An easily adaptable Decision Making Framework based on Somatic Markers on the Nao-Robot

Jens Hoefinghoff * **Laura Steinert** * **Josef Pauli** *

* *Universitaet Duisburg-Essen, Fakultaet fuer Ingenieurwissenschaften, Bismarckstrasse 90, 47057 Duisburg (e-mail: jens.hoefinghoff@uni-due.de; laura.steinert@stud.uni-due.de; josef.pauli@uni-due.de)*

Abstract: Decision making is an essential aspect of artificial intelligent systems such as robots. The adaptation to variant applications of decision making approaches often involves the implementation of previous knowledge and additional changes in the source code. This paper presents improvements to the implementation of an emotional decision making approach. These improvements, composed of an expansion of the decision making algorithm and enhancements of the implementation architecture, ensure a user-friendly and fast adaption to variant applications.

Keywords: Decision making, Autonomous mobile robots, Implementation

1. INTRODUCTION

Decision making is an essential factor for the creation of autonomous systems. Studies show that the human decision making process involves emotions. Therefore, the emotion modelling for artificial intelligent systems has become an important subject in the creation of authentic behaviour of machines, such as robots. Certainly, the choice of the decision making algorithm depends on the application, but in the field of robot companions, a human like behaviour could be preferable to a certain extent. In (Dautenhahn et al., 2005) a study concerning the role of robot companions is presented. The results show that humans prefer a predictable (90%) and controllable (71%) robot behaviour. Most of the subjects (71%) prefer a human-like communication with a robot companion. A human-like behaviour is rated as less essential, however, 36% of the subjects state that they appreciate human-like behaviour. Considering that some aspects of human-like behaviour are more preferable than others, e.g. the possibility of learning behaviour vs. defiant behaviour, some human abilities are still interesting for robot companions. In fact most subjects would like to control and influence the robot's behaviour, which means that the decision making process of robot companions is a major issue.
A popular theory concerning the human decision making process is postulated by Damasio, who basically divided this process into two steps (Damasio, 1994). In his Somatic Marker Hypothesis he describes that the first part of the decision making process is based on emotions, in order to reduce the number of options for a further rational analysis. This emotional filtering process is based on somatic markers, which reflect the emotional memory for each pair of a stimulus and an action.
In (Hoefinghoff and Pauli, 2012) a decision making algorithm based on artificial somatic markers is presented and evaluated in a simulated environment. For the purpose of transferring this algorithm to the real humanoid Nao-robot of the Aldebaran company, an architecture is chosen, which is based on the human brain structure. The intention of the created algorithm and implementation design on the robot should be the creation of a system which is easily adaptable to different applications without the need to change the source code. This allows even users without programming skills to work with the system.
In the following chapters further enhancements of the algorithm, such as the previously missing rational selection part and the possibility to include human-like innate behaviour patterns, are presented. Furthermore, details of the interaction between modelled brain parts are given. Finally, the configuration of an exemplary application is shown in order to demonstrate the single steps a user has to perform.

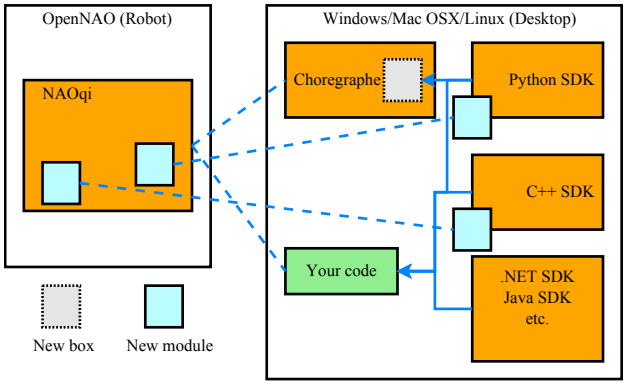

Fig. 1. Provided software architecture of the Nao-robot by the Aldebaran company.

2. BASIC CONCEPTS FOR PROGRAMMING THE NAO-ROBOT

The Nao-robot of the Aldebaran company comes with a considerable Software Development Kit (SDK) for programming (Aldebaran Robotics, 2012) and a software (Choregraphe) to create behaviour networks with a high-level GUI. To ensure a modular development of software for the Nao-robot, the main concept is based on a broker architecture. Therefore, the main software NAOqi, to which new modules can register, is running on the robot. A module can be executed as a local module on the robot (only C++ and Python) or remotely on a desktop PC (all SDKs). For the creation of boxes in the Choregraphe only Python code is supported. An overview of the described architecture is shown in figure 1. Nonetheless, functions of modules written in another supported language, e. g. C++, can be called from within the box's Python code. For all in the following described modules (written in C++), a box in the Choregraphe is created, which can always consist of different parts.

(1) Parameters: A box can have several parameters, whose values can be accessed in the corresponding Python code.
(2) Inputs: A box can receive signals through several inputs. Each input is dedicated to a piece of code, which is executed when a signal is received through this input.
(3) Outputs: A box can have several outputs which can be used to send signals to other boxes.

Furthermore, it is possible that input and output signals relay values and it is also possible cascade several boxes. Figure 2 shows an exemplary box with one input (named start) and two outputs (named case1 and case2). The box gets the class name of the module which should be addressed, the ip of the robot and the port of the NAOqi software. The first step in the corresponding Python code (see listing 1) consists of obtaining a proxy object of the desired module. Therefore, the user-given parameters Modulename, IP, and Port are used. Furthermore, with *onInput_start(self)*, the code that should be executed when a signal is received via the *start* input is defined. Here, the function f() of the module is called, whose return type is an integer value. If the returned value of the function is 0, a signal is sent through the output case1, otherwise the output case2 sends a signal.

