Multicriterial Decision-making Control of the Robot Soccer Team

Petr Tucnik
University of Hradec Kralove
Czech Republic

1. Introduction

This text is a summarization of a series of shorter articles published about this topic (and related problems) during the years 2005-2007 (E.g. Tucnik et al. 2006a, Tucnik et al. 2006b). In previous articles we have been commonly using the term "agent" when referring to the MiroSot robots. We will continue in this practice in this text as well. We believe that there is no conflict in using this term because the idea of autonomous entity (in the way the agent is usually defined, see below) is well-corresponding with the role of the MiroSot player. Therefore, whenever the term of "agent" or "robot" or "player" is used, we are referring to the same entity. Any exceptions will be explicitly mentioned.

The principles of the robot soccer game are well-known, but it is usually a good idea to begin with the basics. Comprehensive description of all rules and specifications is available at www.fira.net (FIRA – Federation of International Robot-soccer Association). Therefore, only basic information will be provided, when we feel it is necessary or important. We will be dealing with the MiroSot (micro-robot soccer tournament) Middle League (5vs5 players). This league of the tournament is played on the pitch of the size of 220cm x 180cm, robots must be smaller than 7.5cm x 7.5cm x 7.5cm and the robot's weight must not exceed 650g.

Together with other categories of the robot soccer (HuroCup, KheperaSot, NaroSot, AndroSot, RoboSot and SimuroSot, for details see the same webpage as in the above paragraph), the MiroSot league also serves a certain purpose: *"MiroSot initiative gives a good arena for multi-agent research, dealing with research subjects such as cooperation protocol by distributed control, effective communication and fault tolerance, while having efficiency of cooperation, adaptation, robustness and being real-time."* (citation taken from http://www.fira.net/about/overview.html, date 20th July 2007). We will try to follow this aim, as it was defined by FIRA, in our work. Illustrational photograph of the MiroSot robot is presented at the Fig. 1. During the game, there is always a coloured patch on the topside of the robot. This colour patch will identify the robots in the team.

This chapter will be focused on the control mechanism and decision-making issues only. We are not interested in image processing or data transmission problems. We have decided that it is no purpose of this text to discuss these matters. Therefore, we will dare to assume that these parts of the system work without any errors or malfunctions and we are obtaining all the data needed.

Source: Robotic Soccer, Book edited by: Pedro Lima, ISBN 978-3-902613-21-9, pp. 598, December 2007, Itech Education and Publishing, Vienna, Austria

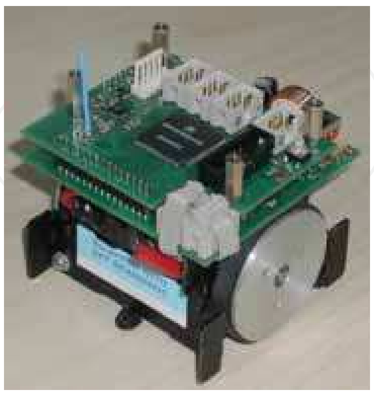


Fig. 1. A MiroSot robot

For the both strategic (team-oriented) and tactical (individual) control of the team, we have decided to implement the multicriterial decision-making (MDM) principle. The more common approach to the robot soccer problem may be found in (Kim et al., 2004). The multicriterial decision-making approach has an excellent adaptability potential but requires perception of the whole game and its environment from another perspective. From the agent's point of view, the environment is perceived as a set of (more or less important, see below) influences and the agent is continually trying to find the most appropriate, the most profitable reaction to the given situation. The description of the MDM algorithm is presented in the third part of this chapter.

2. Terminology and Basic Definitions

The use of multicriterial approach requires a definition of terms that will be used during the explanation of its principles. Explanation of the general MDM algorithm will be provided here (its application will be presented for better understanding on the goalkeeper example in the part 9).
The time given for the decision-making process (DMP) is strictly limited by the time of image processing speed. Generally, this is the most time-demanding part of the whole decision-making process. Therefore, all time-subordinate tasks have to be done in the given time frame. The image processing speed is quoted in frames per second (FPS) and the higher this value is, the better. It may happen that it is a team that has a better reaction time which wins over the team with better strategy, because it is capable of faster reactions. It is an utmost necessity to try to keep image processing speed as fast as possible. The time frame given by the image processing speed and cycle of activities performed during the decision-making process together define the **iteration of the game**. In other words, the iteration is a complete set of activities beginning with image processing tasks, continuing with situation analysis, action, role and task assignment and instruction processing and ending with the transmission of the instructions to the robot, all in the given time frame. E.g. when we have reached the speed of 50FPS in image processing tasks, then we have to be able to make all strategic and tactical decisions in 1/50 [sec] to keep up pace with the rest of the system. Any delay would cause a serious trouble.

2.1 Attributes

The environment is perceived by the agent as a finite set of attributes. The **attribute** is a numerically represented and limited (see below) description of the measure of presence of a certain characteristic of the environment. E.g. a position of the agent on the playground's grid is an attribute of the environment. However, the attributes' domain is not limited to the "tangible" information only. Any information that has a descriptive value may be perceived as an attribute, if we are able to determine its highest and lowest value possible. For analysis purposes, it may be useful to divide the attributes into four groups, according to their descriptive domain, as it is presented at the Fig. 2.

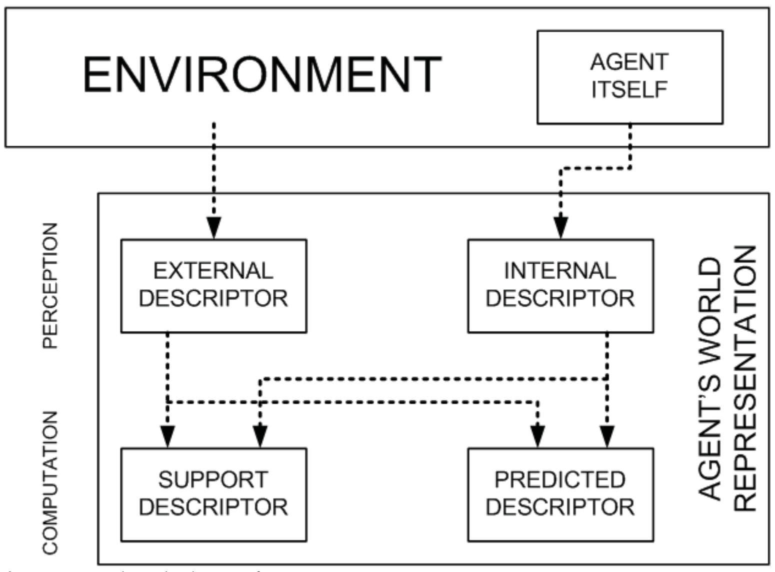


Fig. 2. Attributes are divided into four groups

The agent is an integral part of the environment for both itself and other agents. Other agents are taken as a part of the environment. The agent's representation of the world is formed by the four indicated types of descriptors. When the agent is obtaining an attribute's value, either perception (sensors) or computation is used during the process.

The **external descriptors** are attributes used to describe characteristics of the real-world environment. E.g. speed of other robots, position of the ball (or robot) on the playground, etc. This type of information provides the cornerstone for DMP for the derived descriptors are based on it. During the game, this type of information has to be *reliably* provided by the image processing module (see part 5 – Strategic Control Module). In other words, this is the reliable base for decisions to be made. If we are unable to satisfy the reliability requirement and do not have a set of solid facts which we may use for decision-making, than all decision-making effort is futile. The "real-world environment" expression, used at the beginning of this paragraph, is valid for the environment of the MiroSot game. For the purposes of SimuroSot, the formulation "software environment" should be used and so on, depending on the problem domain.

