AT Humboldt — Development, Practice and Theory

Hans-Dieter Burkhard, Markus Hannebauer, Jan Wendler*

Institute for Computer Science
Humboldt-University of Berlin
10099 Berlin, Germany

Abstract. This article covers three basics of our virtual soccer team AT Humboldt: We describe our development process in the frame of a practical exercise for students. The resulting efficient agent-oriented realization is explained, and we give a theoretical embedding of our planning component based on BDI.

1 Introduction

One of the recent fields of Artificial Intelligence is *Agent-Oriented Programming* (AOP, cf. e.g. [WOOLDRIDGE/JENNINGS, 1994], [SHOHAM, 1993]). AOP is proposed especially for the programming of autonomous components ("agents") in open heterogeneous systems. The agents (players) in Artificial Soccer [KITANO ET AL., 1997] must plan and execute actions individually and efficiently resulting in a successful cooperative team behavior.

This article covers three basics of our virtual soccer team AT Humboldt: We present a description of our development process in the frame of a practical exercise for students at the Humboldt-University of Berlin. We show how this has lead to an efficient agent-oriented realization. We also give a theoretical embedding of our architecture and our planning component.

We hope that our article can be useful for three types of readers: The practician may find some new ideas to enhance his own team, the teacher may get an impression of our work with students, and the theoretician may be interested in an applied *Belief-Desire-Intention* (BDI) architecture.

The article starts with the motivation of our interests in Artificial Soccer. The development process is figured out in Section 3. The main part (Section 4) describes the components and implementation of our agents and Section 5 discusses the theoretical background. Finally, future developments and conclusions are discussed.

* Many ideas and first of all hard implementation work came from Pascal Müller-Gugenberger, Amin Coja-Oghlan, Adrianna Foremniak, Derrick Hepp, Heike Müller and Kay Schröter in a practical exercise. We wish to thank them for their great efforts.

2 Our Interests in Artificial Soccer and First Experiences

The main interests in our group concern Distributed AI, Multi Agent Systems (MAS), Agent-Oriented Techniques (AOT), and Case Based Reasoning (CBR). All of these fields have direct relations to Artificial Soccer and we have fixed the following scientific goals for our participation in RoboCup:

1. *Agent architectures* — We are interested in experiments and evaluations for different approaches like the subsumption architecture ([Brooks, 1990]) and the BDI architecture ([Bratman, 1987], [Rao/Georgeff, 1995]). This concerns the efficient realization of the "belief-to-action"-cycle and the optimal relationship between deliberative and reactive behavior.
2. *Cooperation in MAS* — We are interested in emerging cooperation and improvements by communication, negotiation and (explicit) joint plans. Further interests concern the formation of a team out of different roles and "characters".
3. *Agent learning* — We are interested in the usage of CBR for the training of capabilities and decision procedures (off-line learning) and for the adaption to opponents' behavior (on-line learning), respectively. Especially the treatment of time in these two cases is challenging.
4. *Decision making using vague information* — We have developed a special technique for vague matching which we use in CBR applications ("Case Retrieval Nets", [Lenz/Burkhard, 1996], [Burkhard, 1997]). We want to test it as control structure for the mind of agents (e.g. for the choice and adaptation of precompiled plans using similarity of sensor information/belief conditions).

The implementation of our program until August 1997 has covered mainly point 1 and partially point 2. We have discussed different approaches and finally we have implemented an architecture which is best comparable with the BDI-approach (cf. section 5). The *Advanced Skills* (cf. section 4.4) could also be considered as a layer in a subsumption architecture. Because the higher levels of deliberation have continuous control, it is a more goal oriented architecture in our understanding. Under this view, the *Advanced Skills* are considered as predefined plans.

Emerging cooperation has already given very exciting results: players act according to their expectations concerning the behavior of their team-mates. Different roles result in an efficient usage of the whole playground. But we have also identified several situations where communications would improve the behavior (but this was not implemented up to now).

