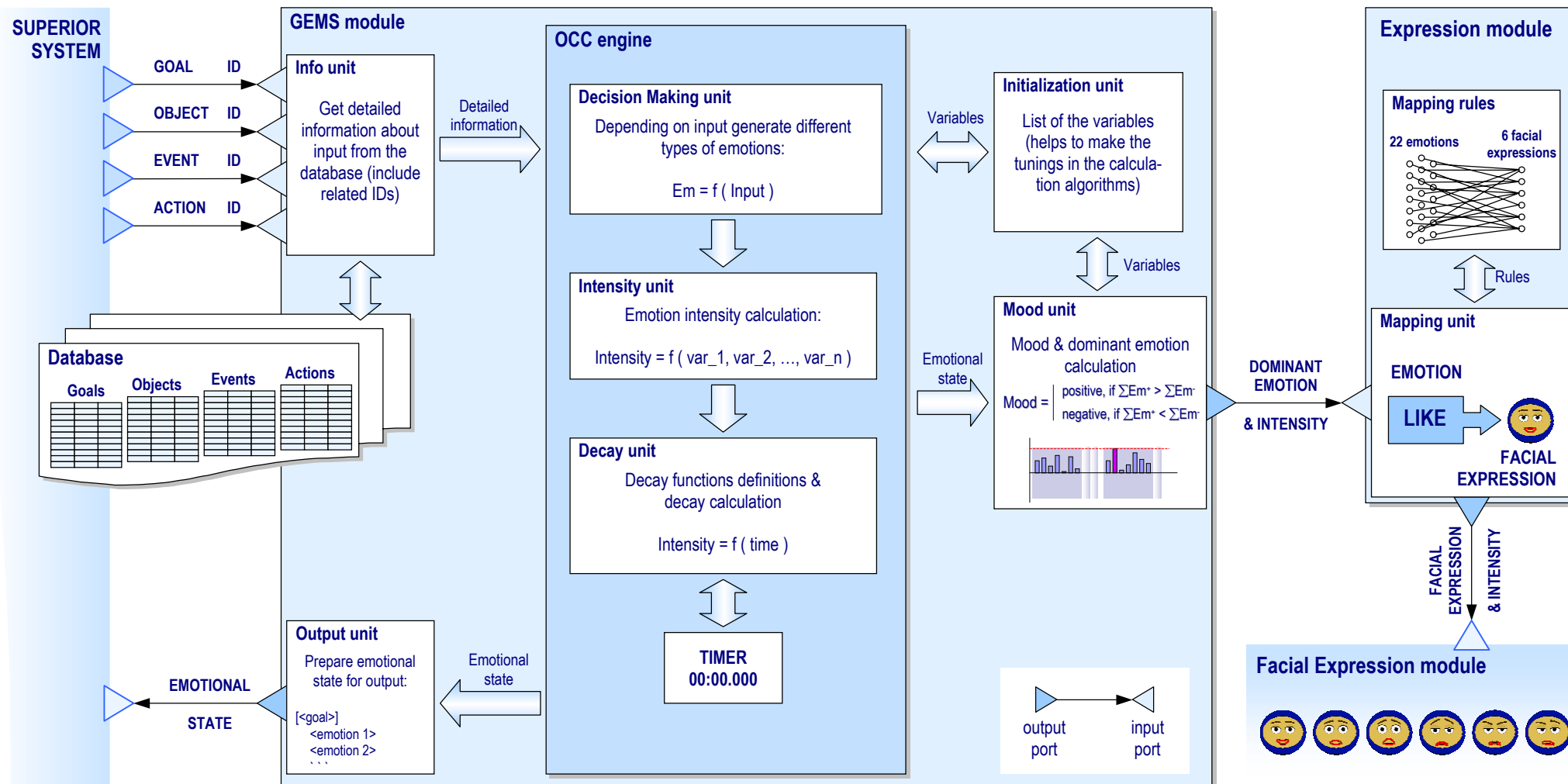


Figure 2.2 GEMS architecture

2.3.1. Input information

GEMS is assumed to be connected to some superior system of the robot. The Superior System is the system that manages all robot modules, and has knowledge about the surrounding environment – "the world model".

GEMS has four **input ports**:

```
InputPortBufferedInteger m_GemsGoal;           // input for Goals ID
InputPortBufferedInteger m_GemsAction;         // input for Actions ID
InputPortBufferedInteger m_GemsEvent;          // input for Events ID
InputPortBufferedInteger m_GemsObject;         // input for Objects ID

m_GemsObject("GemsObject", 1, NO_MAX_CONNECTIONS, Push_e, (t_EventHandler) vNewObject),
m_GemsEvent("GemsEvent", 1, NO_MAX_CONNECTIONS, Push_e, (t_EventHandler) vNewEvent),
m_GemsAction("GemsAction", 1, NO_MAX_CONNECTIONS, Push_e, (t_EventHandler) vNewAction),
m_GemsGoal("GemsGoal", 1, NO_MAX_CONNECTIONS, Push_e, (t_EventHandler) vNewGoal)
```

These ports receive information from the Superior System about current Goals, Objects (Agents), Events or Actions.

All ports have the same type: `InputPortBufferedInteger`. It means that the ports can receive a single integer value at once, that all values will be buffered, and that no values will be lost [4]. The integer value represents an ID number of the record from the GEMS Database where detailed information about the Goals, Objects (Agents), Events and Actions is stored. The buffered ports are necessary to prevent loss of information from the Superior System.

The **Information unit** receives information from the input ports and manages a connection to the GEMS Database. When some of the input ports receive an ID value, the Information unit tries to find a record with this ID in the corresponding table of the database. If the search is successful, the Information unit retrieves the data from the database record and sends these data to the OCC Engine. In cases when such ID doesn't exist the Information unit sends information to the console that a non-existent ID is received.

2.3.2. GEMS database

Having information about the current goals, actions, events and objects (agents) is important for the calculation of the emotional state of the robot. This information should be received from the Superior System of the robot.

The GEMS Database contains all necessary information for the OCC Engine [3] concerning the Goals, Objects (Agents), Events and Actions. The database includes four tables, and has the structure and the connections shown in *Figure 2.3*.

All connections between database tables have relationship type "One-To-Many" (*fig. 2.3*).

The database is represented by four structured plain text files (*Appendix B*). Fields in the records are separated by a carriage return (\r) and a new line (\n), also referred to as CR/LF on Windows platforms or by a new line (\n), also referred to as a linefeed (LF) on Unix platforms. The line with three minuses "---" can be used for visually separating records from each other and helps to read the data and enter input data into the database.

Goal table

Name	Format (range)	Comment
ID	int (1...∞)	id number
Name	char* (0...255 chars)	name
Hierarchy	int (0...∞)	hierarchy
DoS	int (0...100)	desirability of success
DnF	int (0...100)	desirability of not fail
LoS	float (0...1)	likelihood of success
LoF	float (0...1)	likelihood of fail
Agent	int (1...∞)	related agent id

Event table

Name	Format (range)	Comment
ID	int (1...∞)	id number
Name	char* (0...255 chars)	name
dLoS	float (-1...1)	change in goal LoS
dLoF	float (-1...1)	change in goal LoF
Goal	int (1...∞)	related goal id

Action table

Name	Format (range)	Comment
ID	int (1...∞)	id number
Name	char* (0...255 chars)	name
Priseworth	int (-100...100)	praiseworthiness
Agent	int (1...∞)	related agent id
Responsib	float (0...1)	responsibility of the agent
Event	int (0...∞)	related event id

Object (Agent) table

Name	Format (range)	Comment
ID	int (1...∞)	id number
Name	char* (0...255 chars)	name
Type	int (0 1)	type Object Agent
Appeal	int (-100...100)	appealingness
Goal	int (0...∞)	related goal id

Figure 2.3 GEMS Database tables and connections

It is possible to change data in the database when the GEMS module is running (it is protected by a semaphore). The Information unit reads a data item from the database each time when some input value is received.

Goal Information

The point of the goal is the execution of some action or the achievement of some particular state of the world. The goal has the following characteristics: *desirability*, *likelihood*, *hierarchy* and *agent* (who has this goal).

In the Goal table desirability of goals is broken down into two parts: importance that the goal succeeds (DoS) and importance that the goal does not fail (DnF). For example, "imagine Jake hates being late for appointments; being on time doesn't make him happy, but being late does make him upset." [2]. This can be modelled by creating Jake's goal to be on time a low (or 0) DoS and a medium-high DnF.

The likelihood of a Goal is also divided into two parts: the likelihood of the goal succeeding (LoS) and the likelihood of the goal failing (LoF). These values are independent from each other and show how likely this particular goal is to succeed and fail. This was made to add flexibility to the module and to distinguish between calculations of positive and negative emotions [2].

It is up to the robot developer how to manipulate these two variables: to use them independently or to use the rule $LoS + LoF = 1$. The developer himself should take care of that.

Success means the goal has changed to a likelihood of succeeding (LoS) of 1. Failure means the goal has changed to a likelihood of failing (LoF) of 1.

The field *Hierarchy* defines a value of the internal hierarchy of this goal inside the goal structure of the Superior System.

The field *Agent* contains the ID value from the Object (Agent) table and defines the Agent who has this goal.

Object (Agent) Information

Ortony, Clore and Collins explain objects as follows: "Objects are objects viewed qua objects" [3]. The Agent represents an Object, which can perform some *actions*.

The field *Type* defines distinguish between an Agent and an Object. A value of 1 means Agent, 0 means Object.

The robot itself is represented in the table as the Agent with the ID = 1.

The field *Appeal* shows the attraction of the robot to this Object (Agent).

The field *Goal* was added to make the information structure more flexible. It gives the possibility to define different attraction levels of the robot for the same object but "inside" different goals.

Event Information

The conception of events is very straightforward – events are simply people's interpretations about things that happen, considered independently of any beliefs they may have about actual or possible causes [3].

Each event has a relation to some goal, defined in the field *Goal*.

When some event has happened, the likelihood of the goal succeeding and the likelihood of the goal failing are changed. The changes in either likelihood value are stored in the fields *dLoS* and *dLoF*.

Action Information

Actions are the "steps" that an agent performs to achieve current goals.

Action characteristic elements:

Agent – agent ID who is responsible for this action.

Responsib – level of responsibility of the agent.