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)
        self.ProxyOfModule = ALProxy(
        self.getParameter("Modulename"),
        self.getParameter("IP"),
        self.getParameter("Port"))

    def onLoad(self):
        pass

    def onUnload(self):
        pass

    def onInput_start(self):
        result = self.ProxyOfModule.f()
        if result == 0:
            self.case1()
        else:
            self.case2()
```

Listing 1. Exemplary python code

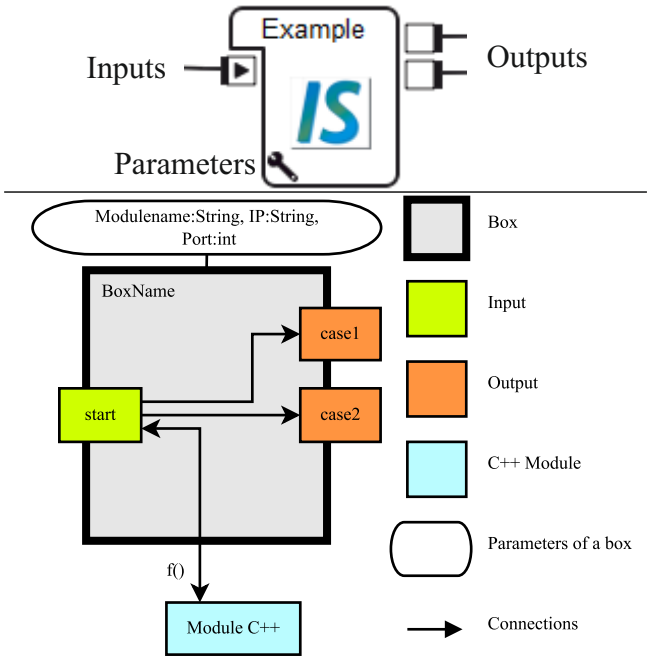

Fig. 2. Exemplary box in the Choregraphe (top) and a schematic representation (bottom).

In addition, the Nao-robot comes with a global memory (ALMemory) which is accessible from all modules. Modules are able to write variables into it and to subscribe to variables in the ALMemory, so that they receive a notification when the value is changed. Also, most of the robot's sensors provide their measurements through the ALMemory.

3. SUMMARY OF THE EXISTING APPROACH

In this chapter, an overview of the decision making approach and the implementation on the Nao-robot is given. First, the functionality of the decision making algorithm and its results in the simulation will be discussed. Afterwards the used implementation architecture is described.

3.1 Decision making algorithm

In (Hoefinghoff and Pauli, 2012), a mathematical model for somatic markers and a thereon based decision making algorithm is described. The modelled agent consists of the following:

- A set $S = \{s_1, ..., s_m\}$ that contains all stimuli that could be recognized. A stimulus can be a single signal or sensory value but also a combination of different inputs that describe a whole situation.
- A set $A = \{a_1, ..., a_n\}$ which contains all actions that could be executed.
- m sets $R_{s_i} = \{r_1, ..., r_l\}$ which contain all possible rewards that can be received for executing an action in consequence of each stimulus. Just like in real life, some decisions are riskier than others, e.g. the gravity of the worst case scenario depends on the existing stimulus. Therefore an own set of rewards for every stimulus is necessary.

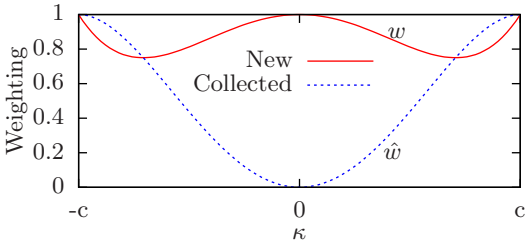

Fig. 3. Weighting of new and collected knowledge, depending on reliability κ_i.

Based on the number of stimuli and actions, the agent creates a matrix that contains all somatic markers (see (1)). Each value $\sigma_{i,j}$ represents the made experience with the execution of an action a_j, after a stimulus s_i was recognized. Those values are the basis for the following decision making algorithm.

$$M = \begin{pmatrix} M_1 \\ \vdots \\ M_m \end{pmatrix} = \begin{pmatrix} \sigma_{1,1} & \cdots & \sigma_{1,n} \\ \vdots & \ddots & \vdots \\ \sigma_{m,1} & \cdots & \sigma_{m,n} \end{pmatrix} = (\sigma_{i,j}) \qquad (1)$$

According to Damasio's theory, the algorithm should filter the number of actions for the subsequent rational analysis, therefore the output of the algorithm is defined as a set $A' \subseteq A$. The definition of A' is shown in (2), in which θ_i is an individual, dynamically adapted threshold for every stimulus. This threshold can be seen as a frustration level. A lower value indicates a higher frustration, which possibly leads to a consideration of more options.

$$A' := \{a_j \in A \,|\, \sigma_{i,j} >= \theta_i\} \qquad (2)$$

As the rational selection has not been considered so far, an action a_j is randomly selected out of the subset A' and executed. After the execution a reward $r_{i,j}$ is given, which is used to update the corresponding somatic marker and the threshold. The update of the somatic marker is shown in (3) and (4), in which w and $\hat{w}$ are weighting parameters for the inclusion of new knowledge and collected knowledge. Figure 3 shows the weighting, depending on the value κ_i, which expresses the reliability of collected knowledge, while $c \in \mathbb{N}$ is a user-given constant. Additional information about the scaling of rewards and the computation of κ_i can be found in (Hoefinghoff and Pauli, 2012).

$$\overline{r_{i,j}^t} = w \cdot r_{i,j}^t + \hat{w} \cdot \overline{r_{i,j}^{t-1}} \qquad (3)$$

$$\sigma_{i,j}^{t+1} = \tanh(\overline{r_{i,j}^t}) \qquad (4)$$

For the update of the frustration level θ_i, the same computation as for the somatic marker is used. The difference is the number of updates. While the frustration level will always be updated when a stimulus is recognized, the update of a somatic marker is depending on a combination of both a stimulus and an action. In a nutshell, the algorithm consists of the following steps:

(1) Recognition of a stimulus s_i