The realization of Point 3 is supported in the architecture by the history mechanism within the world model (cf. section 4.2), but it was not used for learning up to now. There was some misunderstanding in the announcements: We have declared CBR as our research goal for RoboCup (hoping to get ready with it until the competition in Nagoya). But in fact we did not use it in RoboCup97. We have observed in our studies that learning may lead to suboptimal (or even worse) behavior if it is applied to insufficiently analyzed/developed underlying skills. To give an example: A directed kick needs some preparation (stop the ball, place the ball for kicking, final kick). Hence the players have to learn a related *sequence* of parameterized actions. A careful analysis can specify a skeleton for a successful skill, while parameter settings according to a given situation are due to training (as for humans). Another example concerns the decision component: You can learn proper deliberation only if you can rely on the proper execution of the skills.

The Case Retrieval Nets (Point 4) seem to be suited for the learning of deliberation processes, but we have no experience up to now.

3 The Development

The overall development strategy was to keep in mind all our interests from the very beginning (e.g. histories for learning, desires and intentions for deliberation). That means to develop a structure which is open for further refinements and extensions. We also tried to obtain best results with least effort (e.g. cooperation without communication as far as possible).

We have used protocols (log-files) of internal and external information flow and decisions. The study of those files gave us hints to inexact implementations of our ideas (but it also occurred that "wrong" executions were caused by net overload).

We started with basic work on a class library for UDP/IP, sensors and commands. First ideas were developed by prototyping (extended "simple client"). The development of soccer agents was the topic of a practical exercise for students in connection with a lecture on MAS/AOP/CBR during summer semester (April - July 1997). Besides the discussion of concepts some modules and routines were implemented in C++ and JAVA.

Starting in the second half of July a team of three students finished the work on the implementation of the modules (only in C++ for reasons of computational speed). The modules were combined and tested. The first results showed approximately equivalence to the power of the "Ogalets". After fixing some bugs, tuning and further refinements we could improve the scoring to 29:1 before leaving for Nagoya.

Some further changes during the breaks of the competition in Nagoya concerned especially positioning (fuzzy positioning, positions for corner kick and goal kick, changes in the home positions).

4 The Ideas of the Realization

This section gives an overview and some details of the different components of our soccer playing agent team. Additionally it may help the reader to get through our published code[2]: We will mention the particular files which implement the

[2] (see `http://www.ki.informatik.hu-berlin.de/RoboCup97/index_e.html`)

described parts. The whole project was implemented in C++ under Solaris and Linux, respectively.

4.1 Clear Architecture

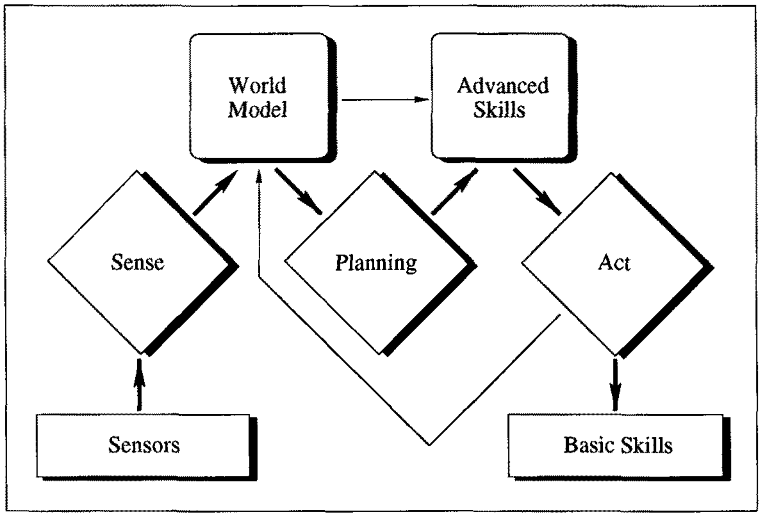

Fig. 1. General Architecture of a Soccer Agent.

The general architecture is identical for all agents, the player identifies its role (goal-keeper, defender, attacker ...) by the player-number given from the server. Special behaviors according to the roles are due to different parameter values. Figure 1 shows the agent's abstract architecture. Thick arrows indicate the main information flow. Rhombs symbolize concurrently running components. An agent (`agent.*`) consists of all these components which can be seen as "organs".