Event – event ID that is a result of this action.

Praiseworth – praiseworthiness of this action according to some standard of behaviour. Standards can be of moral quality: thou shall not kill. Or they can be of performance quality: I should be able to do well in school [2]. The level of the value represents the strength of the standard.

2.3.3. OCC Engine

The OCC Engine is a group of units that create an emotional state of the robot at each moment in time.

The **Decision Making** unit utilises the model from Scott Neal Reilly's thesis [2] and the OCC model [3] to create emotions depending on input received from the Information unit. *Table 2.1* presents the whole list of the Cognitive-Appraisal Emotion types with their description. *Figures 2.4, 2.5 and 2.6* show the algorithms.

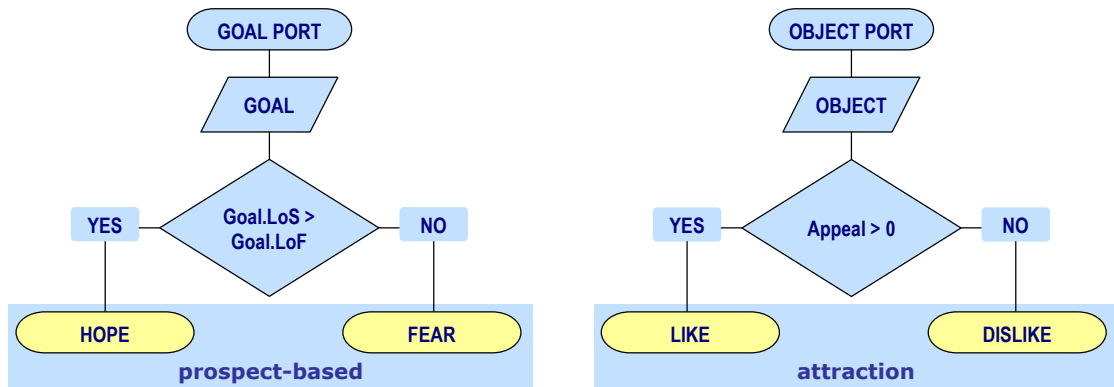


Figure 2.4 Algorithm processing the data from Goal & Object port

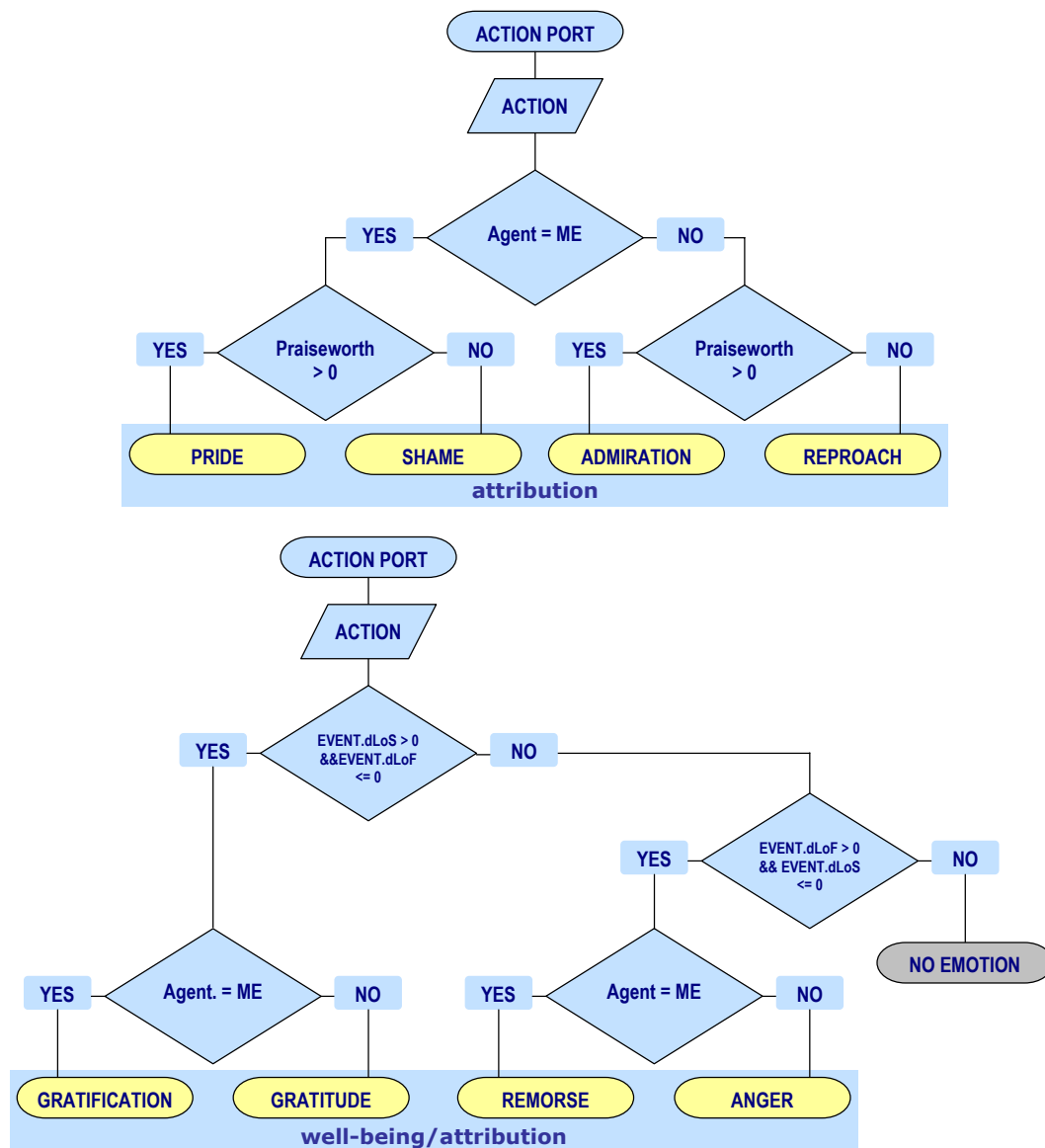


Figure 2.5 Algorithm processing the data from Action port

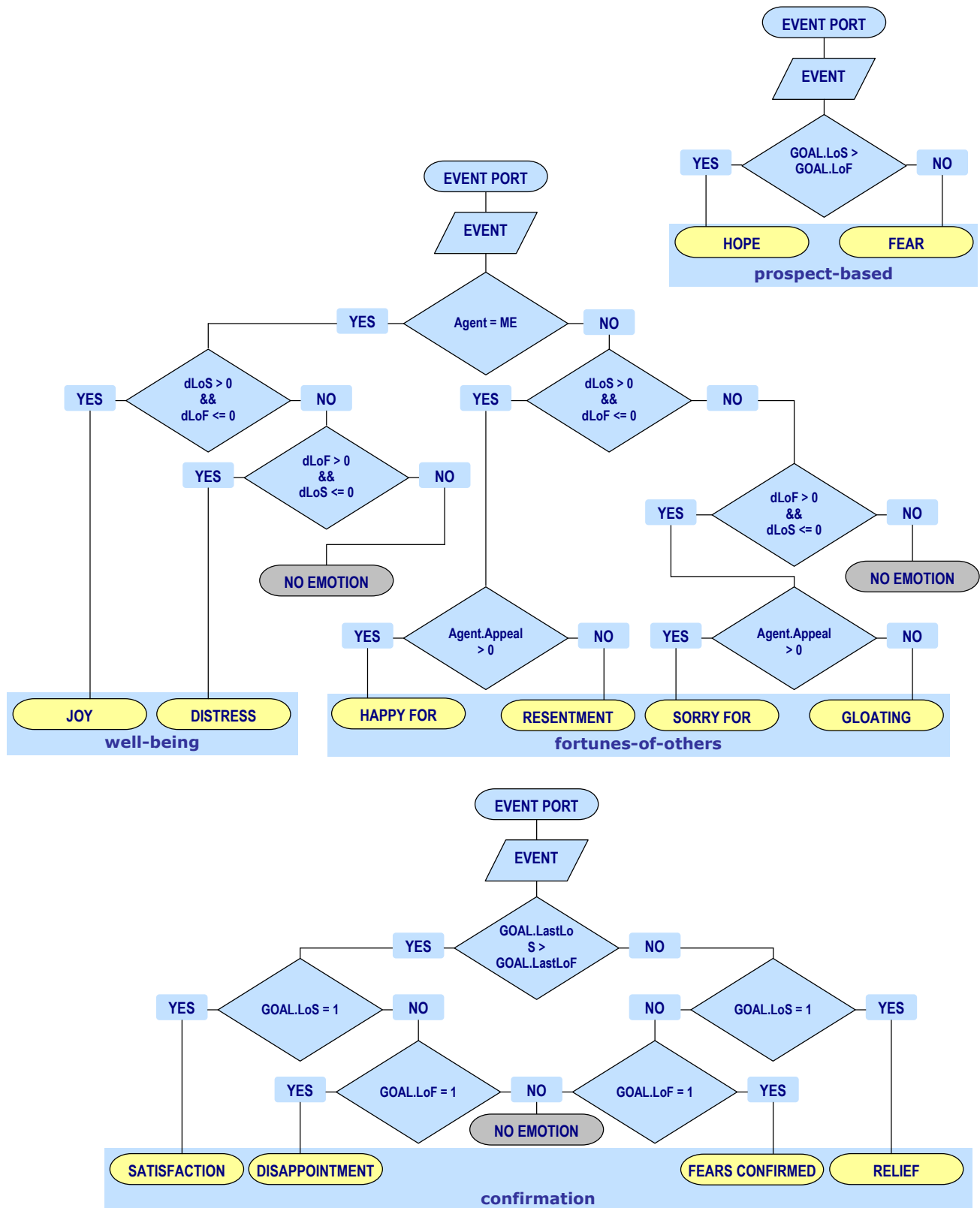


Figure 2.6 Algorithm processing the data from Event port

Table 2.1 Emotion categories and intensity calculation (based on Scott Neal Reilly [1])