The **internal descriptors** are used to describe internal states of the agent. E.g. velocities of its wheels, battery level, etc. For our problem domain, they usually have only limited impact on the DMP. E.g. the velocities of the robot's wheels are, in fact, output of the DMP. Also, the other internal characteristics of the agent are designed to fulfil the needs of the robot soccer game perfectly and therefore are not so important. But they may play an important role when we are trying to apply this decision-making method to another type of problem (other than robot soccer). In certain cases, the internal states of the agent may have a critical decision-making importance. This type of attributes has to be *reliable* as well.

The **support descriptors** are based on the real-world information (both external and internal nature) and are derived from it. Support descriptors are category of attributes computed on the basis of reliable data from the environment and agent itself. They are used for supplemental calculations (if they are needed) when planning movement, collisions, turning, etc. The realization of many tasks depends on these support calculations.

The **predicted descriptors** are used to take the effect of the action into account when making decision or planning. The predicted position of the robot is an example of this type of descriptor. This type of (predicted) information is, in general, *unreliable* ("Is unreliable" or "may be unreliable" – it is depending on the given attribute. Own actions, for instance, may be predicted precisely, actions of opponent's agents very imprecisely, as we have no way how to affect them directly.) to a certain degree, but is important for making decisions, that needs to reflect possible impact of an action. Prediction has gradually lower reliability as we try to predict further in future. When the speed of the game is taken into account, it is reasonable to predict up to <u>maximum</u> (rather less) time of approximately 1-2 [sec] (depending on the both team's capabilities). Prediction of more distant future is useless and may lead to incorrect conclusions in the DMP. Precise time interval of applicable prediction is a question of extensive testing and image processing speed.
It is important to stress up the fact that we are always dealing with a certain degree of imprecision in the real-world cases. This is caused by the imprecision of our sensors used for monitoring of attributes, wrong calibrations, rounding of numbers in calculations and other undesirable influences. There are several major sources of uncertainty in the environment (Decker, 1995): (1) uncertainty of the environment – monitoring of environment by sensors is made with imprecision. It may happen that the agent is unable to recognize the ball or other players on the pitch for a moment; (2) uncertainty of other agents – the opponent's agents next actions are not certain (also it is possible that hardware malfunction occurs in the own team); (3) uncertainty of actions – the actuator apparatus of the agent may fail to perform the action as planned.
Nevertheless, it is possible to keep these effects in reasonable boundaries. Detection of failures and consequent actions has to be implemented in the decision-making mechanism. However, for the purposes of this text we will avoid further discussion about the failure detection and handling, for these are complicated problems unrelated to the decision-making and control issues which are our main aim. Much useful information regarding the topic of failure detection and solving may be found in (Aguilera et al., 1997; Byrne & Edwards, 1996; Kit, 1995; Menzies, 1999; Tichý, 2003).
All four types of the descriptors are only special types of attributes. The agent's internal concept of the world is formed by all four types of attributes. This separation into groups is

useful for analysis purposes only and apart from the reliability of the information value there is no other influence on the DMP. The whole set of all attributes A used in the DMP may be formally described:

$$A = (a_1,...,a_z),\ \forall a_i \in A : Dom(a_i) \in \langle a_{i\min}, a_{i\max} \rangle, -\infty < a_{i\min} < a_{i\max} < \infty \tag{1}$$

A is a set of all attributes and every attribute has a finite minimal and maximal value. Every attribute is normalized before it is used in DMP. The following formula is used for normalization:

$$norm(a_i) = \left| \frac{a_i - a_{i\min}}{a_{i\max} - a_{i\min}} \right|,\ a_i, a_{i\min}, a_{i\max} \in R \tag{2}$$

After application of the *norm* function for each attribute, a following is valid:

$$norm(a_i) \in \langle 0;1 \rangle \tag{3}$$

Therefore, a comparison of attributes of different units of measurement is possible. After normalization, no measurement units are used and normalized attribute value is just a number.

2.2 Environment

The definition of the attribute and its properties was discussed in the part 2.1. The environment is defined by a limited set of attributes. The whole environment has to be perceptible by an agent (in case of derived descriptors, the agent has to be able to compute them). Attributes that are not perceptible by an agent may be omitted for they have no influence on the DMP. From the agent's point of view, the environment E may be formally defined as:

$$E = (norm(A)) \tag{4}$$

Therefore, the agent is working with normalized attributes when making a decision. Although the set of all attributes A was defined by (1), we present following specification for better understanding (these sub-sets are representing types of descriptors):

$$A = A_{EXTERNAL} \cup A_{INTERNAL} \cup A_{SUPPORT} \cup A_{PREDICTED} \tag{5}$$

An important role in the DMP plays time. The temporal aspect of DMP may not be omitted for the effects of the actions have to be taken into account. For these purposes a **configuration** C is defined by the following formula:

$$C^T = \left(a_1^T, \ldots, a_z^T\right) \tag{6}$$

T stands for the game iteration (time) index. The configuration C of the game represents a state of all attributes (their values) in the given moment. The time index will always be positioned in superscript position in the following text.

Note: A total number of game iterations may be calculated from the game length and video processing speed. E.g. for 10 [min] of the game and image recognition speed of 50 [FPS], we have a 10 x 60 x 50 = 3000 iterations of the game. This is a 3000 potential decision-making moments, although in many iterations there will be no decision-making at all (agents will be processing designated tasks). Important is that the number of iterations is always finite and data about the game and decision-making process should be stored in some kind of log or data storage medium for the game analysis after the match.

2.3 Definition of Agent

The universal definition of an agent, according to (Ferber, 1999), is following:

"An agent is a physical or virtual entity
(a) which is capable of acting in an environment,
(b) which can communicate directly with other agents,
(c) which is driven by a set of tendencies (in the form of individual objectives or of a satisfaction/survival function which it tries to optimize),
(d) which possesses resources on its own,
(e) which is capable of perceiving its environment (but to a limited extent),
(f) which has only a partial representation of this environment (and perhaps none at all),
(g) which possesses skills and can offer services,
(h) which may be able to reproduce itself,
(i) whose behaviour tends towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representations and the communications it receives."

In the case of the MDM agent (agents that are using MDM algorithm for decision-making), the main deviations from this universal definition are in (e) and (f). The MDM agent possesses all the information about the environment that it needs to make a qualified decision. This is both necessary and sufficient condition that must be fulfilled in order to function properly. All the information types are defined in the design phase of the agent's development process and there should be no other information needed. Otherwise, it is a design flaw.
The definition mentioned above is very universal and very general and it is not very suitable for our purposes. We would like to present a more specific (formal) definition of the agent. We will proceed from the formalized definition of the (reactive) agent as it is presented in (Wooldridge, 2002; Genesereth & Nilsson, 1987; Kelemen, 2001; Kubik, 2004), but some significant changes have to be made in agent's formalized definition for the MDM-based agent. The reactive agent basis will be sufficient enough at this moment because we will try

to keep it as simple as possible for comprehensibility purposes. Further extension of this definition is possible to reflect social aspect of the game. Following definition is translated specifically from (Kubik, 2004), but similar versions are also present in other sources (Wooldridge, 2002; Genesereth & Nilsson, 1987; Kelemen, 2001) (for perception P and environment E):

$$percept : E \rightarrow P \tag{7}$$

"Perception of the agent has immediate effect on its actual state:"

$$change\_of\_state : P \times I \rightarrow I \tag{8}$$

"Every action from the set of actions A, which is agent able to perform, the agent performs on the basis of application of the perception function on the actual internal state of the set I (set of all states of the environment)*:"*

$$action : P \times I \rightarrow A \tag{9}$$

"Apart from the change of internal state have actions also impact on the environment:"

$$impact : A \times E \rightarrow E \tag{10}$$

"The agent is a 6-tuple:"

$$\{P, A, I, percept, change\_of\_state, action\} \tag{11}$$

*"The set of goals of the agent C is a subset of the set I. The goal may be characterized as a **maintaining goal** and may be theoretically running infinitely (to be in a state of readiness and continual functionality), or **reaching goal**, which is defined by a state which achieving is characterized as a goal accomplishment."*

This is the end of the definition taken from (Kubik, 2004).