The component *Sensors* (`sensors.*`) parses and transforms the string coded sensor information which the player receives from the SoccerServer. The component *Basic Skills* (`baseskills.*`) provides methods for sending basic commands to the server.

4.2 Stable World Modelling

In a dynamic and uncertain domain like Artificial Soccer a consistent modelling of the environment is necessary. Short-term false information has to be corrected, un-precise information must be evaluated and inferences are necessary for missing information. This leads to a certain stability of the agent's belief. It is realized by the complex component *World Model* (`weltmod.*`, `fussball.*`, `positions.*`, `sensori.*`, `spielfe.*`), which provides e.g. basic classes for linear algebra. Every object on the field is described by a special class. Inheritance is strongly used, and additional features like timed objects and encapsulated environments make synchronization easier.

The agent's absolute position on the field is calculated using the visual information concerning lines, flags and goals. The triangulation considers all possible cases (actually several hundred). The agent's velocity is estimated from the commands sent to the server. Additionally, the player records its stamina. Absolute positions of all other seen objects can be computed because the own absolute position is known. A special algorithm matches new information on unnamed objects (e.g. a player with a missing number) to known objects. The world model can also close the information gaps for unobservable objects by simulation.

Simulation is also used to predict future situations by using the knowledge about positions and velocities. This ability is extensively used by *Planning* and *Advanced Skills* to estimate consequences of possible commands. For example, *Advanced Skills* can instantiate a new ball object, simulate it for some time steps and look at the position and speed. Additional features like wind could be easily taken into account.

World Model logs an adjustable number of environments to keep track of the player's history. This ability was implemented to support on-line learning, but it was not exploited in RoboCup97.

4.3 Bounded Rationality

The component *Planning* (`entscheidung*.*`) embeds the planning and reasoning process which leads successively from coarse plans to concrete command sequences. The theoretical background of this procedure is described in section 5. Here we show how the internal process works.

A new planning process is initiated each time a new sensor information has arrived . The situation is classified: If the ball is under control, the agent is able to pass the ball or to dribble. If the player has no control of the ball, it can decide whether to intercept it, to watch the surrounding or to run to a certain position. This goal (target) finding is done by a usual decision tree which selects a goal out of a fix goal library. Some of the decisions are trivial ("Is the ball in the kick range or not?") but some are really tricky ("Shall I run to the ball or shall my team-mate do it?"). The latter decision is done by a reachability simulation: If the agent supposes to be the first of its team to reach the ball, it will run. If not, it relies on its team-mates and runs back to its home position. Because we have not used communication yet, the stamina of team-mates has not been regarded and may have caused wrong expectations.

After the goal selection the player has to find the best way to achieve its goal. This phase of the planning process produces a coarse long-term plan with some parameters (*partial plan*). The plans must be based on the skills of the agents. This means that plans are in a close correspondence to the *advanced skills*, which are themselves building blocks for the execution of the long-term plans. There are two major cases to cope with: The agent is out of the kick range or it can control the ball (i. e. has ball possession).

In the first case the player calculates an optimal interception position if it has decided to get the ball. For several reasons (e.g. to regard the wind if necessary) we decided to utilize the simulation capability of the world model. The agent tries mentally to reach the ball in one step, in two steps and so on until it finds a certain number of steps in which it can reach the ball. This procedure also provides a distance measure because it can be applied to every player and ball instance. Figure 2 shows this process graphically and gives a commented example.

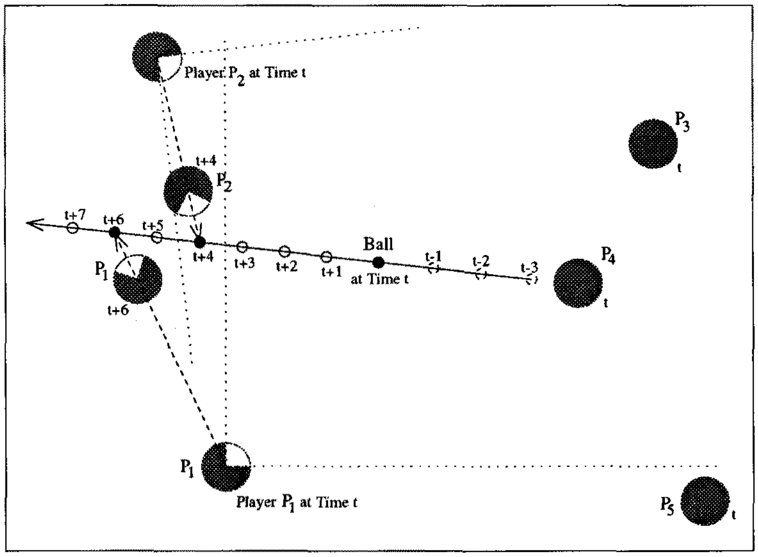

Fig. 2. Decision Finding for Ball Interception.