GROUP	GROUP SPECIFICATION	EMOTION: DESCRIPTION	INTENSITY (CALCULATION RULES)
Well-Being	appraisal of a situation as an event	joy : pleased about a desirable event	if (Event.dLoS > 0 && Event.dLoF <= 0) Int = Event.dLoS * Goal.DoS
		distress : displeased about an undesirable event	if (Event.dLoF > 0 && Event.dLoS <= 0) Int = Event.dLoF * Goal.DnF
Prospect-based	appraisal of a situation as a prospective future event	hope : pleased about a prospective desirable event*	if (Goal.LoS > Goal.LoF) Int = Goal.LoS * Goal.DoS
		fear : displeased about a prospective undesirable event*	if (Goal.LoS < Goal.LoF) Int = Goal.LoF * Goal.DnF
Confirmation	appraisal of a situation as confirming or disconfirming an expectation (present event)	satisfaction : pleased about a confirmed desirable event	if (LastLoS > LastLoF && Goal.LoS == 1) Int = LastLoS * Goal.DoS
		fears-confirmed : displeased about a confirmed undesirable event	if (LastLoS < LastLoF && Goal.LoF == 1) Int = LastLoF * Goal.DnF
		relief : pleased about a disconfirmed undesirable event	if (LastLoS < LastLoF && Goal.LoS == 1) Int = LastLoF * Goal.DnF
		disappointment : displeased about a disconfirmed desirable event	if (LastLoS > LastLoF && Goal.LoF == 1) Int = LastLoS * Goal.DoS
Fortunes-of- Others	presumed value of a situation as an event affecting another agent	happy-for : pleased about an event desirable for another agent	if (Agent.Appeal > 0) Int = abs(Agent.Appeal) * Event.dLoS * Goal.DoS
		sorry-for(pity) : displeased about an event undesirable for another agent	if (Agent.Appeal > 0) Int = abs(Agent.Appeal) * Event.dLoF * Goal.DnF
		gloating : pleased about an event undesirable for another agent	if (Agent.Appeal < 0) Int = abs(Agent.Appeal) * Event.dLoF * Goal.DnF
		resentment : displeased about an event desirable for another agent	if (Agent.Appeal < 0) Int = abs(Agent.Appeal) * Event.dLoS * Goal.DoS
Attribution	appraisal of a situation as an accountable action of some agent	pride : approving of one's own (me) praiseworthy action	if (Agent == ME && Action.Priseworth > 0) Int = Action.Priseworth
		shame : disapproving of one's own (me) blameworthy action	if (Agent == ME && Action.Priseworth < 0) Int = Action.Priseworth
		admiration : approving of another's (agent) praiseworthy action	if (Agent != ME && Action.Priseworth > 0) Int = Action.Priseworth
		reproach : disapproving of another's (agent) blameworthy action	if (Agent != ME && Action.Priseworth < 0) Int = Action.Priseworth
Well-being/ Attribution	compound emotions	gratitude : (admiration + joy) approving of another's (agent) praiseworthy action and pleased about the related desirable event	if (Agent != ME && Event.dLoS > 0 && Event.dLoF <= 0) Int = Goal.DoS * Action.Responsib
		anger : (reproach + distress) disapproving of another's (agent) blameworthy action and displeased about the related undesirable event	if (Agent != ME && Event.dLoF > 0 && Event.dLoS <= 0) Int = Goal.DnF * Action.Responsib
		gratification : (pride + joy) approving of one's own (me) praiseworthy action and pleased about the related desirable event	if (Agent == ME && Event.dLoS > 0 && Event.dLoF <= 0) Int = Goal.DoS * Action.Responsib
		remorse : (shame + distress) disapproving of one's own (me) blameworthy action and displeased about the related undesirable event	if (Agent == ME && Event.dLoF > 0 && Event.dLoS <= 0) Int = Goal.DnF * Action.Responsib
Attraction	appraisal of a situation as containing an attractive or unattractive object	liking : liking an appealing object	if (Object.Appeal > 0) Int = Object.Appeal
		disliking : disliking an unappealing object	if (Object.Appeal < 0) Int = Object.Appeal

* The initial values for the emotions Hope and Fear also could be calculated when GEMS receives information about the Goal.

When some emotion is created the Decision Making unit saves the time when this emotion was born and starts the **Timer**. Therefore, the Timer indicates the age of each emotion.

The **Intensity unit** calculates the intensity values for each emotion created in the Decision Making unit.

The last column of *Table 2.1* presents the conditions and rules (formulas) to calculate the intensity for each type of emotion. The formulas are based on the work of O'Reily [2]. Using these formulas, the OCC Engine calculates the initial intensity value for each new emotion.

The **Decay unit** connects the decay function to each emotion and changes the intensity value depending on the age of the emotion (Timer value).

By default, the type of the decay function for a new emotion is taken from the INI file, except for the Hope and Fear types of emotion. Hope and Fear stay constant during a goal, and start to decay after the goal is finished (this decay function type is faster than for other types of emotion and is also defined in the INI file).

It is possible to choose several types of decay functions. They can be divided into three types: *constant*, *linear* and *non-linear*.

Constant and linear functions are very simple:

$$\begin{aligned} \text{d_Const} \quad & \text{Int} = I \\ \text{d_Linear} \quad & \text{Int} = I - \frac{t * 100}{s} \end{aligned}$$

where: Int – current emotion intensity ($\text{Int} \in 0...100$),
 I – initial emotion intensity ($I \in 0...100$),
 t – age of the emotion in seconds ($t \geq 0$),
 s – max decay time in seconds ($s > 0$).

The "max decay" time determines the angle α of the linear decay function (*fig. 2.7*). The angle α determines the speed of decay; a bigger angle value indicates slower decay.

The non-linear decay functions can also be divided in two groups: *parabola* and *inverse parabola*. The difference between these two types is a change in the speed of decay. In the parabola case the speed of decay is low at the start and increasing during the time. In the inverse parabola case – the speed of decay is high at the start and decreasing during the time (*fig. 2.8*).

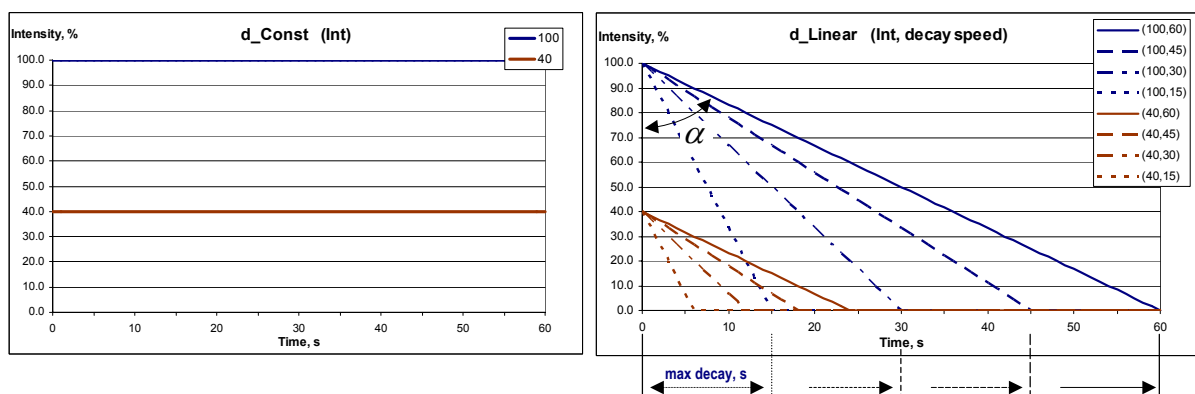


Figure 2.7 Constant and linear decay functions

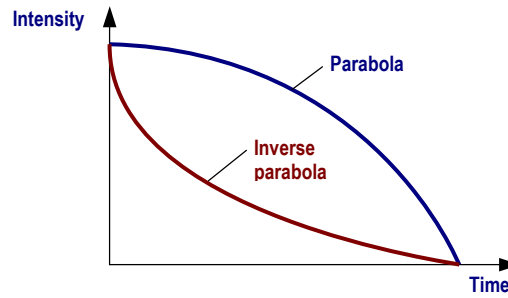


Figure 2.8 Parabola & inverse parabola decay functions

Also the non-linear functions could have different behaviour if the initial intensity of the emotion does not have the maximal value. Figure 2.9 shows that there exist three ways to define a non-linear decay function for a non-maximal intensity.

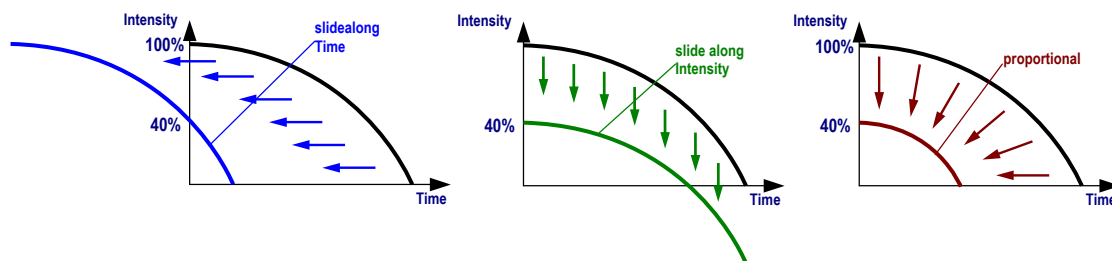


Figure 2.9 Different definitions of decay functions with non-max intensity

The *first* one is to slide the function graphically along the Time coordinate axis. It means that the speed of decay will be depending on the intensity value.

The *second* one is to slide the function graphically along the Intensity coordinate axis. It means that the speed of decay will be depending on the time value.

And the *third* one is to scale function graphically along both coordinate axis. It means that the speed of decay will be proportional to the Intensity value in comparison to the maximum intensity.