For the purposes of the MDM implementation, the cited definition has to be modified.

The MDM-oriented agent is a 6-tuple:

$$\{C, S, P, T, V, W\} \tag{12}$$

Where:
C ... set of game configurations (defined by (6)),

S ... set of internal states of the agent (definition is following),

P ... set of perception functions (definition is following),
T ... transition relation for transition between states (definition is following),
V ... set of all possible variants of solutions (definition is following),
W ... set of weight functions (definition is following).

In the case of the MDM-based agent, the goal orientation of the agent is specified by its actual state. **State** is, in fact, an algorithm, performing certain operations in the environment, but it is more useful to use the abstraction of the term "*state*" for easier understanding. Thus, whenever the "action" or "state" of the agent is mentioned, we are referring to the same thing.

The **set of states** S contains two subsets S_S and S_T. Thus, $S = S_S \cup S_T$**.** The **set of stable states** S_S contains maintaining goals (continual functionality of an agent). The agent is remaining in the stable state and there are two situations which may force him to change it: (a) an exceptional situation occurred – the agent has to take actions leading to correction of the exceptional state of the environment in order to be able to be in its stable state again; (b) the game situation requires agent to change its basic behavioural pattern and to do something else (E.g. the agent in the role of an attacker was waiting for the pass and as soon as he gets the ball under its control, its is trying to score a goal). In the case (a), the agent is making a transition to the temporal state, in the case (b) to another stable state. It is important to remain as much as possible in the stable states for these are defining functional behaviour leading to the highest benefit of the team.

The **system of sets of temporal states** S_T is containing states used to handle exceptional situation which is disrupting standard behavioural pattern, defined by the stable state. There is a special type of dependency of temporal states on stable states. Formally, it may be defined this way:

For every agent, there exists the set of stable states $S_S = \{s_{S1}, \ldots, s_{Sn}\}$ and the system of sets $S_T = \{S_{T1}, \ldots, S_{Tm}\}$. $\forall s_{Si} \in S_S$ has assigned a subset $S_{Ti} \subseteq S_T$. For $\forall s_{Tj} \in S_{Ti}$ is (s_{Si}, s_{Tj}) an ordered tuple. $\forall S_{Ti} \in S_T : \bigcup_{i=1}^{m} S_{Ti} = S_T$ and subsets $S_{Ti} \subseteq S_T, S_{Tk} \subseteq S_T$ are pairwise disjoint. In other words, each stable state has assigned a set of temporal states (it may be even an empty set). For better understanding, Fig. 3 is representing this situation.

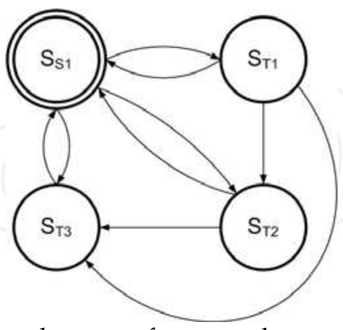


Fig. 3. Stable state has assigned a set of temporal states used to handle exceptional conditions that may occur

Another part of the agent is the **set of perception functions** P. P contains functions $p_j \in P$ representing agent's sensors. The definition from (Ferber, 1999), mentioned at the beginning of the part 2.3, is describing agent's definition of the world in this way: *"...the agent (e) is capable of perceiving its environment (but to a limited extent) and (f) has only a partial representation of this environment (and perhaps none at all)"*. There is an important difference in the MDM-based agent case. The MDM-agent has all the information needed for making the decision. Because the environment $E = norm(A)$ is from the agent's point of view all the information needed (fulfilment of this condition is a matter of proper design and testing), then following expression must be valid for the agent in order to be able to function properly:

$$\forall a_j \in A, j \in (1,...,z) \exists p_j \left| p_j(a_j) = norm(a_j) \right. \tag{13}$$

In other words, all attributes of the environment are monitored by perception functions. Following expression is valid for $p_j \in P$:

$$\forall p_j \in P : Dom(p_j) \in \langle k, l \rangle \left| -\infty < k < l < \infty \right. \tag{14}$$

$$\forall p_j \in P : \mathrm{Im}(p_j) \in \langle 0,1 \rangle \tag{15}$$

Thus, every percept a_j is normalized to interval $\langle 0;1 \rangle$.

The **transition relation** T is used to set transitions between states. At the Fig. 4 is shown the situation of the stable state with maximum number of transitions for temporal states and two transitions to other stable states from the state S_{S1}. Because the transitions between temporal states themselves are not so important (after finishing the behaviour in temporal state, the agent continues its functioning in another state), we are interested in a transition index T_i, which is a number of possible ways beginning in given stable state S_{Si} and ending either in itself or another stable state. Transition relation must follow these restrictions: (a) There must be no cycles between temporal states, (b) the state must not have a transition directly leading to itself, (c) total maximum possible number of ways how to solve the situation (every transition is a reaction to the configuration in the given moment) is represented by $complicatedness$ value, given by the formula:

$$complicatedness(s_{Si}) = (2^a - 1) + b, s_{Si} \in S_S \tag{16}$$

Where a is a number of temporal states related to the given state s_{Si} in the means of definition in the part regarding the system of sets of temporal states S_T (see above). b represents a total number of stable states which are accessible from s_{Si}. This is important because we are able to specify total computational capacity needed for considering reactions (making prediction computations during the DMP).

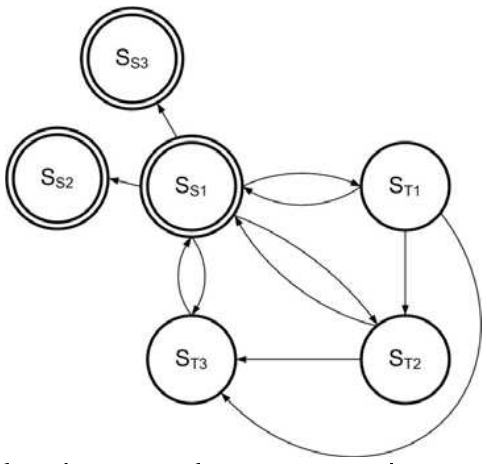


Fig. 4. A maximum number of transitions between states is finite

The three restrictions mentioned in the above paragraph specify the characteristics of the T relation: (a) T is not symmetric, (b) T is not reflexive, (c) T is transitive.

The **set of variants of possible solutions** V contains s-tuples of attributes subsistent to every transition between two states. Formally:

$$\forall s_i, s_j : s_i \xrightarrow{T} s_j \big| \exists v_r \in V : v_r = (a_1, ..., a_s), a_i \in A, i = (1, ..., s), r, s \in N \tag{17}$$

Elements of the set V are used for the calculation of convenience (see below).

The **set of weight functions** W contains weight functions assigned to the each attribute of every $v_r \in V$. Formally:

$$\forall a_i \in A : \exists v_r \in V : v_r = (a_1, \ldots, a_s) | \exists w_i \in W : w_i(a_i) \tag{18}$$

For further explanation of weight functions, see part 3.2: "Attributes and Weight Functions".

2.4 Other Agent's Characteristics

Agent was defined in the part 2.3. In this part we would like to present some other characteristics and design conditions in addition to this definition.

For every state $s \in S$ must be true, that it has at least one transition defined:

$$\forall s_i \in S : |V_i| \geq 1 \tag{19}$$

In this case:

$$\exists s_i \in S : |V_i| = 0 \tag{20}$$

There is a danger of deadlock, because there is a state with no transition defined. Agent would not be able to do anything else.