(2) Selection of subset A' respective to s_i

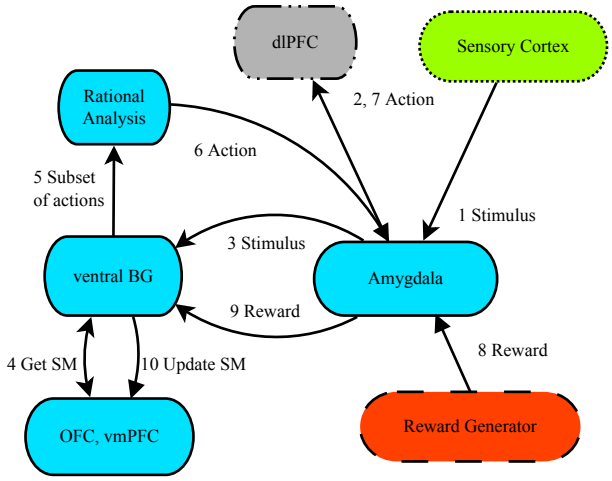

Fig. 4. Software architecture for the implementation on the Nao-robot, which is adapted from (Vitay and Hamker, 2011). The different parts can be divided into four areas of responsibility: stimulus recognition (green), decision making (blue), execution of actions (grey) and creation of rewards (red).

(3) Random choice of action a_j from A'
(4) Reception of reward r^t for the executed action a_j
(5) Update of the corresponding somatic marker $\sigma_{i,j}$
(6) Update of the frustration level θ_i
(7) Update κ_i

For evaluation purposes, the Iowa Gambling Task (IGT) was used. The IGT, in which human subjects have to choose cards out of four different decks, was developed by Bechara and Damasio to support the Somatic Marker Hypothesis (Damasio, 1994; Bechara et al., 1994). Every card gives the subject a benefit or a penalty. Two decks are disadvantageous and the other two are advantageous. The results show that the modelled agent and human subjects both generally prefer the advantageous decks, while simultaneously preferring disadvantageous decks in the early phases of the experiment. This effect is due to higher benefits in the disadvantageous decks. Differences can be observed especially in later phases, at which human subjects occasionally chose disadvantageous decks, whereas the agent kept choosing the advantageous decks. This difference can be explained by other factors involved in the human decision making process, like for example personal characteristics such as curiosity or risk taking. These factors are not considered in the algorithm.

3.2 Implementation of the algorithm on the Nao-robot

In order to transfer this algorithm to the Nao-robot, a modular architecture is chosen, which models the involved human brain parts and their interactions among each other. Figure 4 shows the architecture that is presented in (Hoefinghoff et al., 2012) and highly influenced by the work of (Vitay and Hamker, 2011), in which an overview of the brain parts involved in the emotional decision making process is given. Of course the architecture and the function of each part is simplified, especially as a clear separation of some functions is not possible.
The main issue of the developed algorithm and its implementation architecture is a fast and easy adaption to

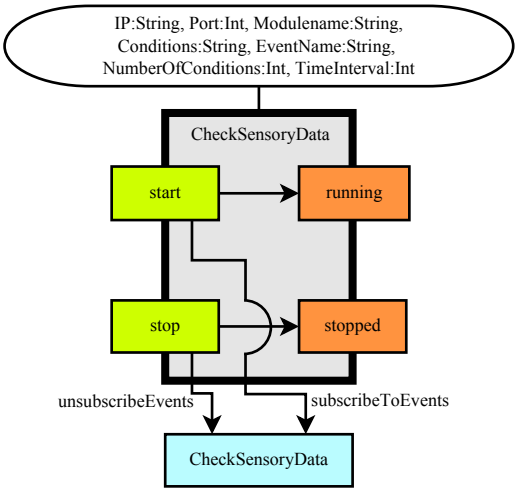

Fig. 5. The CheckSensoryData box can be used to create different stimuli without the need to change any source code.

different applications. The adaption should even be possible for users without programming skills. In (Hoefinghoff et al., 2012), a detailed description of the stimuli creation is given, so that the user can define to which situations the agent should react. As described above, the agent consists of a set S that contains all stimuli, each of them a single sensory value or a whole situation. Therefore a module CheckSensoryData and a corresponding box in the Choregraphe have been developed (see figure 5), which offers users the possibility to create new stimuli without programming. For the parameter *Conditions*, the user can define multiple conditions for sensory values or other values in the ALMemory. Those conditions will be checked periodically. When at least the number of conditions given by *NumberOfConditions* is fulfilled, a variable with the name specified in *EventName* is written or updated in the ALMemory.
Every time the variable is updated, which means that the modelled stimulus is present, a further module called SensorySystem (part of the sensory cortex) is notified and relays the name of the occurred stimulus to the amygdala for further processing. Then the decision making process starts. Its output is an action that is sent to the dorsolateral prefrontal cortex (dlPFC), which initiates the execution.

4. IMPROVEMENTS OF THE APPROACH

In the following, the enhancements and further explanations concerning the decision making framework are presented. First, the implementation of the modules that are involved in the decision making process is described in depth. The explanation of the executing part and the generation of rewards follows. Further information on the stimulus recognition can be found in (Hoefinghoff et al., 2012). For the description of the interactions between the modules, the notation shown in (5) is used. Semantically, the notation reflects that if the box *BoxName* receives a signal through the input *InputName* with some values, it will send a signal to the *Receivers* through the output *OutputName* with some values.

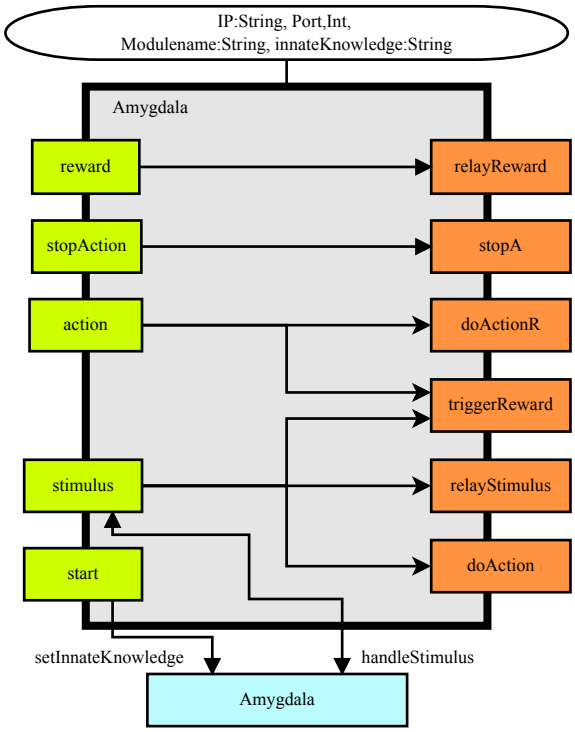