Therefore, we can define the following six non-linear decay functions (fig. 2.10):

$$\text{d_pParabol} \quad \text{Int} = I - \sqrt{I * \frac{t}{s} * 10}$$

$$\text{d_iParabol} \quad \text{Int} = I - \sqrt{t * \frac{10}{s}}$$

$$\text{d_tParabol} \quad \text{Int} = 100 - \sqrt{\frac{t}{s} + \frac{(100 - I)^2 * s}{10000}} * 100$$

$$\text{d_pParabolInv} \quad \text{Int} = I - \frac{t^2 * 10000}{I * s^2}$$

$$\text{d_iParabolInv} \quad \text{Int} = I - \frac{t^2 * 100}{s^2}$$

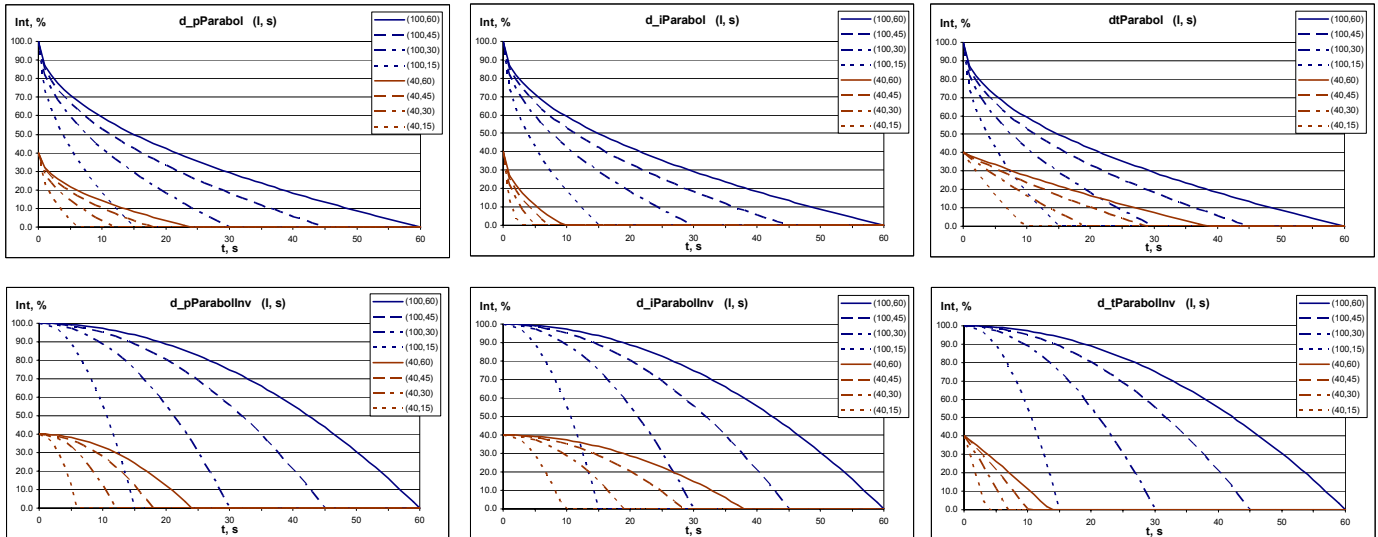


Figure 2.10 Six nonlinear decay functions

$$d_tParabolInv \quad Int = 100 * \left(1 - \sqrt{\frac{t}{s} + \frac{(100 - I)^2 * s}{10000}} \right)$$

where Int – current emotion intensity ($Int \in 0...100$),
 I – initial emotion intensity ($I \in 0...100$),
 t – age of the emotion in seconds ($t \geq 0$),
 s – max decay time in seconds ($s > 0$).

During this project only two decay functions have been used (constant and linear). The rest have been defined and implemented for usage in further research.

As a result the OCC Engine produces the *emotional state* of the robot. The emotional state of the robot consists of 22 types of emotion in a multi goal environment with various intensities (fig. 2.11).

Each emotion carries the following information:

- Reason – reason of appearing this Emotion (an ID of related Object/Event/Action) (Table 2.2)
- Goal – goal ID from the GEMS Database
- EmType – type of the emotion
- EmDecay – type of the decay function
- DecaySpeed – max decay time, the time in seconds necessary to decay from 100% intensity to 0%. That value helps to manage the speed of decaying
- Intensity – value of initial emotion intensity
- CurIntensity – current emotion intensity
- StartTime – the time when this emotion was born
- DecayTime – the time when this emotion start decaying.

2.3.4. Output information

The GEMS supports two kinds of outputs: the *emotional state* of the robot and the *dominant emotion* each moment over time.

The **Output unit** represents the emotional state as a formatted string to prepare it for output through the port connected to the Superior System:

```
OutputPortLastString m_GemsOutEmState; // output emotional state
```

The port has the type: `OutputPortLastString` and is intended to send string values. The emotional state has been converted to the formatted string with the following view:

```
[<goal_1 ID>]
  <emotion_1 type> <intensity> <reason>
  <emotion_2 type> <intensity> <reason>
  ...
[<goal_2 ID>]
  ...
```

Where: `<goal_i ID>` is a goal ID from the GEMS Database (integer $1 \dots \infty$)

Table 2.2 The types of the emotions and the reason values

Group	Emotion	Type	Reason
Attraction	Liking	1	Object (Agent)
	Disliking	2*	
Well-being	Joy	5*	Event
	Distress	6	
Prospect-based	Hope	7	
	Fear	8	
Confirmation	Satisfaction	9	
	Fears-confirmed	10	
	Relief	11	
	Disappointment	12	
Fortunes-of-others	Happy-for	13	
	Sorry-for	14	
	Gloating	15	
	Resentment	16	
Attribution	Pride	17	Action
	Shame	18	
	Admiration	19	
	Reproach	20	
Well-being / Attribution	Gratitude	21	
	Anger	22	
	Gratification	23	
	Remorse	24	

*Two emotions (Love and Hate) with the types 3 and 4 are missing. They are analogous to Like and Dislike and were defined for future use.

<emotion_i type> – emotion type (integer 1...24 as shown Table 2.2)

<intensity> – intensity value, % (integer 0...100)

<reason> – reason of appearing this Emotion (an ID of related Object/Event/Action as shown Table 2.2) (integer 1...∞)

For example,

```
[32]
 7 56 12
 5 17 12
 1 43 142
17 22 27
[7]
 8 68 45
 6 34 45
...
```

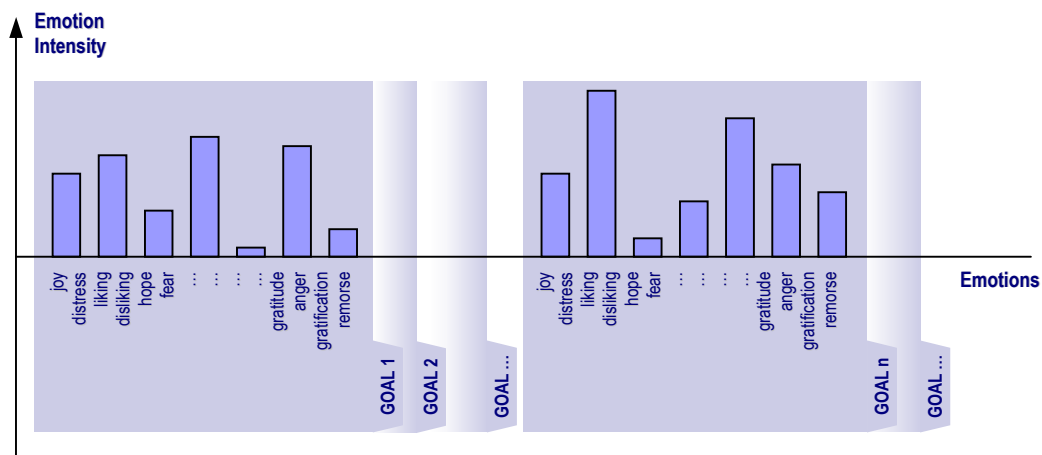


Figure 2.11 Emotional state of the robot

In this way the Superior System could have at all time the information about the current emotional state of the robot.

The **Mood unit** analyses the emotional state of the robot, defining a Mood value and a dominant emotion each moment over time.

The *mood* of the robot is a ratio between the sum of intensities for positive and negative emotions [26](2.1).

$$MOOD = \frac{\sum Intensity(Em+)}{\sum Intensity(Em-)} \quad (2.1)$$

Where: $Em +$ is a positive emotion

$Em -$ – negative emotion (Table 1.1).

If the Mood has a value under 1, the Mood to be considered as positive. If the value below 1, the Mood to be considered as negative.

The Mood unit calculates the *dominant emotion* and presents information as to integer values to prepare it for output through the port connected to the Expression Module:

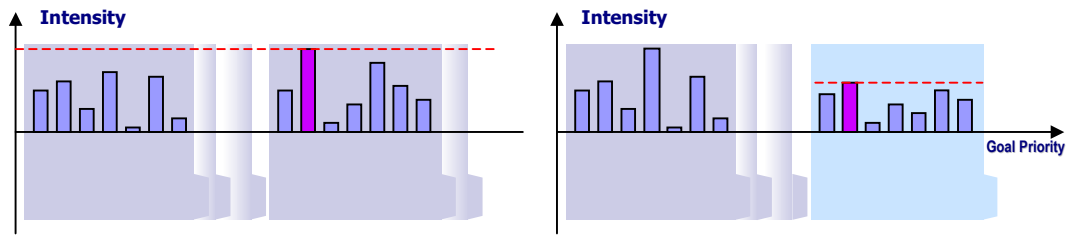


Figure 2.12 Dominant emotion: goal priority

```
OutputPortLastInteger m_GemsOutDomEm; // output dominant emotion
```

The port has the type: `OutputPortLastInteger` and is intended to send two integer values:

<emotion type> <intensity>

Where: <emotion type> – emotion type (integer 1...24 as shown Table 2.2)

<intensity> – intensity value, % (integer 0...100).

There are two possible ways to select the dominant emotion: to use the all existing emotions in the emotional state or to use only the emotions "inside" the goal with the highest priority (global variable `_domGoal_`) (fig. 2.12).

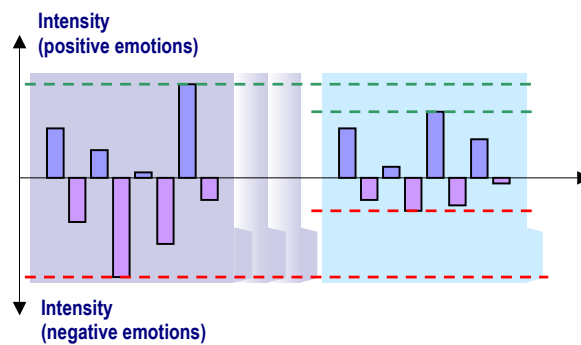


Figure 2.13 Dominant emotion: current mood

Also it is possible to take into account the current value of the Mood. The Mood could be positive or negative. The dominant emotion could be selected among positive or negative emotions according to the current value of the Mood (global variable `_domMood_`) (fig. 2.13).