The solid line shows the ball movement with the ball positions at the time steps $t \pm i$. The dotted lines represent the view sectors of the different players P_1 and P_2 at time t. The dashed lines show the movements of the players.
P_1 does not see P_2 and P_5. It calculates distance 26 for P_4, 29 for P_3 and 6 for itself. The agent decides to intercept the ball. P_2 whose view sector includes P_1 calculates 6 for P_1, 30 for P_3, 26 for P_4, 39 for P_5 and 4 for itself. It also decides to go for the ball. Hence both players of the same team go for the ball.
P_1 calculates interception position at ball position $t+6$. Likewise does P_2 for ball position $t+4$. At time $t+3$ a new sensor information comes in. P_1 cannot see the ball. It will keep its plan according to the implementation of our planning component for such situations. P_2 sees the ball and also continues intercepting the ball.

If the player has decided not to intercept the ball, it returns to its home position, or (if it is already there) collects information by turning and waiting. Further development will lead to more sophisticated behavior (e.g. double passes and explicit team strategies). The architecture is prepared for such extensions (we have implemented special behavior for corner kicks or goal kicks just in the breaks of the competition in Nagoya).

If the player controls the ball, it has to decide whether to pass the ball or to dribble. Furthermore it has to decide in which direction to kick or to dribble, respectively. It should prefer a direction with best chances to score or to pass the ball to a team-mate. At the same time it should prefer directions promoting an offensive play style. A fixed direction d is evaluated by the following formula:

$$\Delta(d) = \omega_b(\beta_t(d) - \beta_o(d)) + \omega_m(\mu_t(d) - \mu_o(d)) - \gamma_o(d) + \gamma_t(d)$$

The indexes t are used for terms indicating values of the own team, indexes o for the values of the opponents, respectively. ω_b and ω_m are role dependent weight factors with $\omega_b + \omega_m = 1$.

β is the minimum of the distances δ to the ball for all players of a team (indexes t or o). The distances δ are calculated by simulation as described before, if the ball is kicked in the given direction d with a certain velocity. It is computed for the own team by:

$$\beta_t(d) = \min\left\{ \sqrt{\delta(d,i) \cdot \lambda(d,i)} \;\middle|\; i \in S_t \right\}$$

S is the set of seen players. The additional factor λ is the length of the perpendicular from player i to the ball line (this value provides an influence of stamina loss).

μ provides the mean distance of a team to the ball line. The unseen players are approximated using a worst distance δ_w (P is the set of all team players):

$$\mu_t(d) = \frac{\sum_{i \in S_t} \left(\sqrt{\delta(d,i) \cdot \lambda(d,i)} \right) + |P_t \setminus S_t| \cdot \delta_w}{|P_t|}$$

γ is a goal hitting bonus, where δ_k is the "typical" kick distance and δ_g is the distance to the goal, respectively:

$$\gamma_t(d) = \begin{cases} \max\left(0, \delta_w - \frac{\delta_w \cdot \delta_g}{\delta_k}\right), & d \text{ hits our goal} \\ 0 & , \text{ otherwise} \end{cases}$$

The values for ω_b, ω_m, δ_w and δ_k have to be tuned (it is intended to use learning). β_o, μ_o and γ_o are defined analogously.

Having the formula to evaluate directions d by their values $\Delta(d)$ we can look for an optimal direction. Due to the definition (convex combination by weight factors, symmetrical values for opponents' players and team-mates) the evaluation function is normalized to zero. This means that negative values indicate a "good" direction and positive values indicate a "bad" direction. In our recent implementation several discrete directions are evaluated and then the best direction is taken. Figure 3 exemplifies such an evaluation process.