3. Principle of the MDM, Attributes and Weight Functions

Attributes describing environment must have certain properties. Above all, they must be numerically represented. Issues regarding attributes´ design are discussed in this part. Examples of attributes and their weight functions will be provided here.
There are some limitations in the attribute design process. Following conditions must be fulfilled before the attribute may be implemented in the system: (a) attribute must be numerically represented, (b) attribute must be limited (see (1)) and (c) the agent must be able to obtain the attribute's value from the environment (see (13)). The (a) condition is the most restrictive, because many influences are difficult to be represented numerically. In human decision-making, for instance, the emotions play an important role. Although it may be possible to implement it somehow, it is obvious that there is a big problem for analysis and design and the solution will always be very questionable.
For many influences we would like to implement in the system and which have a problem with the numerical representation, there exists an emergency solution. We may use techniques used for fuzzy expert systems definition (resp. fuzzy sets definition) where we use the subjective division of linguistic variables into sets, which returns a numerical value for the variable. Still, the subjective nature of this approach brings some degree of unreliability into decision-making process because we may never be sure if the subjective

definition was made properly. However, the most of the attributes is usually of "technical" nature and numerical representation is not very problematic issue, fortunately.

3.1 Principle of Multicriterial Decision-making

The MDM is used on two levels in the robot soccer game: (a) individual agent's decisions, (b) team behaviour decisions. However, the principle is same in both situations. The following formula is used to compute the convenience value for any solution v_r (this formula is an adaptation of standard formulas used in multicriterial decision-making, presented in (Ramik, 1999; Fiala et al., 1997)):

$$convenience(v_r^T) = \sum_{i=1}^{s} w(norm(a_i^{T+x})), v_r \in V, a_i \in A, i = (1,...,s), r, s \in N \tag{21}$$

Where x is time needed to perform the action v_r and we are considering the game situation in a certain time moment T. The w represents a weight function, assigning to each attribute its importance and it is described in the next paragraph.

Before the attribute $a_i, a_i \in A$ is evaluated and we obtain its convenience value, the weight function $w_i(a_i), i \in (1,..., z)$ must be applied on it. Thus, every attribute in the environment has at least one weight function assigned. One attribute may have assigned more than one weight function depending on the number of variants it participates in. The utility of usage of the weight function is in the setting of attribute importance value for the DMP and decisions are reflecting reality more accurately. All attributes taken into account during the decision-making are supposed to have some influence on it. However, the one attribute's importance is not equal to another. Therefore, the application of the weight functions may not be omitted.

The formula (21) is the most critical for the MDM-agent because it represents the most fundamental basics of the multicriterial decision-making principle.

3.2 Attributes and Weight Functions

The form of the weight function is depending on the ideal state of environment we are trying to achieve and on the extent of tolerance for the difference between the actual and ideal state. In many cases, the ideal state is only a theoretical abstraction, but it is providing a reasonable orientation for the agent in the means of what is good and what is not good for it.
Example: The ideal would be to move to the desired position as fast as possible. Therefore, the ideal solution is consuming neither time nor energy. This is, of course, physically impossible, but the agent is trying to get as close to is as it is possible, saving as much time and energy as it can.

There is also a difference whether the agent is trying to process an action and reach the desired state of the environment or whether it is trying to remain in its state except when the exception occurs. The former case is typical for the temporal states of the agent (see the definition in the text above); the latter is typical for the stable states, where the agent is trying to perform its behaviour as long as possible. For better understanding, see Fig. 5.

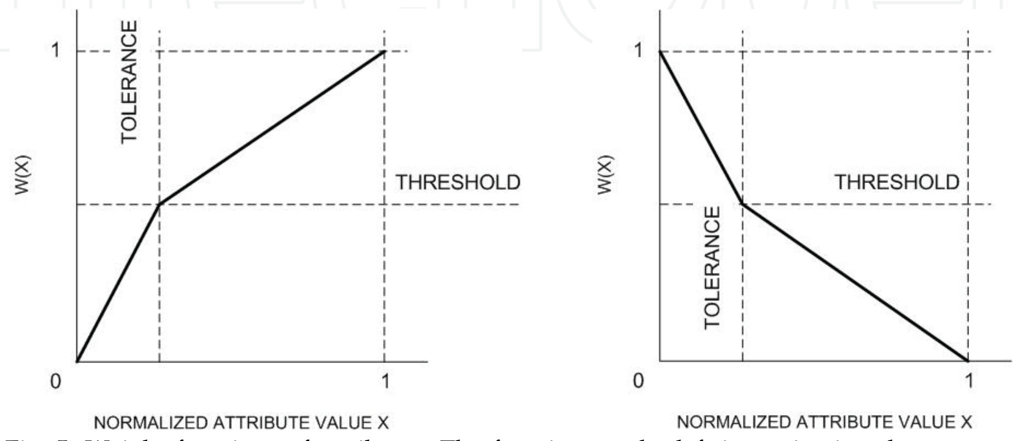


Fig. 5. Weight functions of attributes. The function on the left is motivating the agent to change its stable state to solve the exceptional situation, the function on the right is representing agent in the temporal state, trying to change the attribute's value to the tolerable level

For each type of the state, a threshold value is set. In the case of stable states, the threshold is used to ignore minor differences between the actual and ideal state. In other words, percept may be not intensive enough for the agent to be reasonable to react to it. On the other hand, there are also temporal states which are used to solve the exceptional situations. We need to know when the state of the environment is suitable enough to continue in the stable state algorithm processing. In both cases, the threshold value is assigned to the state.
In the part 9 of this text, the case study describing behaviour of the goalkeeper agent is presented. Further explanation of problems opened in this part will be provided there.

4. Environment Description and Attribute Composition and Decomposition

The proper description of the environment is essential for the function of the MDM control. The agent-environment interaction has special requirements that have to be met and such matters are discussed at the end of this part.
The composition and decomposition of attributes is important in the analysis phase of the development process. It has no influence on the decision-making itself. In fact, the decision-making should be giving the same results after application of composition/decomposition technique. The benefit is in the easier understanding of the environment for the human operators because they may use any level of the differentiation as they see fit.

4.1 Agent - Environment Interaction

The range of the agent's actuators is limited. Usually, agent is not able to change all attributes of its environment. This is generally valid for all agents in the real-world situation. E.g. the human is able to perceive much more than he is able to influence directly. The situation is similar in the robot soccer environment. The way it works is described on the Fig. 6:

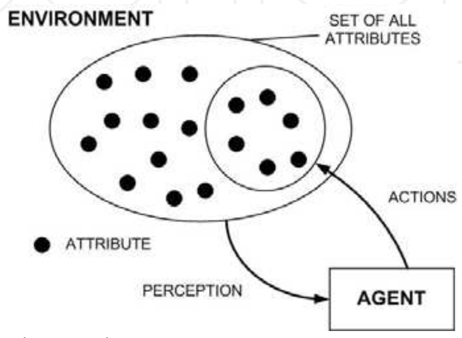


Fig. 6. Agent-environment interaction

The set of attributes (that we are able to identify and define) is often not providing description of the environment with sufficient precision. On the other hand, many minor influences are forming a significant power together, which is important for the decision-making process only as a sum of these negligible influences.
The solution to problem how to get a description of the environment that suits our purposes is represented in a technique of **composition and decomposition of attributes**. It is common that the final value of an attribute is obtained as a sum of several less important attributes. While using the MDM principle, we are able to regulate the description for our purposes while maintaining its descriptive power.

4.2 Composition and Decomposition of Attributes

Let us begin with an example which is not connected with the robot soccer problem. We believe that this example is well-suitable for the explanation purposes. Let us consider the environment of our agent described by just one attribute "FINAL PRICE" which does not provide as much descriptive potential as we need. We would like to be able to work with it, but with more precision than it is now possible. We know that the "FINAL PRICE" is composed of two constituents "COST" and "PROFIT", as you can see at the Fig. 7.