The *Dominant emotion* being chosen depends on the "normalized" intensity of emotions. "Normalization" is necessary to balance the weight of different types of emotions. It gives the possibility to compare emotions among each other by "normalized" intensity values.

The GEMS module gives the possibility to manipulate with the two global variables (`_priorMapCoeff_` and `_priorLifeTime_`) what influence "normalization" of the dominant emotion has.

The first variable (`_priorMapCoeff_`) allows to make a choice what to compare: intensities of the emotions or intensities of the facial expressions. In the first case, original intensities of the emotions participate in the comparison. In the second the original intensities are converted to the appropriate facial expressions using the "Mapping rules" of converting emotions to facial expressions (Formula (2.2)).

$$Int = I * MapCoeff \quad (2.2)$$

where Int – "normalized" emotion intensity ($Int \in 0...100$),
 I – original emotion intensity ($I \in 0...100$),
 $MapCoeff$ – mapping coefficient from the emotion to the facial expression (fig. 2.15).

The second variable ($_priorLifeTime_$) is making the connection between the intensity and the lifetime of the emotion. The original intensity of the emotions is recalculated with the following formula (2.3).

$$Int = I + (100 - I) * \left(\frac{1 - EmLifeTime}{S * K} \right) \quad (2.3)$$

where Int – "normalized" emotion intensity ($Int \in 0...100$),
 I – original emotion intensity ($I \in 0...100$),
 $EmLifeTime$ – age of the emotion in milliseconds ($EmLifeTime > 0$),
 S – time of the normalization in milliseconds ($S \in 1000...10\ 000$),
 K – divisor ($K \in 1...5$).

It is giving the possibility for the emotions with a low intensity to win the dominance at the first seconds of its life (fig. 2.14). In that way the robot becomes more expressive and has visible reactions on the events that produce emotions with the low intensity.

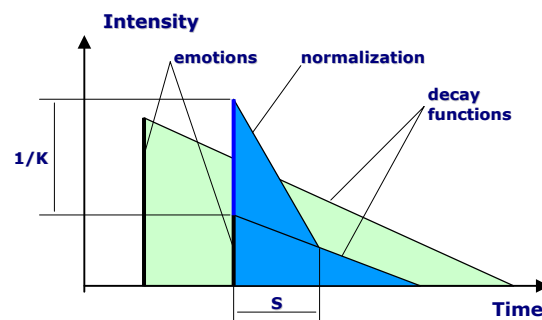


Figure 2.14 Emotion "normalization"

Table 2.3 shows all possible combinations for the global variables $_priorMapCoeff_$ and $_priorLifeTime_$.

Table 2.3 Dominant emotion calculation method

Global variables		Used formulas	
$_priorMapCoeff_$	$_priorLifeTime_$	(2.2)	(2.3)
0	0		
1	0	✓	
0	1		✓
1	1	✓	✓

2.4. The INI file content

The GEMS module reads global constants and variables from Gems.INI file each time it is started. Each constant or variable has a default value, which will be taken if the module will be unable to read some value. The detailed descriptions for the constants and variables are presented in Appendix A.

Table 2.4 Mapping rules

Emotion	Facial expressions	Expressions in the mix, %	Intensity, %
liking	happiness surprise	70 30	100
disliking	disgust happiness	100 0	100
joy	happiness surprise	100 0	100
distress	sadness anger	100 0	100
hope	happiness surprise	100 0	20
fear	fear sadness	50 50	20
satisfaction	happiness surprise	100 0	80
fears-confirmed	fear sadness	50 50	80
relief	happiness surprise	70 30	70
disappointment	fear sadness	50 50	80
happy-for	happiness surprise	100 0	100
sorry-for (pity)	fear sadness	50 50	100
gloating	disgust happiness	50 50	100
resentment	anger disgust	50 50	100
pride	happiness surprise	100 0	100
shame	sadness anger	50 50	100
admiration	happiness surprise	50 50	100
reproach	anger disgust	50 50	100
gratitude	happiness surprise	100 0	100
anger	anger disgust	100 0	100
gratification	happiness surprise	100 0	100
remorse	sadness anger	100 0	100

2.5. Expression module

The Expression module is necessary to evaluate the results of the work of the GEMS module.

The module receives the information about the dominant emotion from the GEMS module and makes a mapping from this emotion to the facial expression.

There is very small amount of information about this type of mapping presented in the literature. A further research and a serious user testing are necessary to receive such kind of mapping. The general description of the possible user tests was defined.

Meanwhile, the following mapping data was generated using my own opinion and opinions of my colleagues. The rules of the mapping are shown on *Figure 2.15*. Each dot represents the particular emotion mapped to the particular place on the facial expression disk that means the particular parameters of the facial expression (*Figure 1.7*).

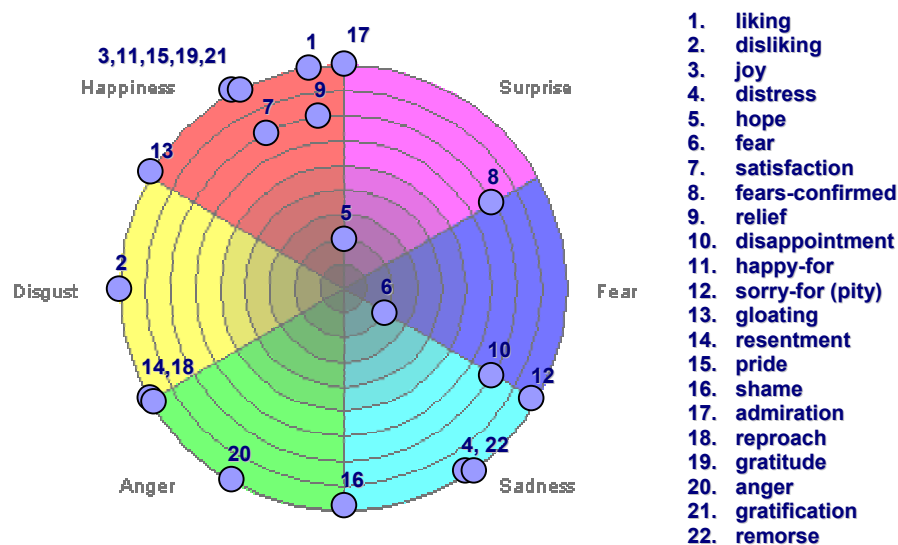


Figure 2.15 Mapping 22 Emotions to the Facial Expressions

The connection information for each emotion includes the names of the two facial expressions in the mix, the percentage of the first facial expression in the mix and the facial expression intensity value which is correspond to the 100% intensity of the emotion (*Table 2.4*).

2.6. Compatibility

The GEMS module has been created under Windows 2000 using Microsoft Visual C++, without using any of the Microsoft Foundation Classes (MFC).

The GEMS module fits into the overall robot architecture in designing compiler- and platform-independent code. But unfortunately it was possible to test the GEMS only under *Windows 2000*.

The source code of the GEMS module has been compiled with *Microsoft Visual C++* (version 6.0) using the of the DML library [4].

3. User testing

3.1. Scenario-based user testing

In the field of usability engineering, scenarios depict system usage in the form of possible or actual action and event sequences. They describe how system components, environments and users work concurrently and interact in order to generate system level functionality. Therefore, the system designers can detect problems and come up with capable solutions based on the developed scenarios.

Focus groups are a somewhat informal technique that can help developers to assess user needs and feelings both before interface design and long after implementation. In a focus group, the moderator brings together from six to nine users to discuss issues and concerns about the features of a user interface.

Focus groups often bring out users' spontaneous reactions and ideas and let the moderator observe some group dynamics and organizational issues. The moderator can also ask people to discuss how they perform activities that span many days or weeks: something that is expensive to observe directly.

The user scenarios are used to develop the templates that could show how the users could interact with the robot. Three demo scenarios were developed: 1) the robot is searching for a red ball in the room; 2) an example of emotional conversation and 3) an interaction between the robot and a human at home. The last two scenarios were used for the further user testing.

3.2. "Search of the red ball"

This is a very simple demo scenario with one goal. It was developed for the pilot test of the GEMS module output. In this scenario the robot has only one goal: to search the red ball (*Appendix C A.9*). When the robot receives the task to find the ball he has to turn into the four sides of a room and to search the red ball.

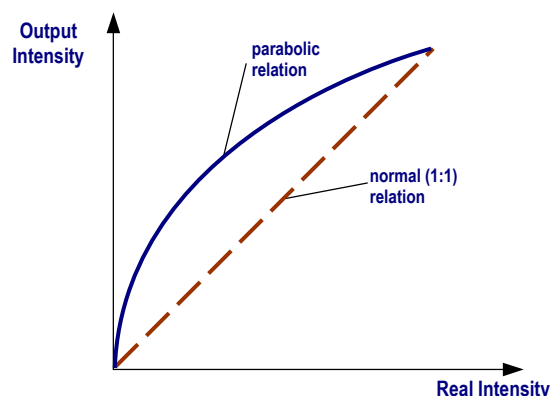


Figure 3.1 The parabolic relation between the real and the output intensity of the facial expression

The scenario was implemented with two possible results. The first case is the robot didn't find the red ball, the goal fails. The second case is the robot found the red ball, the goal succeeds.

During this pilot test the participants noticed that the level of expressiveness was not enough for a good perception and recognition of the robot's facial expression.

It turned out that the people perceive the intensity of the facial expression lower than its real value. After that the description of a further user test was defined [28]. This user test concerns with the perception and cognition by people of the robot's six facial expressions with the different intensities. Unfortunately, due to the time constraints it was impossible to run this user test.

The following method was proposed and used as a temporary solution to correct the intensity of the facial expressions of the robot. The method used a parabolic function to define the dependency between the real value of the facial expression intensity (generated by the OCC engine) and the intensity that will be used for the output on the robot's face (*fig. 3.1*).

3.3. "Emotional conversation"

This is the demo scenario of a dialog between the robot and a human (*Appendix C A.10*).

During the demo the robot has multiple goals and shows different facial expressions. For example, the robot has the following goals: to recognize the human's speech, to recognize physical objects (e.g. banana or lemon), to receive a prize, to defend yourself, to keep battery charged, etc. During the demo some of the goals succeed, some fail. In some cases the goal becomes more likely to fail and then succeeds (for other goals it was the other way around).