Values around zero are neutral. Especially these values around zero are difficult to judge. For this purpose a randomized function was implemented which decides afterwards whether to kick or to dribble.

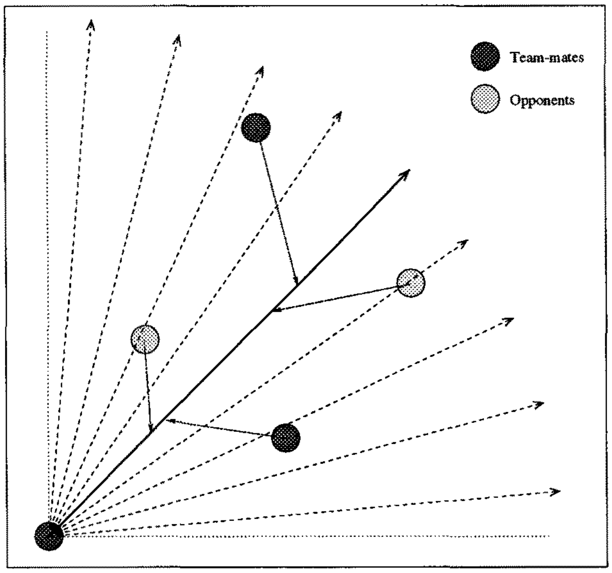

Fig. 3. Evaluation of Discrete Kick Directions.

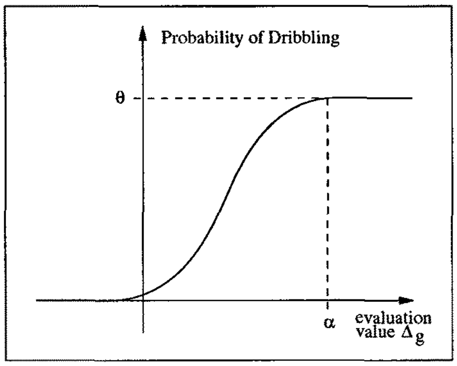

Fig. 4. Probability function for Dribbling.

The function is shown in figure 4. It is on the left half a Gaussian function with the following definition:

$$P(\text{Dribble}) = \begin{cases} \theta \cdot \mathrm{e}^{-\frac{(\Delta_g - \alpha)^2}{3.38}} & , \quad \Delta_g \leq \alpha \\ \theta & , \quad \Delta_g > \alpha \end{cases}$$

θ is a role dependent dribble factor (e.g. the goal-keeper has $\theta = 0.1$, the attacker $\theta = 0.5$). Δ_g is the minimal distance over all evaluated directions. α is an acceptance constant for kicking which is also role dependent.

The main problem of the evaluation strategy described above is that players move in unpredictable ways. Therefore the positions of unseen players are uncertain. This is why we decided to evaluate only directions in the view area (we used a view angle of 90 degrees and high quality). To prevent too many backward shots, every area on the field has a defined preferred direction. Our first idea was the following procedure: Turn to preferred direction and evaluate the kick directions. If there is no "good" kick direction, determine a turn direction and turn, evaluate this sector and so on. This was a safe way to find the global best direction, but was actually to slow because the agent had to wait 300 ms to get the next information. So we implemented the following behavior: If the agent looks approximately into the preferred direction (the definition of "approximately" is role dependent), it will do the evaluation process. If not, it will do an "emergency kick" directed to the opponents goal. With additional information via communication this simple behavior could be omitted in favor of full 360 degree evaluation in the future.

Cooperation between team partners emerges by relying on the behavior of team-mates, that means team-mate modelling. The player relies on the fact that the team-mate with the shortest distance to the ball will try to intercept it when it is passed in its direction.

As soon as the agent has decided for a partial plan, it is given to the component *Advanced Skills* for execution.

4.4 Efficient Execution

The component *Advanced Skills* (`advancedskills.*`) is a library of skills which facilitate efficient ball handling and optimal movement. The technical task of this component is to split the long-term partial plans into short-term plans, which means concrete command sequences with full parameters. These short-term plans are not longer than the interval between consecutive sensor information (with our preferences these are actually three basic commands). This way the long-term plans are executed by iterated calls of advanced skills after each sensor information.