Fig. 6. The amygdala is the control centre of the framework and spreads all information.

$$BoxName : I(InputName[Values]) \xrightarrow{Receivers} O(OutputName[Values]) \quad (5)$$

4.1 Amygdala

As the amygdala plays a key role in the somatic marker theory, it is implemented as a kind of control centre in which nearly all information is incoming and spreading to the dedicated parts. Figure 6 shows the box for the Choregraphe. When the amygdala is receiving a stimulus, it can trigger the execution of an innate behaviour pattern. Furthermore, the stimulus is relayed to the ventral basal ganglia (ventral BG) for further analysis. After the execution of an action, the amygdala collects the rewards in order to initiate the update of the somatic marker.
With the parameter *innateKnowledge*, the user is able to define one pre-wired action for each stimulus. Every time a stimulus comes in, the function handleStimlus is called and returns the innate action. The innate action is sent to the dlPFC for execution (6) without further analysis.

$$Amygdala : I(stimulus[s_i]) \xrightarrow{dlPFC} O(doAction[a_j]) \quad (6)$$

No matter if an innate action exists, the stimulus is sent to the ventral basal ganglia, in which the emotional selection is performed (7).

$$Amygdala : I(stimulus[s_i]) \xrightarrow{ventralBG} O(relayStimulus[s_i]) \quad (7)$$

In order to notify the framework that a reward can be obtained, a signal is sent through the output triggerReward. The name of the stimulus and the action are sent as values, so that the reward can be clearly assigned (8).

$$Amygdala : I(stimulus[s_i]) \xrightarrow{RewardGenerator} O(triggerReward[s_i, a_j]) \quad (8)$$

Beside the possibility to trigger the execution of an innate action, the amygdala receives the action which is chosen by the parallel decision making process. If the chosen action differs from the innate one, the new action is sent to the dlPFC. Otherwise no signal is sent, which means that the previously started action is continuously executed (9). When a signal is sent, the output triggerReward is also activated (10).

$$Amygdala : I(action[s_i, a_j]) \xrightarrow{dlPFC} O(doActionR[a_j]) \quad (9)$$

$$Amygdala : I(action[s_i, a_j]) \xrightarrow{RewardGenerator} O(triggerReward[s_i, a_j]) \quad (10)$$

When a reward is given, it is sent to the amygdala, which relays the information to the ventral BG, in order to initiate the update of the somatic marker (11).

$$Amygdala : I(reward[s_i, a_j, r_{i,j}]) \xrightarrow{ventralBG} O(relayReward[s_i, a_j, r_{i,j}]) \quad (11)$$

Furthermore, the amygdala can receive a signal to initiate the interruption of an action. This is helpful when a negative reward is obtained, which supersedes the continuation of the action (12).

$$Amygdala : I(stopAction[a_j]) \xrightarrow{dlPFC} O(stopA[a_j]) \quad (12)$$

4.2 Ventral basal ganglia

The ventral basal ganglia are part of the forebrain. One of their main functions is the action selection based on the information provided by the amygdala, the orbitofrontal cortex (OFC) and the ventromedial prefrontal cortex (vmPFC). Figure 7 shows the schematic visualization of the ventral BG.
When the box is started, all available actions, provided by a variable in the ALMemory, are set. From now on

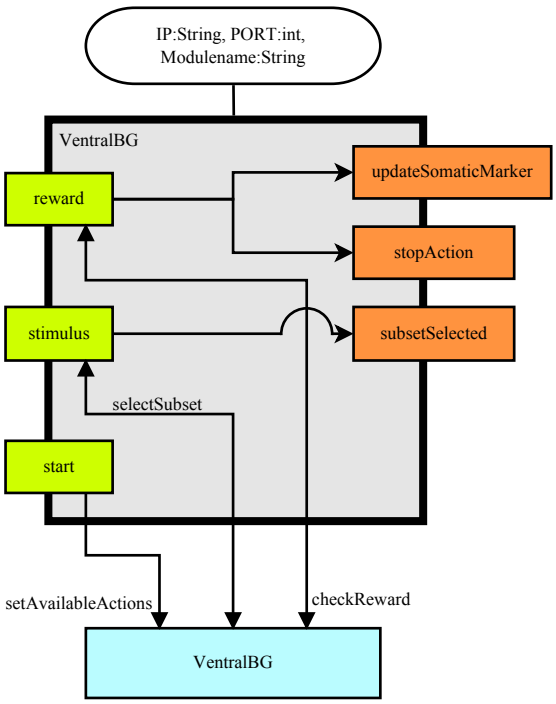

Fig. 7. The ventral basal ganglia are mainly responsible for the emotional preselection.

the box is ready to handle incoming rewards and stimuli. The module is performing the emotional preselection based on the above described algorithm. The information about the somatic marker values and the other parameters is read from the ALMemory. If a stimulus comes in, the handleStimulus function is called, which returns the set $A' \subseteq A$. This subset is sent to the following rational analysis (13).

$$VentralBG : I(stimulus[s_i]) \xrightarrow{RationalAnalysis} O(subsetSelected[s_i, A']) \quad (13)$$

Additionally, the ventral BG are receiving information on the rewards and are able to initiate the interruption of an action when the checkReward function identifies a negative reward (14).

$$VentralBG : I(reward[s_i, a_j, r_{i,j}]) \xrightarrow{Amygdala} O(stopAction[a_j]) \quad (14)$$

In order to update the somatic marker and the other parameters, the reward is relayed to the OFC and vmPFC.

$$VentralBG : I(reward[s_i, a_j, r_{i,j}]) \xrightarrow{OFC,vmPFC} O(updateSomaticMarker[s_i, a_j, r_{i,j}]) \quad (15)$$

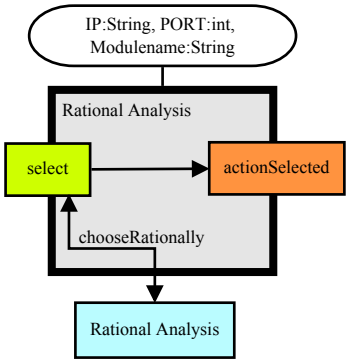

Fig. 8. After the emotional preselection, the final action will be chosen in the rational analysis.

4.3 Rational analysis