During the work on this scenario the best way for choosing the dominant emotion was defined. The best result shows the following set of the variables:

`_Tick_ = 500`: loop (or tick) time in milliseconds.

`_priorLifeTime_ = 1`: to take into account the emotion Life Time using the formula (2.2). The best results were received with the following values of the coefficients: $S = 10000$, $K = 2$.

`_priorMapCoeff_ = 1`: to take into account the mapping coefficient from the emotions to the Facial Expression disk.

`_domGoal_ = 0`: how to find the dominant emotion in the multigoal space – to use the complete set of the emotions from all goals.

`_domMood_ = 0`: how to find the dominant emotion with the current Mood – do not take into account the current value of the mood.

The other variables have default values (*Appendix A*).

The video demonstration was recorded using this scenario.

3.4. "Home interaction"

This is the demo scenario about a simple interaction between the robot and its owner at home (*Appendix C A.11*). In the first version of this scenario all goals of the robot succeed and

"everything is going well". In that case the robot presents only positive reactions (happiness – smile) with different levels of intensity. This type of user scenario was defined as "inexpressive" from the emotional point of view and could be used only for showing the functionality of the robot.

In the emotional "expressive" scenario the robot has to show different types of facial expressions (emotions). It should show not only positive but also different negative expressions. It could be done when in the scenario some of the goals fail or become more likely to fail first but then succeed. During this demo the robot can't reach success in several goals (e.g. to find the TV, to switch on the TV, to wake up Homer, etc.)

The video demonstration was recorded using this scenario.

3.5. User test procedure

For the user testing the following procedure was used. The participants were invited in groups with 4-10 people. The two demo scenarios were demonstrated to the participants. Then the moderator asked the participants to fill the questionnaire (*Appendix D*). After that the moderator initiated the discussion inside the group of the participants about demo scenarios. During the focus group session the moderator collected the participant's reactions, ideas and advices about the possible ways to improve the current scenarios and create more natural conversation between the robot and a human.

3.6. Results of the demo scenario evaluation

For the demo evaluation 20 people watched two demonstrations according to the last two scenarios (18 males and 2 females, from 22 till 45 years old). The participants have different backgrounds (in psychology and technical areas). They are all researchers in User Interfaces field.

Twelve participants filled out the questionnaire after the demonstration (*Appendix D*). After the demonstrations and focus group discussions the following results were defined.

1. The general impressions of the observers about the demos were very positive. People used following words to express their attitudes: "very nice", "funny", "amazing", "cool". Almost all highlighted that the dialogs (scenarios) were appealing and fun but one person made a note that the dialogs were over polite and too long.
2. The observers recognized the facial expression of the robot quite well. Only three people sometimes experienced problems with the recognition. All participants listed basic facial expressions they saw but one of them mentioned that the content of the dialogs helped to guess about the robot's emotions. Another participant added that the human recognition of the robot's facial expressions should be studied differently.
3. The spectators found the robot's reactions adequate. From the reasons "Why?", they mentioned that the reactions were "quite human like" and "looked quite natural" Some people noted delays in the robot's reactions and found the robot's anger in the beginning of the home interaction scenario too extreme.
4. Participants emphasized that the robot had a cartoon like way to express its emotions. Half of the observers realized that the robot could have an internal emotional state

and half perceived the robot's emotions as simple reactions, which could be pre-programmed for the demonstrations.

5. Almost all spectators rated the level of the expressiveness of the robot as "high". They mentioned that on the one hand the robot was perceived like more open, transparent in making decisions and very funny, on the other hand the emotions looked extreme and probably annoying.
6. The believability of the robots behaviour was rated as average. The observers had doubts about the robot's ability to express emotions, they thought that the behaviour of the robot was pre-programmed and "played". The facial expressions of the robot changed with the same speed in every situation, which decreased the believability. To increase the believability the participants proposed to decrease the robot's politeness, add speech intonations, body movements and nodding. Sudden changes of the facial expressions should be avoided.
7. To improve the scenarios the observers proposed to change the dialogs to make them more realistic and short. Also it would be nice to combine this demonstration with the real applications. If the robot can talk with the audience and/or describe what's happening in the room the demonstrations would be more realistic. Also it would be very useful to involve the observers in the interaction process.
8. Some of the spectators noticed that the colour of the robot was inappropriate and also that its gender was not defined but the voice was female.

4. Options for future research

The possible improvements for the GEMS module concern self-learning functions and a personality. *Figure 3.1* shows the GEMS architecture with the two additional units (blocks with the yellow background): *memory* and *personality*.

The **memory unit** – maintains past events (history), executes statistical analysis of the past events and makes changes in the GEMS database.

This module should represent an important aspect of artificial intelligence – self-learning functions. The self-learning functions are necessary for creating that part of the artificial intelligence of the robot that can evaluate current events in the light of the knowledge about the same events happened in the past. In other words the robot could change his appraisal of the events during his life.

The memory unit can represent the part of the past experience of the robot from the emotional point of view.

The **personality unit** – contains information about individual characteristics of the robot and knowledge how to change the personality over the time.

Have you ever met two persons with the same behaviour? People are very different from each other. Each person has its own individual character, own behaviour and temperament. There are four temperaments, accompanied by different degrees of strength and activity in the brain: Choleric, Melancholic, Sanguine, and Phlegmatic [16].

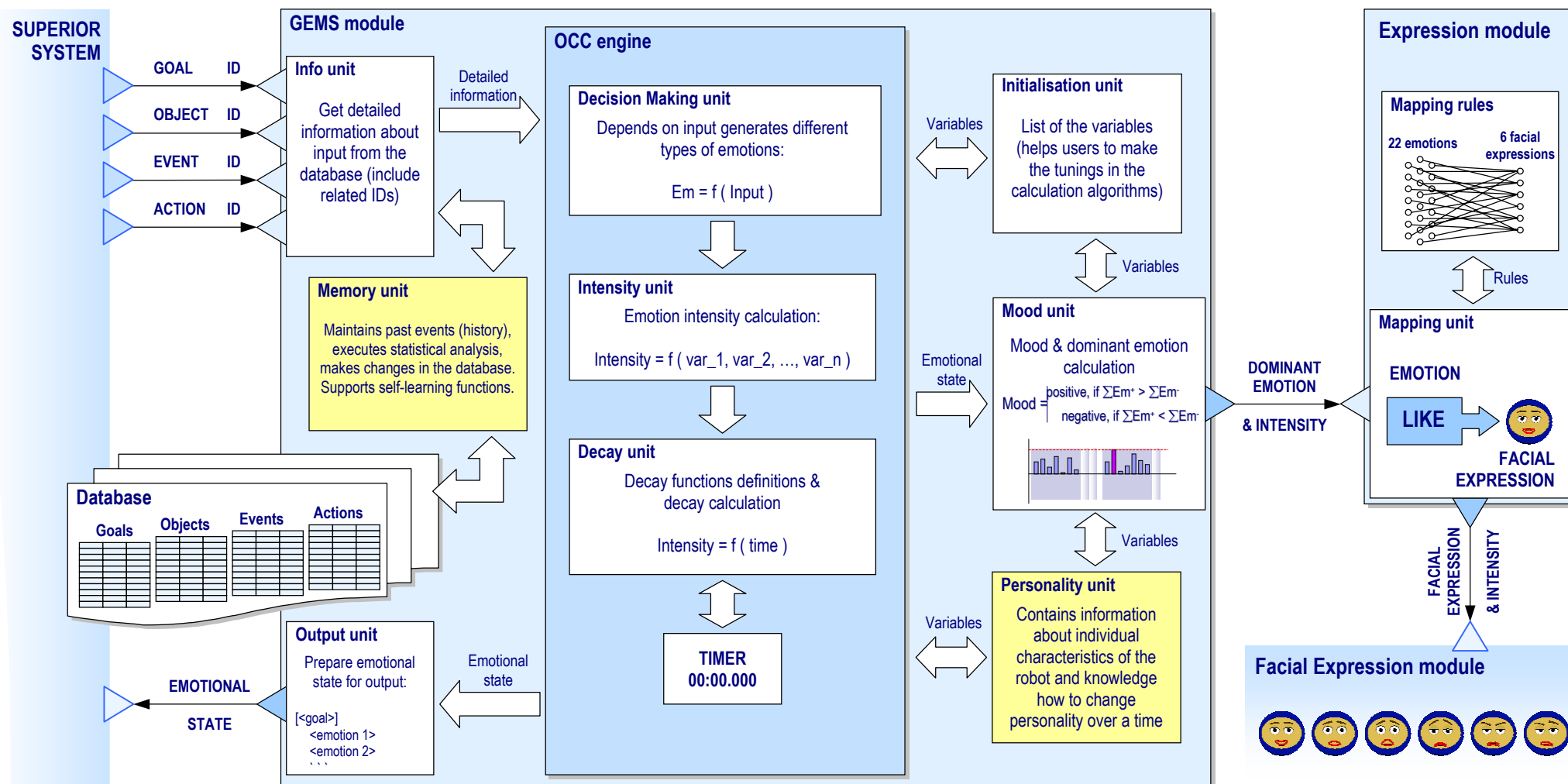
The personality of the human being is also changing when the human becomes older. The human "collects an experience" during his or her life and changes in the behaviour are consequences of this process.

It would be boring if all robots would behave in completely the same manner. A personality is necessary to provide the robot's behaviour with the individualism and the personal features.

It is possible to use the global variables from the OCC model to support such features. However, the OCC model doesn't refer to specific methods and rules for the calculation influence of these variables on the emotional state or particular emotions.

Another area of the future research is how to gather data, which are necessary to define the content of the **GEMS database**. On the current stage of the GEMS module design the robot's developers should fill in the GEMS database manually. It is time consuming and laborious work.

The existence of an Artificial Intelligence supposes an automatic filling the variables in the GEMS database and also automatic correction of these values during the "robot's life".



During the project the description of the two further user tests were defined [28].

The first concerns the human perception of the **facial expression intensity**. The result of this user test could be used for defining further relation between intensity value of the facial expression and positions of the mechanical parts on the robot's face.

The second user test concerns the **mapping rules** for the conversion from 22 types of emotion to the six facial expressions.