It was one of the major decisions during development to use this strategy of plan execution. The more common strategy is to fix a long term plan which may be adapted during execution if necessary. Such a long term plan can start with some initial actions to achieve a well defined situation (e.g. a suitable ball position for dribbling). Afterwards the actions are performed in the fixed sequence according to the plan, such that each action relies on the successful execution of its predecessors.

In our strategy, a strictly new deliberation process can start for each new sensor information (actually each 300 msec), and in this case we have a new long term plan started just at this time point. If we would need always certain (new) initial actions for preparation, then we might never come to the continuation of a plan (cf. Section 5 for further discussions). To overcome this problem, the advanced skills are designed to deal immediately with any situation which might appear at the beginning or during the execution of a long term plan (e.g. to continue dribbling in any situation). As a side effect, the advanced skills are able to realize the fastest way for goal achievement in a very flexible way from arbitrary start situations.

Advanced Skills uses *Basic Skills* to send basic commands to the server. We modified the basic *Turn* command because the original command which is directly supported by the server is influenced by the player's speed. We enhanced the *Turn* command to compensate this influence by increasing the turn angle or — in worst case — stopping first and then turning. The other basic commands are sent without modification to the server.

The main advanced skills of the player agent are: *Directed Kick*, *Go to Position* and *Dribble*. *Go to Position* enables the agent to reach every absolute position on the field. It produces one *Turn* (if needed) and/or up to two/three *Dashes*. If demanded, this procedure avoids obstacles like other players. *Dribble* moves the ball into a certain direction without loosing contact to it. This includes the production of several *Kick*, *Turn* and *Dash* combinations.

The *Directed Kick* skill was one of our competitive advantages and therefore it will be described in detail in the following. This capability allows the players to kick the ball into any direction with a demanded power (as far as possible). It handles difficult situations like high velocities and situations where the player itself is an obstacle for the desired direction. If the desired direction with the desired speed cannot be achieved, the skill tries to meet the demands as good as possible.

First of all the skill tries to determine the kick angle and power which is necessary to transform the current movement vector into the demanded movement vector (Part A of figure 5). If the length of the necessary kick vector (the power) is physically impossible, the skill tries to keep at least the right direction.

A complication occurs for kick vectors which are possible but hit the player itself. In this case an intermediate target is calculated which is at the side of the player (Part B). The first kick leads to this point and further kicks are calculated from there (Part C). In some cases the ball can be kicked once more (Part D).

This leads to the efficient kicks which were observable in the matches of our team. All this can be done with three basic kicks: The implementation of the SoccerServer for Nagoya permits addition of velocities by consecutive kicks following the laws of vector algebra. According to the Nagoya settings, the players can start with approximately 10 m/sec and the ball with 13 m/sec. By additional dashes, the players can get a speed of approx. 16 m/sec. Two consecutive kicks in the same direction give the ball a speed of approximately 25 m/sec. which is less than two times faster than the players' speed. It is reasonable that it takes two actions (i.e. more time and more precise action settings) to make a stronger kick: The two kicks can be considered as a compound action (just like turn + dash). It can even be considered as a triplet together with the need to place the ball at a certain position before a strong kick. This setting leads to an interesting challenge for the learning of compound actions.

Problems arise for the interception of a fast running ball: Especially increasing velocity for fast moving balls by likewise small additional kicks of an intercepting player looks problematic. Furthermore, we found it difficult (but not