The algorithm presented in section 3.1 selects a subset of all possible actions according to the emotional experience. Afterwards an action is randomly picked out of the remaining subset and executed. Yet the underlying somatic marker theory by (Damasio, 1994) states for the human decision making process that the action to be executed is chosen rationally from the subset. In order to comply with this part of the somatic marker theory, the presented algorithm is extended by a rational selection (see figure 8 and equation (16)).

$$\begin{aligned} &RationalAnalysis : I(select[s_i, A']) \\ &\qquad \overset{Amygdala}{\rightarrow} \\ &O(actionSelected[s_i, a_j]) \end{aligned} \tag{16}$$

In the architecture implemented on the Nao-robot, a single, independent module selects the action that is supposed to be executed from the given subset. This allows an easy exchange of this module with a module whose selection is rationally motivated. One possible algorithm for the rational selection is given below.
Whenever the agent executes an action, it is logged, together with the received reward in a set of variables in the ALMemory. In total the β last actions and their rewards are stored for every stimulus. When the agent receives an ordered subset A' of all possible actions A to choose from, the history H_{s_i} of the current stimulus s_i is taken into account.
Let H_{s_i} consist of a maximum of β pairs $p \in (A \times R)$ where R is the set that contains all possible rewards. For every action $a_j \in A'$ the average reward $\widehat{r_{i,j}}$ is calculated based on the occurrences of a_j in the history H_{s_i} ((17), (18)).

$$\widehat{r_{i,j}} = \frac{1}{|R_{a_j}|} \cdot \sum_{r_{i,j} \in R_{a_j}} r_{i,j} \tag{17}$$

$$R_{a_j} = \{r_{i,j} \mid (a_j, r_{i,j}) \in H_{s_i}\} \tag{18}$$

Thus, if no pair with a_j as the first parameter is included in H_{s_i}, $\widehat{r_{i,j}}$ equals zero. The chosen action a'_j is the action with the maximum average reward $\widehat{r_{i,j}}$. If this action is not unique, the first action from A' that fulfils the condition is selected. To ensure that all actions have an equal chance to be selected, the order of A' is randomly generated at the beginning of the algorithm.
An exemplary application could consist of $S = \{coffee\}$, $A = \{serveMilk, serveSugar\}$ and $\beta = 10$. Furthermore, the rewards are constant, so that $r_{coffee,serveMilk} = 100$ and $r_{coffee,serveSugar} = -50$. In addition, it is assumed that $A' = A$. This is the case if the choice of an action solely relies on the rational decision part.
Let $p_{serveMilk}$ denote the probability that the action serveMilk is chosen. In this scenario serveMilk is clearly the favourable action. Without the rational selection the probability of serveMilk being chosen randomly is always 50 %.
Table 1 illustrates that the probability of choosing serveMilk is 100 % if any action has been chosen before. This is independent from the size of H_{coffee}, due to the constant rewards, as long as the history holds at least one entry. If the history is empty, the probability is still 50 %. On the one hand, if only entries for serveMilk exist in the history, the average reward for serveMilk equals 100 whereas that of serveSugar is estimated with 0. On the other hand if only serveSugar is listed in the history, its average reward is calculated as -50 and that of serveMilk with 0. So in both cases serveMilk is chosen with a probability of 100 %. If the history contains entries for both actions, the calculated average rewards resemble the realistic rewards and serveMilk is again chosen with a probability of 100 %. Compared to a probability of 50 %, if the actions are chosen randomly, this is a considerable improvement.
The above presented scenario assumes that $A = A'$. This gives rise to the question whether the emotional preselection is needed at all. First of all the emotional preselection is a filtering process to reduce the number of available actions. As the rational analysis can be complex this is a reasonable approach to meet real-time expectations. For instance, a rational analysis might include calculating the length of the way to a target, which can be more time-consuming than the overhead caused by the emotional preselection.
Furthermore, the results with and without the emotional preselection do not only differ in computing times. Without the emotional preselection, the algorithm might rationally choose an action, which would have been eliminated during the emotional preselection.

4.4 Orbitofrontal cortex and ventromedial prefrontal cortex