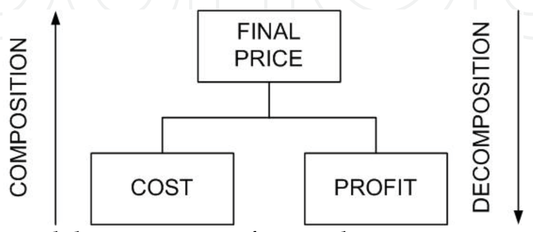


Fig. 7. The composition and decomposition of an attribute

Let us leave aside the exact participation of lower-level attributes on the upper-level attribute's value. It is simply composed of two constituents of lower-level of description. The lower level is more precise. It is same the other way. General situation is presented at the Fig. 8:

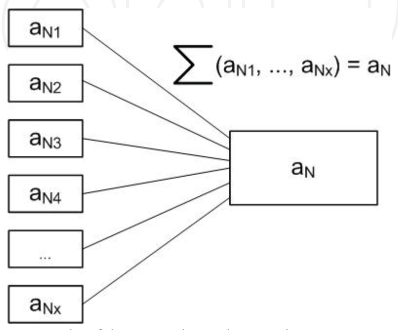


Fig. 8. An attribute a_N is composed of lower-level attributes a_{N1}, ..., a_{Nx}.

The variable $a_N \in A$ is any attribute of the environment and it is a composition of constituents $a_{Ni}, i = (1,...,x)$. In general, we may use the decomposition/composition as we see fit, but there are two restrictions:

$$w(a_N) = \sum_{i=1}^{x} w(a_{Ni}) \tag{22}$$

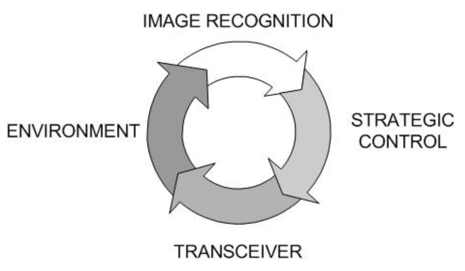
The weight of an attribute must remain the same as it was before the decomposition and:

$$a_N \xrightarrow{DECOMPOSE} (a_{N1},...,a_{Nx}) \xrightarrow{COMPOSE} a_N \tag{23}$$

The process of decomposition may be inverted by the process of composition without any change (in the means of empiric or descriptive value).

5. Strategic Control Module

The schematic representation of the strategy control module (and the rest of the system) is presented in this part of the chapter. It is necessary to perceive control mechanism as a part of a whole. The whole system controlling the multi-agent team must act co-ordinately.
Basically, only three modules are needed to control the robot soccer team. The information needed for the decision-making process are provided by the video recognition module, the control module is responsible for decision-making tasks and transceiver module is sending instructions to the robots on the pitch. The work of the first and third module is automated; the control module's activity is depending on the complexity of the situation on the playground. The basic scheme described above is represented on the Fig. 9.



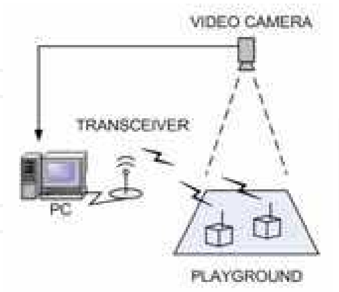


Fig. 9. An information-flow circle (on the left) and the decision-making cycle (on the right)

However, the situation is not that simple. There is a whole sequence of activities in the strategic control step, which is most important to us. More detailed description of the control process' communication is described in the part 5.1. of this chapter. Explanation of the strategic module functioning is in the part 5.2.

5.1 Communication

The system is communicating according to the following scheme (Fig. 10):

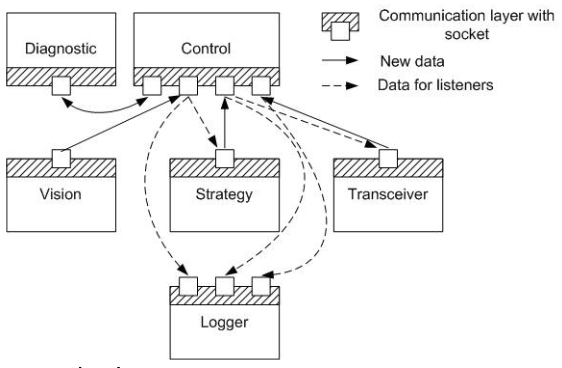


Fig. 10. Scheme of communication

The architecture client-server is used for communication. There is only one server in the system, represented by the "control" module. Clients are not communicating directly, but through the "control" module only. Clients are connected to the "control" module and set into listening mode. Information obtained by the "control" module is distributed to the respective modules. This is providing the necessary data-flow inside the system. As the system is divided into separated modules, the physical location of them is no longer of any interest to us due to the use of socket communication. When the dedicated network is used, we are not limited by the other form of data traffic and the control system is able to fully utilize the computational potential of the terminal it is running on.
Every module uses obtained data and is attaching own part of information to it. Exception is the transceiver module which is sending data on the field without any additional information.

The two parts of the system are not necessary: The diagnostic module and the **logger module**. For the latter, it was already mentioned that it is useful for us to have some kind of feedback from the system for subsequent analysis of the game. The **diagnostic module** is monitoring communication flow and in the case of any error, the human operator is informed. However, these two parts are unnecessary for the decision-making process and have only a support role.
The decision-making cycle is formed by the following sequence of activities: (1) the image from the video camera is processed and positions of all players and of the ball are recognized; (2) the strategic module is using this information to compute all the remaining attributes needed to make a decision and appropriate reaction is decided upon them; (3) the output of the strategic module – instructions for the robots – is send by a transceiver module to all players; (4) new information is being obtained from the game by the video camera. The situation is shown at the Fig. 9.

5.2 Strategic Control Module

From our point of view, the most important part of the system is the strategic control module. It was mentioned above that there is a sequence of activities present, as it is shown at the Fig. 11.

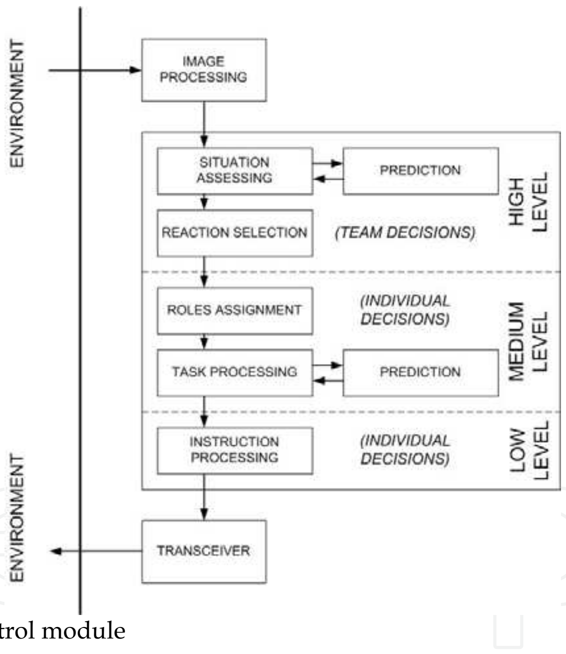


Fig. 11. Strategy control module

There are three levels of decision-making present. At the **high level**, the game configuration recognition is made in the situation assessing step. In other words, the system is trying to recognize any known pattern of behaviour and react to it. The most important attribute of the game is the possession of the ball. Whoever controls the ball controls the game. Accordingly, the set of all possible strategies is divided into three subsets: (1) offensive strategies (our team has the ball); (2) defensive strategies (opponent has the ball); (3) conflict strategies (no one has the ball or the possession of it is controversial – both teams have players very close to it). The most appropriate reaction is selected by the MDM, while the attributes are calculated or predicted as needed in order to be able to make a qualified decision. When the reaction is selected, every agent has basic conception of its behaviour in the next step of the game. The decision-making process is continuing at the **medium level**. At this level, the individual adaptations are made to the assigned behaviour. The agent has assigned a role, which is determining its coordinating position (if it is subordinate to the other agent or not). The specification of a reaction is vague and it must always be adjusted to the actual situation. Again, as the agent is adapting the parameters of the task to its actual situation, it may predict the course of actions as needed. At the **low level**, there is in fact no decision-making at all. The action is divided into atomic operations and these are just adjusted to the physical limitation of the robot's kinetic apparatus (we need to avoid slipping, falling, going into a skid, etc.). These atomic operations adjusted to physical parameters of the environment are called **instructions**. Instructions are, in fact, nothing more than commands used to setting the wheels velocities of the agent. This is the output of the strategic module and the instructions are send by a transceiver to all respective agents on the pitch.