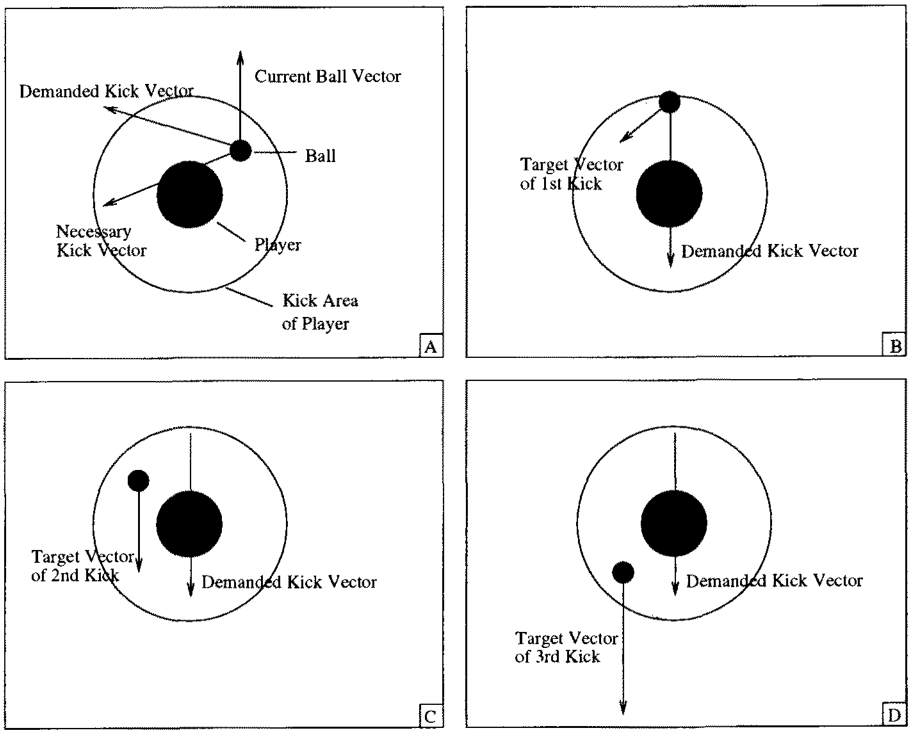

Fig. 5. Several Steps of the *Directed Kick* Skill.

impossible) to stop a ball by defending players.

Alternatively the SoccerServer could simulate kicks as new settings of the ball velocity instead of additions. In this case, the following problem arises: Setting of a fast arriving ball with a moderate single kick to a new slow velocity (defense could be too simple with such settings)[3].

4.5 Precise Synchronization (system programming)

In the real time environment of Artificial Soccer it is essential to keep synchronized with the server. For this purpose some components have to run in parallel. In our implementation the components *Sense* (`sensorik.*`) and *Act* (`sender.*`) run concurrently with the main routine to allow an undisturbed reasoning-process and to keep track of the server.

There are different ways to provide pseudo concurrency in a UNIX environment. Processes raise difficulties with communication and storage organization. Threads are not widely supported (Solaris has native support, Linux and others have only library support like pThreads) and have a lot of administration overhead. In addition they are difficult to time. On the other hand UNIX system

[3] This point is still under discussion while writing this article.

signals are efficient, well timed and easy to use. In fact one signal handler is enough to provide all the concurrency the agent needs.

The signal is adjusted to 50 ms. The signal handler looks at top of the incoming message system queue and parses new information if necessary. Accordingly information flags for the planning process are set. In addition the handler peeks every 100 ms on top of the outgoing command queue, sends one command if necessary and initiates a simulation cycle of *World Model*. If the time information of an incoming sensor information differs from the agent's internal time, additional simulation cycles are done until synchronization is achieved. This is essential because the estimation of the agent's own speed relies only on sent commands and passed time. If the calculation of the agent's own speed is wrong, then all other speed calculations will be wrong because they are known only relatively to the values for the agent.

5 The Theory

The consideration of programs as agents focuses at first on the aspect of autonomy: Programs have to act in an appropriate way to changes in the environment. Therefore they need some input or sensor facilities and some output or actoric components. The mapping from input to output can be done in very simple ways (e.g. strictly reactive) or in more sophisticated ways up to models which are inspired by human decision processes. We found that mental notions like capabilities/skills, belief, goals/desires and intentions/plans are very useful pictures to make agent programming transparent. The aspect of rationality forces agents to deal efficiently with their resources, especially with time.