The boxes OFC and vmPFC are responsible for the storage and update of the somatic markers and all further values, which are part of the algorithm (see figure 9). One main function of the vmPFC is described as the comparison of several emotions on a unified basis (Vitay and Hamker, 2011). As this step is not necessary here, because all

Tab. 1. The probability that serveMilk is chosen depending on the history H_{coffee}; here the action serveMilk is abbreviated as a_1, serveSugar is given as a_2.

H_{coffee}	$\widehat{r_{1,1}}$	$\widehat{r_{1,2}}$	p_{a_1}
$\emptyset$	0	0	50 %
$\{(a_1, 100)\}$	100	0	100 %
$\{(a_2, -50)\}$	0	-50	100 %
$\{(a_2, -50), (a_1, 100)\}$	100	-50	100 %

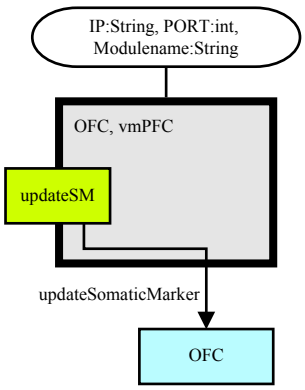

Fig. 9. The function of the orbitofrontal cortex and ventromedial prefrontal cortex is the update of the somatic marker and all other values for the framework, based on the obtained rewards.

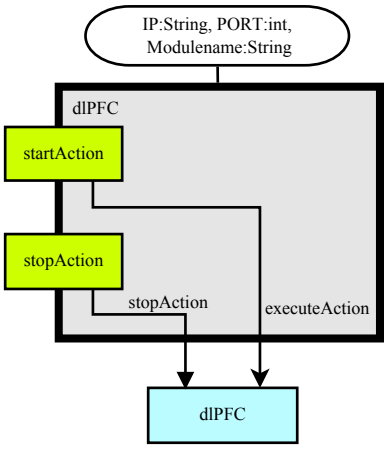

Fig. 10. The dorsolateral prefrontal cortex is the executor and is able to start and stop actions.

emotions and rewards are represented by numerical values, the vmPFC is merged with the OFC to one box. When a signal is received at updateSM, a function of the OFC module is called, in which all values are updated according to the computations (19).

$$OFC, vmPFC : I(updateSM[s_i, a_j, r_{i,j}]) \qquad (19)$$

4.5 Dorsolateral prefrontal cortex

After the decision making process, the final, chosen action is sent to the dlPFC (see figure 10), which is responsible for the execution of actions (20). When an action comes in while the robot is already executing one, the current action will be stopped and the robot strikes an initialisation pose in order to protect itself. When the initialisation pose is reached, the execution of the new action is started.
Furthermore, via the input stopAction a specific action can be stopped. After the action has been stopped, the robot returns to the initialisation pose as well.

$$dlPFC : I(startAction[a_j]) \qquad (20)$$

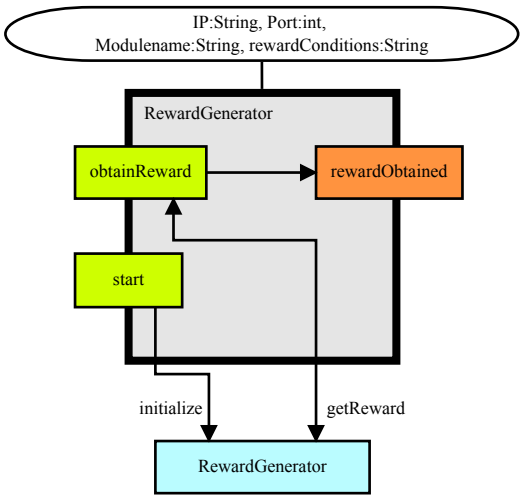

Fig. 11. The reward generator can create or receive rewards.

4.6 Reward generator

There are variant possibilities to generate rewards. In some cases, the rewards for different pairs of a stimulus and an action are known and fixed. Therefore, the information can be specified through the parameter *rewardConditions* and will be set when the input start receives a signal (see figure 11). Otherwise, rewards could come from the environment or internal states. In terms of human robot interaction, the rewards should be given by the interacting human. Therefore, the robot can be rewarded by touching one of the three tactile sensors on the head. Every sensor triggers a different kind of reward (negative, neutral or positive). Furthermore, it could also be interesting to use speech for rewarding purposes. Beside the possibility to give rewards through an external input, internal information could also be used. An example for that are the values of the gyroscope when the robot should be protected from falling.
No matter which implementation is used for giving rewards, the robot is able to obtain a reward when a signal is coming in through the input obtainReward (21).

$$\begin{aligned} RewardGenerator &: I(obtainReward[s_i, a_j]) \\ &\overset{Amygdala}{\rightarrow} \\ &O(rewardObtained[s_i, a_j, r_{i,j}]) \end{aligned} \qquad (21)$$

5. CASE EXAMPLE

5.1 Configuration of an application

Aiming at a fast adaption of the decision making framework to different applications, even for users without programming skills or technical expertise, the following example should demonstrate the required steps for the user. First of all the user has to decide in which situations the robot should show a reaction. Therefore the user has to define the set S. For this purpose the box SensorySystem (part of the sensory cortex) gets the names of the desired stimuli through a parameter (see figure 12). For the example, three different stimuli should be recognized (22).

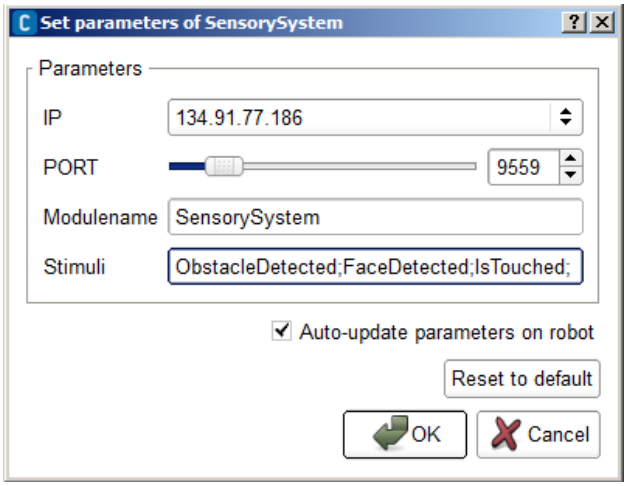

Fig. 12. Exemplary input of parameters of a box in the Choregraphe (here the specification of the stimuli in the box SensorySystem).

$$S = \{\underbrace{ObstacleDetected}_{s_1}, \underbrace{FaceDetected}_{s_2}, \underbrace{IsTouched}_{s_3}\} \tag{22}$$