Also the **further user test** of the GEMS module is necessary. The current state of the robot's development (notably the speech recognition) doesn't allow people to participate at the test process. At the next stage the users should be involved into the test process and participate in the dialogues with the robot.

5. Results and conclusions

1. The software-based prototype of the General Emotion Management System (GEMS) was implemented during the project. The GEMS prototype is completely integrated into the robot architecture and can be used for creating more natural, believable behaviour and interaction between the robot and a human
2. The GEMS module demonstrates the possibility to utilise the OCC model to create an emotional behaviour of the character as the robot or other physical or virtual creature. It proved that the OCC model is useful for creating the internal emotions of the character and can be used to define the dynamic emotional state in the multigoal environment.
3. It is essential to provide the GEMS module with the necessary variables about the current goals, events, actions and objects (agents) to produce the emotional state of the robot. The most valuable variables are: 1) *desirability* of the goals; 2) *likelihood* of the goals and events; 3) *praiseworthiness* of the actions and 4) *appealingness* of the objects (agents).
4. The values of the variables are stored in the GEMS database. The structure of the GEMS database contains four tables: goals, events, actions and objects (agents). In the current stage of the robot's development the values of the variables should be filled manually by people who develop the robot. In an ideal case the GEMS database content should come out from "the life experience" of the robot. The GEMS database represents part of the Artificial Intelligence of the robot.
5. The emotional state of the robot consists of 22 types of emotion. The emotional state of the robot is presented as the space with the current goals of the robot. Each goal includes a variable number of corresponding emotions. The emotional state keeps the emotions as they were created without any blending. Each emotion at the emotion state has certain characteristics. The most important characteristic is intensity.
6. The intensity of the emotions decays in time. There are eight decay functions (including the constant and the linear functions). During this project only two of them have been used (constant and linear). The rest have been defined and implemented for usage in further research.
7. It is not enough to define the internal emotional state of the robot. The emotional state should be used to create the corresponding behaviour of the robot, e.g. body movement, voice modulation, facial expression, etc. In the current stage of the robot's development it was possible to use the emotional state only to define the facial expression of the robot.
8. The facial expression of the robot is based on the current dominant emotion. The dominant emotion has been converted to the facial expression with the help of the mapping rules, which make a connection between 22 types of emotion and six facial expressions.
9. Several methods how to select the dominant emotion were defined during the project. All these methods are based on the intensity of the emotions and used some preparation phases to "normalize" the intensity value before the comparison.

10. Three demo scenarios were implemented during the project. Two of them were used for user observation and evaluation.
11. The preliminary evaluations indicate that the robot has a high level of expressiveness, recognizable facial expressions and adequate emotional reactions during the demo scenarios. The believability of the robots behaviour was evaluated as average.
12. The observation of the demo scenarios shows that it is not enough to express emotions only with the help of the facial expressions. Voice intonations, gestures and even correct movements of the robot's eyeballs could increase the believability of the robot's behaviour.

References

- [1] C. Bartneck, M. Okada, "Robotic User Interfaces", Proceedings of the Human and Computer Conference (HC-2001), Aizu, pp. 130-140.
- [2] S. N. Reilly, "Believable Social and Emotional Agents", Ph.D. Thesis at the Carnegie Mellon University, Pittsburgh, PA, W.S.N., 1996.
- [3] A. Ortony, G. Clore, A. Collins, "The Cognitive Structure of Emotions", Cambridge, Cambridge University Press, 1988.
- [4] D. Taapken, "Using DML 1.0 (Dynamic Module Library)", Technical Note 07/2002, Philips Research, Eindhoven, The Netherlands, July 2002.
- [5] H. Noot, Zs. M. Ruttkay, "CharToon 2.0 Manual", Report INS-R0013, CWI, Amsterdam, The Netherlands, June 2000.
- [6] PVM: Parallel Virtual Machine, <http://www.epm.ornl.gov/pvm/>
- [7] P. Ekman, "Emotion in the Human Face", Cambridge University Press, New York, 1982.
- [8] M. Minsky, "The Society of Mind", New York: Simon and Schuster, 1986.
- [9] R. Woodworth, H. Schlosberg, "Experimental Psychology", New York: Henry Holt and Co., revised edition, 1954.
- [10] R. Picard, "Affective computing", The MIT Press, 1997.
- [11] C. D. Elliot, "The Affective Reasoner: A process model of emotions in a multi-agent system", Dissertation, Evanston, Illinois, USA, June 1992.
- [12] J. S. Dumas, J. C. Redish, "A practical Guide to Usability Testing", Intellect Ltd, 1999.
- [13] J. Nielsen, "The Use and Misuse of Focus Groups", 1997.
- [14] "Jakob Nielsen's Website", <http://www.useit.com/>
- [15] Epictoid BV company website, <http://www.epictoid.nl/>
- [16] D. Dorner, K. Hille, "Artificial Souls: Motivated Emotional Robots", Bamberg University, 1999.
- [17] C. Bartneck, M. Okada, "Robotic User Interfaces", Philips Research, Eindhoven, 2001.
- [18] eMuu project website, <http://www.bartneck.de/work/emuu/>
- [19] C. Bartneck, "eMuu – An Embodied Emotional Character for the Ambient Intelligent Home", Ph.D. these, Technical University of Eindhoven, 2002.
- [20] Sociable Machines Project (Kismet), Artificial Intelligence Laboratory, MIT, <http://www.ai.mit.edu/projects/humanoid-robotics-group/kismet/kismet.html>

- [21] C. Breazeal, "Sociable Machines: Expressive Social Exchange Between Humans and Robots". Sc.D. dissertation, Department of Electrical Engineering and Computer Science, MIT, 2000.
- [22] A.J.N. van Breemen, K. Crucq, B.J.A. Krose, M. Nuttin, J.M. Porta, W. Souverijns, "Development of a Domestic User-Interface Robot with Emotional Feedback", IEEE International Conference on Robotics & Automation (ICRA2003), Taipei, Taiwan, 2003.
- [23] K. Crucq, A.J.N. van Breemen, B. Krose, M. Nuttin, F. Kuijk, "Domestic User-Interface Robot with Emotional Feedback", ITEA Meeting, Turijn, 2002.
- [24] A.J.N. van Breemen, K. Crucq, "An Interactive Module-Based Software Architecture for Experimental Autonomous Robots", submitted to 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems.
- [25] T. Fong, I. Nourbakhsh, K. Dautenhahn, "A Survey of Social Robots", Elsevier Science, 2002.
- [26] M.S. El-Nasr, "Modelling Emotion Dynamics in Intelligent Agents", Master of Science thesis, Texas A&M University, 1998.
- [27] C. Bartneck, "Affective Computing Portal",
http://www.bartneck.de/link/affective_portal.html
- [28] A. Bondarev, "GEMS User Testing Description", Philips Research, 2002.

Appendix A INI file content

A.1 Global constants

The developer can change the global constants only before starting the GEMS module. The changes will take no effect during the operation of the GEMS module. If the developer wants to change a constant he should rerun the module after the changes.

Tick – loop (or tick) time in milliseconds, indicates how often the emotional state will be updated.

Range: integer 1...2147483647

Default value: _Tick_ = 1000

A.2 Global variables

During the research process a developer can change the values of variables in the INI file. To accept these changes the developer has to send a "-1" value to the Goal ID input port; the GEMS module will reread the INI file. It is very useful during the turning process.

MaxEmTime – max LifeTime (in seconds) of the emotions, indicates how long to keep information about emotion after decaying intensity to 0

Range: integer 1...2147483647

Default value: _MaxEmTime_ = 300

defSpeed – default "speed" for the decay functions, indicates the time (in seconds) of decaying max intensity (100%) to 0%

Range: integer 1...2147483647

Default value: _defSpeed_ = 20

defHFSpeed – default "speed" for the Hope and Fear types of the emotions (in seconds)

Range: integer 1...2147483647

Default value: _defHFSpeed_ = 10

defDecay – type of the default decay function

Range: string d_Const | d_Linear | d_pParabol | d_iParabol | d_tParabol | d_pParabolInv | d_iParabolInv | d_tParabolInv

Default value: _defDecay_ = d_Linear

priorLifeTime – calculation method for the dominant emotion: to take into account the emotion LifeTime (0 – no influence, 1 – to use the formula (2.2))

Range: integer 0 | 1

Default value: `_priorLifeTime_ = 1`

`_priorMapCoeff_` – calculation method for the dominant emotion: to take into account the mapping coefficient from the emotions to the Facial Expression disk. This parameter describes what is important for the calculation dominant emotion: emotions intensity or facial expression intensity (0 – emotion intensity, 1 – facial expression intensity).

Range: `integer 0 | 1`

Default value: `_priorMapCoeff_ = 1`

`_domGoal_` – calculation method for the dominant emotion: indicates how to find the dominant emotion in the multigoal space (0 – to use the complete set of the emotions from the all goals, 1 – to use the emotions only from the goal with the highest hierarchical value)

Range: `integer 0 | 1`

Default value: `_domGoal_ = 0`

`_domMood_` – calculation method for the dominant emotion: indicates how to find the dominant emotion with the current Mood (0 – no influence (do not take into account the current mood, 1 – to make the search among the positive or negative emotions depended on the current Mood positive or negative value)

Range: `integer 0 | 1`

Default value: `_domMood_ = 0`

A.3 Variables for log output

This section includes variables which helps the developer to manipulate with output information of the GEMS module during the work.

`_printEm_` – to output the information about emotions (type, intensity, lifetime and decay time)

Range: `string YES | NO`

Default value: `_printEm_ = YES`

`_printZeroEm_` – to output the information about emotions with the intensity = 0

Range: `string YES | NO`

Default value: `_printZeroEm_ = NO`

`_printGoal_` – to output the information about the goals (name and hierarchical rank)

Range: `string YES | NO`

Default value: `_printGoal_ = YES`

`_printFaceExpr_` – to output the information about current facial expression (name and intensity)

Range: `string YES | NO`

Default value: `_printFaceExpr_ = YES`

`_printINIVar_` – to output information about variables (after reading from INI file)

Range: string YES | NO

Default value: `_printINIVar_ = NO`

A.4 Information about emotions

The necessary information about the emotions is presented in the following format:

Emotion = `<name>` – the name of the emotion

`{` – open bracket indicates the start point of the section with the data for the particular emotion

Positivism = `-1|0|1` – indicates the position of the emotion in the highest level of hierarchy (negative | neutral | positive)

FaceExpr = `<type_of_the_FaceExpr>` – the name of the correspondent facial expression

MixPers = `0...100` – the percentage value of this facial expression in the mix with another facial expression

MixWith = `<type_of_FaceExpr>` – the name of the second facial expression in the mix

CoefFaceExpr = `0...100` – mapping coefficient, %

PairEm = `<name_of_emotion>` – the name of the opposite emotion in the highest level of the emotional hierarchy (positive - negative)

`}` – close bracket – end of the data section.