5.3 Prediction

As it was mentioned before, the agents are using a prediction to approximate the future development of the game. This is not unusual; the systems dealing with the robot soccer problem are often using some kind of simulation mechanism for possible solution predictions, computations or verifications. The most important is prediction of the movement of the opponent's agents (We do not need to predict movement of our agents; we have this information stored in the memory. Therefore, for our agents, the prediction is used only for evaluation of possible situations or computing positions in coming iterations.). The situation is shown at the Fig. 12.

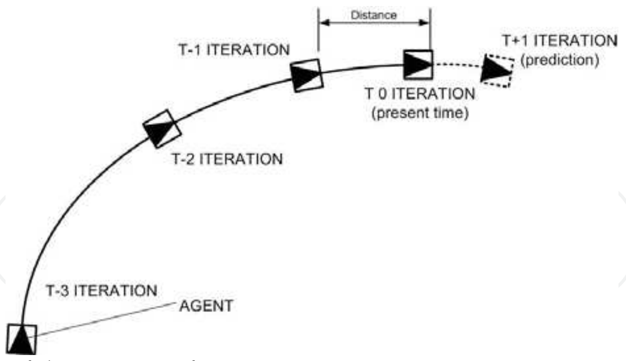


Fig. 12. Prediction of the position of an agent

We are predicting future position from the previous iterations of the game (three or four previous iterations are providing enough information to predict position in the next iteration of the game). For the illustration purposes, we present general for of formulas primarily used for the quadratic extrapolation:

$$x(t) = x_0.l_0(t) + x_1.l_1(t) + x_2.l_2(t), \tag{24}$$

$$y(t) = y_0.l_0(t) + y_1.l_1(t) + y_2.l_2(t), \tag{25}$$

For calculation of $l_0(t)$, $l_1(t)$ and $l_2(t)$ are used following formulas:

$$l_0(t) = \frac{(t - t_1)(t - t_2)}{(t_0 - t_1)(t_0 - t_2)} \tag{26}$$

$$l_1(t) = \frac{(t - t_0)(t - t_2)}{(t_1 - t_0)(t_1 - t_2)} \tag{27}$$

$$l_2(t) = \frac{(t - t_0)(t - t_1)}{(t_2 - t_0)(t_2 - t_1)} \tag{28}$$

t_0 is the oldest iteration.

6. Team Strategy and Individual Behaviour

There are two points of view on the control responsibility – the team strategy decisions and individual actions. Excellent team must find the equilibrium between both approaches. The database of possible strategic formations and movements is used for coordination.
It is a common solution, that the team is maintaining a formation. Each member of the team has assigned a position on the pitch and is responsible for certain area. The goalkeeper's purpose is to guard the goal and passing the ball forward if it is possible. The defender is covering the goal from the angles where it cannot be done by the goalkeeper and is interfering with the opponent's players. The attacker is trying to get the ball under control, avoid the opponent's players and score the goal. Although all agents act individually, the basic pattern is defined centrally.
For the MDM principle, there is no other way than use the combined centralized-autonomous approach where the centralized decision sets the common course of action and every agent act individually, but in correspondence to other agents'. To use some sets of rules to maintain formations (or similar solutions) would make the whole decision-making extremely difficult, complicated and human-control-proofed. Moreover, the analysis and design would be immensely difficult.
Therefore, the principle of coordination is based on the progressive change of behaviour from centralized control to the individual decisions. As the decision-making process is continuing and reasoning is proceeding to lower levels of the strategy module (see Fig. 11),

the agent has gradually greater influence on the form of final realization. Besides, the agent has to adapt universal form of algorithm to actual situation every time.

7. Modular Strategic Control

The actions of the robot players may be decomposed to atomic tasks. The strategic scenarios may also be decomposed to actions is a similar way. In this part, the possibility of change of the set of actions taken into account while making decision is discussed.
The strategic scenario is a coordinated reaction of (one or more) agents to the situation on the pitch. It is useful to have pre-prepared scenarios to most common situations. If the situation on the playground cannot be recognized or if there is a large probability of incorrect reaction, some kind of default behaviour should be used.
There is an advantage hidden in the state-organized structure of behaviour. Once the algorithms are prepared (for either stable or temporal state) the expansion or constriction of the set of states is easily done. When we have the behavioural structure for all states, it may be similar to the one presented at the Fig. 13.

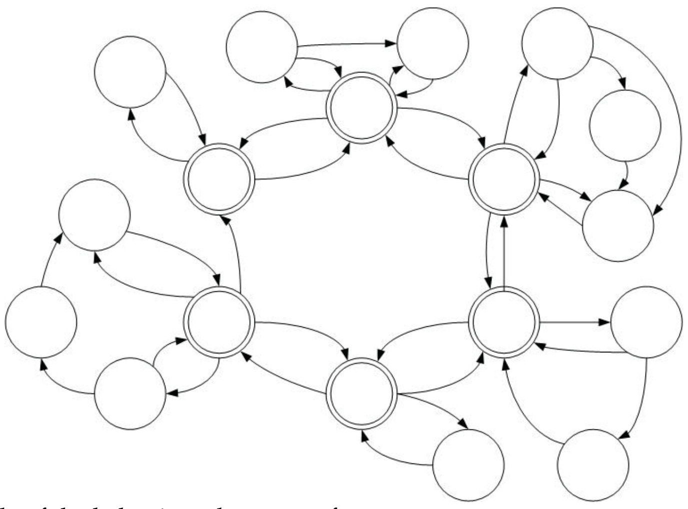


Fig. 13. Example of the behavioural pattern of an agent

To change the behaviour of an agent, some transitions may be suppressed. However, conditions (19), (20) must be fulfilled. The structure presented at the Fig. 13. is called the **behavioural pattern** of an agent and it is defined individually for every type of the agent on the pitch (attacker, goalkeeper, defender). E.g. there would be another behavioural pattern defined for the aggressive strategy and for defensive strategy.
The great advantage is the modularity of the behavioural pattern. New states (algorithms) may be easily incorporated into existing structure and there is no need for changes in the basic structure. Also, this feature is invaluable for the testing and weight function tuning purposes.

7.1 Machine Learning

The most appropriate learning method for the MDM principle is the reinforced learning, according to the MDM principle requirements. Explanation of this method can be found in (Ferber, 1999) and (Kubik, 2004). However, it is important, that it is only a high level of strategic module that is capable of learning. On the medium and low level, the actions must be already debugged and performed precisely; the high level is responsible for the situation recognition and this is a process where the learning may be proven useful.

8. Synchronization of Team Activities

During the robot soccer game, a group of mobile agents must often coordinate their actions to follow the team strategy. Therefore, there is a need to adapt the plan of each participating agent in order to act co-ordinately. Although the basic set of actions, which would be taken, is known from the high level of the strategy module, the final way of realization is depending on the situation on the playground.

As the most appropriate solution was selected the application of hierarchy for the coordination purposes. Generally, it is possible to decide which player has the best position on the pitch. Therefore, such an agent is taking the lead for the execution of co-ordinated action. We have only two types of roles for this purpose. The role of **leader** is assigned to the agent in the best or most important position. The **support** role is used for other agents.