After the definition of the stimuli, the user has to model the different stimuli by creating a CheckSensoryData box for each stimulus. While the configuration of some stimuli is more or less general, like the detection of a face, there can also be stimuli with individual differences. An example for this is the stimulus ObstacleDetected. While in some environments with big rooms it could make sense to detect an obstacle already when the distance is less than 50 cm, other environments with small rooms may require a much smaller distance to ensure that the robot is still able to complete tasks. The input parameters of a CheckSensoryData box for the stimulus ObstacleDetected could look like those given in table 2. For this and for all following input of parameters, the GUI of the Choregraphe is used as seen in figure 12. In the example, an obstacle is detected when the measurement of one of the sonar sensors is less than 50 cm or when one bumper at the robot's feet is pressed. When multiple conditions should be fulfilled, the parameter *NumberOfConditions* has to be increased. The variable *TimeInterval* ensures that the same stimulus can only be recognized every five seconds, otherwise it would continuously be send to the amygdala for decision making. This could lead to continued interruptions in the action execution. For each other stimulus, a separate CheckSensoryBox is created, which is using data provided by existing modules such as face detection.

Tab. 2. Possible parameters of the CheckSensoryData box for the stimulus ObstacleDetected.

Parameter	Example Value
Conditions	LeftBumperPressed,==,1; RightBumperPressed,==,1; SonarLeftDetected,<,0.5; SonarRightDetected,<,0.5;
EventName	ObstacleDetected
NumberOfConditions	1
TimeInterval	5

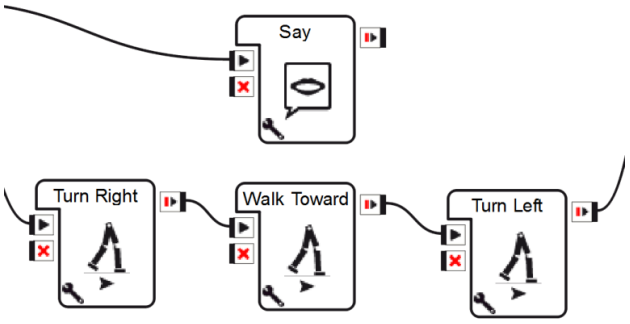

Fig. 13. Exemplary behaviour network of an action to evade from an object.

The last step is the definition of the set of actions A (see (23)). Each action is reflected by the name of a behaviour network created with the Choregraphe, which can be executed. An exemplary action EvadeObstacle (see figure 13) could consist of a command to turn right, walk toward and then turn left. All boxes used for moving are of the same type, but with different parameters and names. While the robot evades, it simultaneously gives an audio feedback to the user (box Say). Here, every box is part of the SDK, provided by Aldebaran. After the user has finished the creation, the network is saved under the specified name of the action (here EvadeObstacle). The name of each action, which should be available for the robot to execute, is provided through a variable in the ALMemory.

$$A = \{\underbrace{EvadeObstacle}_{a_1}, \underbrace{SayHello}_{a_2}, \underbrace{BowHead}_{a_3}\} \tag{23}$$

As the configuration is finished, one possible decision making cycle could proceed like follows: The robot explores the environment and an obstacle is detected, so that the amygdala receives a signal from the sensory cortex. Under the assumption that no innate behaviour exists, the amygdala relays the stimulus to the ventral BG (see (24)).

$$\begin{aligned} &Amygdala : I(stimulus[ObstacleDetected]) \\ &\qquad \overset{ventralBG}{\rightarrow} \\ &O(relayStimulus[ObstacleDetected]) \end{aligned} \tag{24}$$

In the next step the ventral BG receive the stimulus and perform the emotional selection based on the somatic markers. The output A' is sent to the rational analysis (25).

$$\begin{aligned} &VentralBG : I(stimulus[ObstacleDetected]) \\ &\qquad \overset{RationalAnalysis}{\rightarrow} \\ &O(subsetSelected[ObstacleDetected, \\ &\qquad \{EvadeObstacle, BowHead\}]) \end{aligned} \tag{25}$$

A single action is chosen by the rational analysis that is sent to the amygdala (26).

$$RationalAnalysis : I(select[ObstacleDetected, \{EvadeObstacle, BowHead\}]) \xrightarrow{Amygdala} O(actionSelected[ObstacleDetected, EvadeObstacle]) \quad (26)$$

When the amygdala receives the chosen action, it initiates the execution by sending the action to the dlPFC (27). Furthermore, the output triggerReward becomes activated, which enables the robot to obtain rewards (28).

$$Amygdala : I(action[ObstacleDetected, EvadeObstacle]) \xrightarrow{dlPFC} O(doActionR[EvadeObstacle]) \quad (27)$$

$$Amygdala : I(action[ObstacleDetected, EvadeObstacle]) \xrightarrow{RewardGenerator} O(triggerReward[ObstacleDetected, EvadeObstacle]) \quad (28)$$

The dlPFC receives the chosen action and starts its execution (29).

$$dlPFC : I(startAction[EvadeObstacle]) \quad (29)$$

While executing the action, the robot is able to obtain a reward. As evading seems to be a desired behaviour when an obstacle is detected, a positive reward is assumed here (30).

$$RewardGenerator : I(obtainReward[ObstacleDetected, EvadeObstacle]) \xrightarrow{Amygdala} O(rewardObtained[ObstacleDetected, EvadeObstacle, 50]) \quad (30)$$