Appendix B GEMS Database

A.5 Goal table (part of the file: goal.db)

```

;=====
;Information about GOALS
;=====
;Format:
;---      // separator - indicates start point of the new record
;ID        // id number,                int  (1...?)
;Name      // name,                    char* (0...255 chars)
;Hierarchy // hierarchy,                int  (0...?)
;DoS       // desirability of success,  int  (0...100)
;DnF       // desirability of not fail,  int  (0...100)
;LoS       // likelihood of success,    float (0...1)
;LoF       // likelihood of fail,       float (0...1)
;Agent     // related agent id,         int  (1...?)
;=====
---
1
Find red ball
50
50
50
0.5
0.5
1
---
2
Keep battery charged
100
100
100
0.5
0.5
1
---
3
Butler functions (wait for a command)
70
70
70
0.5
0.5
1
---
4
Wake up Homer
50
40
90
0.5
0.5
1
---

```

A.6 Object (Agent) table (part of the file: object.db)

```

;=====
;Information about OBJECTs (AGENTs)
;=====
;Format:
;--- // separator - indicates start point of the new record
;ID // id number, int (1...?)
;Name // name, char* (0...255 chars)
;Type // type not_human|human, int (0|1)
;Appeal // appealiness, int (-100...100)
;Goal // related goal id, int (1...?)
;=====
---
1
ME (robot itself)
1
0
-1
---
2
Red ball
0
50
1
---
3
Homer
1
80
11
---
4
Homers father
1
70
11
---
5
TV
0
0
0
---
6
Andy
1
80
0
---
7
Banana
0
90
0
---
```

A.7 Event table (part of the file: event.db)

```

;=====
;Information about EVENTS
;=====
;Format:
;--- // separator - indicates start point of the new record
;ID // id number, int (1...?)
;Name // name, char* (0...255 chars)
;dLoS // change in goal LoS, float (-1...1)
;dLoF // change in goal LoF, float (-1...1)
;Goal // related goal id, int (1...?)
;=====
---
1
Red ball is detected
1
-1
1
---
2
Red ball is Not detected on this side
-0.125
0.125
1
---
3
Red ball is NOT detected
-1
1
1
---
4
Mr Homer is waked up
1
-1
4
---
5
Battery charged +10%
0.2
-0.2
2
---
6
TV is switched on
1
-1
5
---
9
Command is received
1
-1
3
---

```

A.8 Action table (part of the file: action.db)

```

=====
;Information about ACTIONS
=====
;Format:
;---          // separator - indicates start point of the new record
;ID           // id number,                int  (1...?)
;Name         // name,                     char* (0...255 chars)
;Priseworth   // praiseworthiness,         int  (-100...100)
;Agent        // related agent id,         int  (1...?)
;Responsib    // responsibility of the agent, float (0...1)
;Event        // related event id,         int  (1...?)
=====
---
1
I found red ball
30
1
1
1
---
2
I did not find red ball
-30
1
1
3
---
3
Wake up Mr Homer in time
30
1
1
4
---
4
Switch on TV
10
1
1
6
---
5
Switch off TV
10
1
1
9
---
7
Homer did not wake up
-70
3
1
11
---

```

Appendix C User scenarios

A.9 "Search of the red ball"

This scenario was implemented in two possible variants: when the goal is reached successfully (the red ball is found) and when the goal is not reached (the red ball is not found).

Human: Lino, wake up.

Lino: Yes?

Human: Find red ball.

1st variant: the goal succeeds.

Lino turns to left and starts searching the red ball. He does not find the red ball at this side. He becomes slightly sad.

Lino: No red ball at this side.

Lino turns left again. He found the red ball. He is happy. He turns to the human.

Lino: I found the red ball.

2nd variant: the goal fails.

Lino turns to left and starts searching the red ball. He does not find the red ball at this side. He becomes slightly sad.

Lino: No red ball at this side.

Lino repeats the same action three times and searches the red ball at the all four sides of the room. After each unsuccessful search he becomes sadder and sadder. When the search on the last side was done unsuccessfully he becomes very sad.

Lino: I did not find the red ball.

A.10 "Emotional conversation"

This is the demo conversation between human (*Andy*) and the robot (*Lino*).

Lino is sleeping, his head is lowered and eyes are closed. Andy is sitting not far from Lino. He wants to talk with Lino.

Andy: Lino, wake up.

Lino raises his head, opens eyes and turns to Andy.

Lino: Yes?

Andy: Come to me.

Lino comes to Andy and stands in front of him.

Lino: I am here.

Andy: Good boy.

Lino starts smiling.

Andy: Look what I have.

Lino: Show me.

Andy shows the banana to Lino. Lino recognizes the object as banana.

Lino: Banana!

Lino shows happiness and surprise.

Lino: I like bananas.

Andy: Blah-blah-blah ... (*not recognizable words*)

Lino: Sorry, I don't understand.

Lino shows light sadness.

Andy: Blah-blah-blah ... (*not recognizable words*)

Lino: Sorry, I don't understand what you are saying.

Lino becomes very sad.

Andy: Stupid machine!

Lino: Hey, don't hurt me.

Lino moves his head forward and shows anger.

Andy: Ok, ok, I am sorry.

Lino: No problem, forget it.

The anger on the Lino's face is decaying and disappearing.

Andy: I have another thing. Can you recognize it?

Lino: Show me.

Andy shows the lemon to Lino. Lino can't recognize the object.

Lino: Sorry I cannot recognize

Lino becomes sad.

Andy: This is the lemon.

Lino: Lemon, I don't like lemons.

Lino shows disgust.

Andy: Let's play in word game.

Lino: Sorry Andy, my battery is low. I have to recharge it.

Lino shows sadness.

Andy: Ok, Lino.

Lino moves to his charge station...

A.11 "Home interaction"

7:54 a.m., Wednesday. Today is the usual morning of the working day. Mr. Homer is sleeping. Lino is standing at the special parking place with the connection to the power line for battery recharging. It seems that he is sleeping. His eyes are closed; the head is lowered.

Suddenly Lino starts moving. He raises head, opens the eyes and looks around. Then he goes to the parent's bedroom. In the bedroom he switches on the sound system and run to play the special melody for wake up. Very soft melody starts playing with the gradually increasing loudness.

Lino: Mister Homer, wake up.

Homer continues to sleep.

Lino: Mister Homer, time to wake up

Homer is still sleeping. Lino becomes angry.

Lino: Hey, wake up, NOW!

Mr. Homer rises from a bed ...

Lino: Good morning!

Lino is smiling.

Homer: Good morning Lino

Lino: How are you today?

Homer: Good. How are you?

Lino: Fine. All systems work stable. Battery is charged sixty percents.

Lino is smiling.

Homer: Ok, Lino. Please switch on TV.

Lino turns to the right and stars looking for TV. He can't find the TV.

Lino: I can't find the TV. Where is it?

Lino is sad.

Homer: Oh, I've moved it to another side of the room. Turn left.

Lino turns to the left.

Lino: Oh, I found the TV.

Lino becomes happy.

He is trying to switch on the TV.

Lino: I can't switch TV on. Probably it unplugged from the power or broken.

Lino is slightly sad.

Homer: Oh, sorry I forgot to plug it into the power.

Homer comes to the TV and plugs it into the power socket.

Homer: Lino, you can switch on TV now.

Lino switches on the TV and turns to Homer.

Homer: Thanks Lino. Could you check my e-mail?

Lino: Yes, just a moment.

Lino is happy. He freezes for a moment.

Lino: You have two new messages. The first message is from Andy. The second is from your father.

Lino is very happy.

Homer: Lino, please read the second message.

Lino: Hi son, we with mum did not see you for a long time. We are missing you very much. Whether you will come to us tomorrow for dinner? Mum will prepare amazing apple pie (as usual). Daddy. End of message.

Lino becomes neutral smoothly.

Homer: OK Lino, please write the answer.

Lino: I am ready to record the answer.

Homer: Dear Daddy. I am glad to accept you invitation. See you. Homer.

Lino: The message is recorded. The message was sent successfully. You have one unread message from Andy. Do you want to read it?

Homer: No, thanks Lino, you can go back to you parking place.

Lino: Ok, just call me when you will need something. I have to go to recharge my battery.

Lino goes to his parking place. At the parking place he stars looking for the charger. he can't find it.

Lino: Oh my god, where is my charger?

Lino is very sad.

Lino: Homer, I can't find my charger!

Homer: It was broken. I threw it away.

Lino is scared.

Lino: You have to go to buy the new one immediately!

Homer: OK, I am going.

Lino: The energy is my life. I have to save the rest of my energy.

Lino goes to sleep.

Appendix D Questionnaire

Questionnaire

Emotional robot (demo scenarios):

1. Emotional conversation

2. Home interaction

1. What is your **general impression** about these demos?

2. How often was it difficult for you to recognize the **facial expression** of the robot?

☐

never

☐

sometimes

☐

often

☐

very often

☐

always

Which one did you see?

3. How often did the robot have **adequate reactions** during the interactions?

☐

never

☐

sometimes

☐

often

☐

very often

☐

always

Why?

4. Do you think that the robot has an **emotional state** and shows it on the mechanical face?

☐ yes

Why?

☐ no

5. How would you rate the **level of expressiveness** of the robot?

☐

very low

☐

low

☐

average

☐

high

☐

very high

Do you think this is appropriate level of expressiveness for such robot? Why?

6. How would you rate the **level of believability** of the robot's behaviour?

☐
very low

☐
low

☐
average

☐
high

☐
very high

What **criteria** did you use for an estimation of this characteristic?

7. How do you think we could **increase** the level of believability of the robot's behaviour?

8. How do you think we could **improve** the current scenarios?

9. **Other comments**, suggestions, opinions, etc.