Because we need all team members (except the goalkeeper) to be able to participate in the game properly, the designation of the agent into the position (attacker, defender) is not constant. It may be changed during the game. However, the distribution of power on the pitch must remain the same. Therefore, another agent (usually with the worst position for the actual action) takes its place instead.

It is the leader who chooses the most appropriate solution for his situation. Other players are acting accordingly. E.g. when our team is attacking, it is usually a good idea to have a player near to the opponent's goal. When the ball is near, only a swift move will score a goal. Other actions may be planned in a similar way.

9. Case Study – Goalkeeper's Behaviour Description

For explanation purposes and better understanding, a description of the goalkeeper's behaviour will be provided. The aspects of MDM described in the previous parts of this chapter will be shown on this example.

The goalkeeper's behaviour is not directly dependent on other team member's behaviour. Therefore, the case study is not unnecessarily complicated by the social aspect of the game. Discussion about these features would take a lot of time and for basic understanding it is not necessary to make things difficult.

The goalkeeper's main task it to protect the goal from the opponent's players. To do this, it is important to monitor the position of the ball. Because the ball may not move unpredictably (except the case of collision), but only in the direction of movement of the robot which has it under control, we use prediction described by formulas (24), (25), (26), (27), (28). We have to react to the next position of the ball, not to the actual one. The problem is schematically described on the Fig. 14.

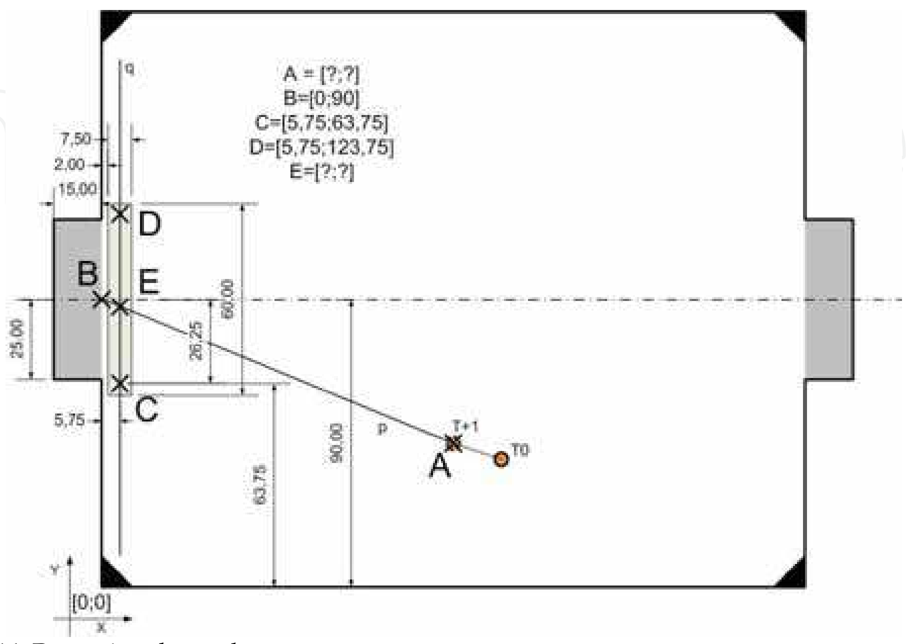


Fig. 14. Protecting the goal

The activity of the goalkeeper is following: (1) remain on the abscissa CD; (2) set the coordinates (point E) to intersection between point B and the predicted position of the ball (point A), (3) the front must remain turned concurrently with the line q (90° or 270°). By following this simple set of rules, the goalkeeper is able to defend the goal. The time difference between the moment T0 and the point A may be adapted as needed (according to the speed of game and capabilities of the opponent's players). However, this simple behaviour is furthermore complicated by handling exceptional situations which may never be entirely avoided in the real world. The most common problem is moving away from the line of movement.

Using the terminology introduced in the beginning of this chapter, the stable state of the goalkeeper agent is defending the goal. Two deviations from the ideal state may occur: the agent may be away from the designated line (CD) or he may be turned in an incorrect angle. If both exceptions occur, it is obvious in what order should be exceptions handled. First will be positioning, second will be turning in the proper angle. Therefore, there are monitored two corresponding attributes – the position (it is compared with the optimal state (see Fig. 14)) and the rotation. The positioning and turning are two temporary states; each of them is a simple algorithm solving the part of the problem. When the exceptional situation occurs, the decision-making procedure is performed and appropriate action selected.

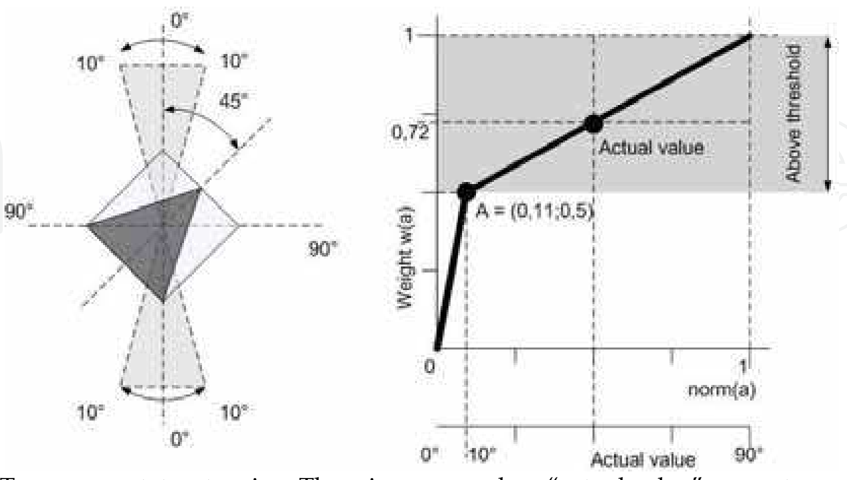


Fig. 15. Temporary state – turning. There is an exemplary "actual value" present

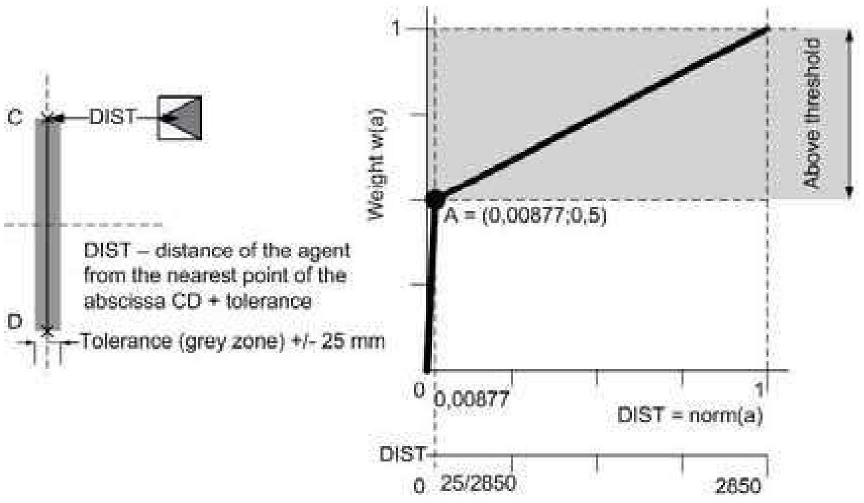


Fig. 16. Temporary state – positioning. The maximum value of the attribute DIST is given by dimensions of the pitch (its diagonal is 2842,53 mm)

It is important that there is a threshold value assigned to the each state. The purpose of the stable state is to prevent treating of every negligible difference from the ideal state. Certain level of imprecision has to be tolerated always. For the stable state, the threshold value is used to recognize when the algorithm should stop. See the Fig. 15 and 16 for details.
The behaviour of the goalkeeper agent is therefore defined by the behavioural pattern shown at the Fig. 17.