The amygdala receives all information from the reward generator and relays them to the ventral BG (31). As the reward is positive, the ventral BG just send the information to the OFC and the vmPFC, in order to initiate the update of all values (32).

$$Amygdala : I(reward[ObstacleDetected, EvadeObstacle, 50]) \xrightarrow{ventralBG} O(relayReward[ObstacleDetected, EvadeObstacle, 50]) \quad (31)$$

$$VentralBG : I(reward[ObstacleDetected, EvadeObstacle, 50]) \xrightarrow{OFC,vmPFC} O(updateSomaticMarker[ObstacleDetected, EvadeObstacle, 50]) \quad (32)$$

Finally, the OFC and vmPFC receive all information to update the somatic marker and all other parameters, which are included in the emotional selection (33).

$$OFC, vmPFC : I(updateSM[ObstacleDetected, EvadeObstacle, 50]) \quad (33)$$

6. CONCLUSION

In this paper, improvements to a decision making approach based on artificial somatic markers and to the implementation architecture on the humanoid Nao-robot are presented. In order to comply with Damasio's theory, an example of the rational analysis part is shown. This added rational part chooses one action out of the subset, which is the result of the emotional selection part. The additional functionality also demonstrates the advantage of the chosen modular architecture, as the code of other modules remains unaffected.

In terms of giving even users without programming skills or technical expertise the possibility to create their own applications, all necessary steps can be arranged in a user-friendly graphical user interface, without the need to change any source code. This is a major factor to popularize robots in the (near) future. Of course there are applications, such as cleaning robots, in which a pre-configured robot is sufficient, but e.g. in cases of robot companions the desired behaviour could be heavily dependent on the owner and the field of applications. Furthermore, it is not excluded that the desired behaviour should change or that in the course of time new actions should be available. Due to those individual factors, it would involve immense costs for manufacturers to deliver a specified robot to each customer. This in turn would lead to higher prices for such robots, so that public access would be restricted.

Furthermore, the architecture could support variant fields of research, in which interdisciplinary skills are often necessary, such as human robot interaction. Additionally, the architecture could be used as a basis for other decision making approaches, especially for those which are based on the Somatic Marker Hypothesis, but also on variant psychological theories. It is also possible to refine this architecture for the artificial life domain, thus allowing more detailed brain activity simulations that include e.g. chemical processes. Certainly, a lot of major changes would be necessary, as the focus of the presented approach is mainly application-oriented.

Subsequent work should concentrate on different aspects. One aspect should be a further evaluation of the decision making algorithm with different tasks than the IGT. A further comparison of the behaviour of human subjects and the agent's behaviour, even in reversal learning tasks, would be interesting. Furthermore, studies on how humans perceive the behaviour of the agent would be helpful in order to improve the algorithm. A strived goal should also be the possibility of learning action sequences. Data fusion is also a main issue. Even if the composition of an action includes parallel processes, the framework treats the composition as an atomic action, which can only be executed individually. However, a parallel execution or a fusion of multiple actions can be expedient.

REFERENCES

Aldebaran Robotics (2012). *NAO Software 1.12.5 documentation*.

Bechara, A., Damasio, A., Damasio, H., and Anderson, S. (1994). Insensitivity to future consequences following damage to human prefrontal cortex. *Cognition*, 50, 7–15.

Damasio, A. (1994). *Descartes Error: Emotion, Reason, and the Human Brain.* Putnam Press.

Dautenhahn, K., Woods, S., Kaouri, C., Walters, M.L., Koay, K.L., and Werry, I. (2005). What is a robot companion - friend, assistant or butler. In *In Proc. IEEE IROS*, 1488–1493.

Hoefinghoff, J. and Pauli, J. (2012). Decision making based on somatic markers. In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*, 163–168. AAAI Press.

Hoefinghoff, J., Steinert, L., and Pauli, J. (2012). Implementation of a decision making algorithm based on somatic markers on the nao robot. In *Autonomous Mobile Systems 2012*, Informatik Aktuell, 69–77. Springer.

Vitay, J. and Hamker, F. (2011). A neuroscientific view on the role of emotions in behaving cognitive agents. *KI - Kuenstliche Intelligenz*, 25, 235–244.