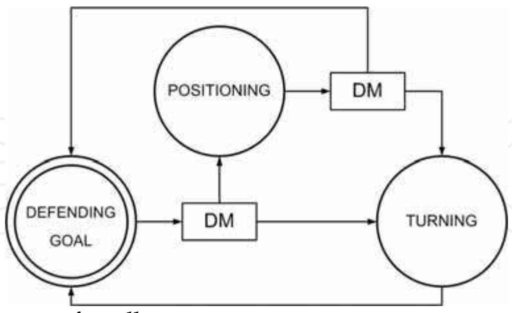


Fig. 17. Behavioural pattern of goalkeeper agent

The "DM" symbols at the Fig. 17 are representing decision-making moments. Until the threshold value is exceeded, the agent remains in the stable state "Defending goal". If the agent is turned in a wrong angle or has a wrong position, the stimulus' power will overcome the threshold and action with highest convenience value is selected and done.

Generally, there are two ways how to continue. The first way requires the agent to make prediction of development more than one step ahead. In this case, when deciding what to do next, the whole sequence of activities is considered (i.e. sequence of actions POSITIONING and TURNING, in our case). In this case, a maximum number of possible solutions will be matching the formula (16). This approach is making the planning and co-ordination more effective.

The other way how to proceed is to select only one next action at the time. This gives the agent an opportunity to quickly react to problems. However, the maximum effectiveness of the agent is obtained when it is staying in the stable state.

The goalkeeper's behaviour may be further complicated by the incorporation of the defender agent into our case. To be effective, the agents need to adapt their behaviour to each other. However, the leading agent is goalkeeper. He continues to perform his task as it was described above, the defender is trying to cover goal where it is impossible for the goalkeeper.

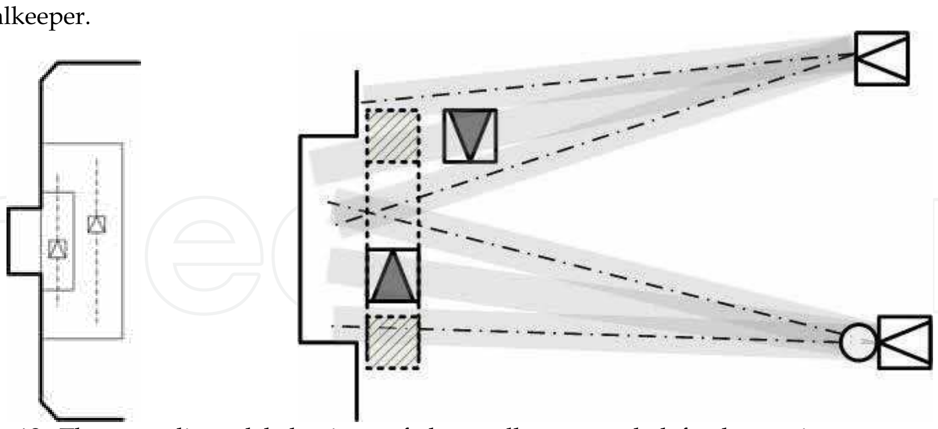


Fig. 18. The co-ordinated behaviour of the goalkeeper and defender against opponent's attack (white agents on the right side)

This description of the goalkeeper's behaviour is not meant to represent a complete behaviour of an agent. Its purpose is to illustrate the application of the MDM principle on the exemplary situation.

10. Conclusion and Future Work

The MDM method shows a great potential in the means of adaptability and speed. From this point of view it is suitable for the purposes of the robot soccer game. It is able to react quickly enough in the strict time-frame limitation of the image processing speed. Its modular framework (implementation of the actions is made separately) allows an easy modification of behavioural control. The behaviour of the individual agent or the team as a whole may be easily changed by the human operator or some form of heuristic control. This allows the team to easily change the tactics and strategy and surprise the opponent.
However, there are some serious setbacks present. The control mechanism development is difficult mainly in the analysis and testing phases. Implementation of isolated tasks is not a problem; this work has to be done no matter what control method is used. What is difficult is the initial design of weight functions and transition-between-states debugging. Both issues require extensive testing and optimization, which takes a lot of time and effort. Our future work will be focused on these problems.

11. Acknowledgements

The work and the contribution were supported by the project from Grant Agency of Czech Academy of Science – Strategic control of the systems with multiagents, No. 1ET101940418 (2004-2008) and by the project IPSV3/07 of University of Hradec Kralove.

12. References

Aguilera, M. K.; Chen, W. ; Toueg S. (1997). Heartbeat : A Timeout-free Failure Detector for Quiescent Reliable Communication. *Proceedings of 11th International Workshop on Distributed Algorithms*, pp. 126-140, ISBN 3-540-63575, Saarbrücken, Germany, September, 1997, Springer-Verlag, Berlin

Byrne, C.; Edwards, P. (1996) Refinement in Agent Groups. In: *Adaptation and Learning in Multi-agent Systems, Lecture Notes in Artificial Intelligence 1042* (Ed. Weiss, G.; Sen, S.), pp. 22-39, ISSN 0302-9743 (Print) 1611-3349 (Online)

Decker, K. S. (1995). Environment Centered Analysis and Design of Coordination Mechanisms, Ph.D. thesis, University of Massachusetts

Fiala, P.; Jablonsky, J. & Manas, M. (1997). *Vicekriterialni rozhodovani,* The University of Economics, Prague, ISBN 8070797487, Prague

Ferber, J. (1999). *Multi-agent Systems: An Introduction to Distributed Artificial Intelligence,* Addison-Wesley, ISBN 0-201-36048-9, New York

Genesereth, M. R., Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publ., ISBN 0-934613-31-1, Los Altos, Cal.

Kelemen, J. (2001). Reaktivni agenti, In: *Umela inteligence 3* (Ed. Marik, V., Stepankova, O., Lazansky, J. et al.), pp. 161-188, Academia, ISBN 80-200-0472-6, Prague

Kim, J.; Kim, D.; Kim, Y. & Seow, K. (2004). Soccer Robotics. *Springer Tracts in Advanced Robotics,* Vol. 11/2004, pp. 205-256, ISSN 1610-7438 (Print) 1610-742X (Online)

Kit, E. (1995). *Software Testing in Real World: Improving the Process*. Addison-Wesley, ISBN 0-201-87756-2

Kubik, A. (2004). *Inteligentni agenty – tvorba aplikacniho software na bazi mutiagentovych systemu,* Computer Press, ISBN 80-251-0323-4, Brno

Menzies, T. (1999). Knowledge Maintenance: The State of the Art. *The Knowledge Engineering Review,* Vol. 14, No. 1, (1999) 1-46, ISSN 0269-8889

Ramik, J. (1999). *Vicekriterialni rozhodovani – Analyticky hierarchicky proces (AHP),* Silesian University, ISBN 80-7248-047-2, Karvina

Tichy, P. (2003). *Social Knowledge in Multi-agent Systems,* Ph.D. thesis, CVUT, Prague

Tucnik, P.; Kozany, J.; Srovnal, V. (2006a). Multicriterial Decision-making in Multi-agent systems, In: *Lecture notes in Computer Science* (Ed. Alexandrov, V. N., Dick van Albada, G., Sloot, P. M. A., Dongarra, J.) pp. 711-718, ISBN 3-540-34383-0, ISSN 0302-9743, Springer, Berlin

Tucnik, P.; Kozany, J.; Srovnal, V., Pokorny, J., Lukas, D. (2006b). Software Structure of Control System for MiroSot Soccer Game, In: *Proceedings of FIRA Roboworld Congress 2006,* (Ed. Weiss, N., Jesse, N., Reusch, B.), pp. 183-188, University of Dortmund, ISBN 3-00-019061-9, Dortmund

Wooldridge, M. (2002). *An Introduction to Multi-agent Systems*, J. Wiley & Sons, ISBN 047149691X, London