The so-called BDI model fits best to our concept of soccer agents. BDI stands for belief – desire – intention, and the approach is based on the philosophical work of BRATMAN [BRATMAN, 1987], and the theoretical and practical work by RAO/GEORGEFF [RAO/GEORGEFF, 1995] and others (cf. e.g. [WOOLDRIDGE/JENNINGS, 1994], [BURKHARD, 1996]). Agent-Oriented Programming is a fast developing field of research, and Artificial Soccer is a very suitable field for experiments. Rao and Georgeff write about typical characteristics of problem domains which can be successfully solved by BDI. These characteristics fit well to the soccer domain:

- The SoccerServer and the opponents create a non-deterministic environment.
- The agent itself reacts non-deterministically because parts of the planning process are randomized.
- The player can have different goals at the same time, e.g. to reach the ball while covering an opponent.
- The success of the player's own commands depends strongly on the simulated environment and the opponents.
- The whole information is local and different for every player.
- The environment pushes *bounded rationality* because too deep reasoning is without pay-off in a dynamic surrounding.

In the BDI-approach, agents maintain a model of their world which is called *belief* (because it might not be true knowledge). The way from belief to actions is guided by the *desires* of the agent. BRATMAN has argued that *intentions* are neither desires nor beliefs, but an additional independent mental category. Intentions are considered as (partial) plans for the achievement of goals by appropriate actions. Commitment to intentions is needed which has impact on the rational usage of resources: The (relative) *stability of committed intentions* prevents overload in deliberation and useless plan changes, and it serves trustworthiness in cooperation.

All components of a BDI-architecture can be identified in our planning process. *Belief* equals the component *World Model* (cf. section 4.2 in our realization. The strict relativity of the server statements leads to an individual image of the world in every agent. Additionally the agent cannot rely on the accuracy of the received and interpolated data, therefore it is belief not knowledge. The update routines (including the simulation for unseen objects), the simulation of expected future situations and the history of situations are also considered to be parts of the belief.

In our implementation *Desires* are goals which are selected out of a fixed goal library. The list of possible goals is still small, but the set will be extended e.g. to allow joint goals (like double passes) in the future. In the present realization different (even opposite) goals may be achievable but the agent selects only one of them[4]. To do this the environment is classified by a decision tree. The selection function and the execution of the chosen goal must be fast because it must not be interrupted by new information which could be decision relevant. In spite of that, reasoning must be accurate enough to fulfill its task optimally. This is the same trade-off as described by Rao and Georgeff. The risk of full execution of long-term plans is that the agent cannot adapt correctly to unforeseen events. On the other hand permanent evaluation and control of every planning step is too expensive in the sense of computing resources.

This implies *Intentions* which are divided into two stages of planning in our system. At first the best possibility to reach the chosen goal is computed and fixed as an intention (cf. section 4.3). This corresponds to the long-term plan with some parameters which can be also seen as a partial plan. Its calculated end is the fulfillment of the selected goal. The execution of the intention is split into smaller pieces which are implemented as short-term plans in the component *Advanced Skills* (Section 4.4). As mentioned above *Advanced Skills* provides precompiled plan skeletons of a size that fits between two time points of sensor information. They have their own calculation capability which is used to compute the short time optimal command sequence. Looking at the mentioned trade-off these short-term plans are atomic and cannot be interfered by sensor information. But in composition they build a long-term plan that is complex enough to fulfill higher

[4] In the future we may deal with the commitment to concurrent goals. In such a case we will have to regard the "scope of admissibility" (BRATMAN) set by previous intentions. For example, an existing intention to preserve the off-side position of an opponent may restrict the commitment to later goals for reaching special positions.

goals. There is a certain overlapping with the decision procedures for desires: This is necessary, since the decision process has to look for achievable desires. The realization of the intention relies on the capabilities of the agent, which are implemented by the advanced skills.

The consideration of our agents as BDI-constructs is appropriate since we have for each new sensor information a complete deliberation process with update of belief, choice of a desire, commitment to an intention and execution of a plan part.

A problem arises from the fact that commitment of intentions is mostly performed independently from the previous intentions: This might contradict the mentioned principle of stability of committed intentions, which is a central point in BRATMAN's theory. The "canonical" deliberation process has to maintain an old intention as long as there are no serious counter indications.

Because a new deliberation process might be initiated every time a new sensor information comes in and then new plans are created, the planning strategy has to ensure stability of the long-term plans to avoid constantly changing goals or intentions. The stability of our goal tracking relies on the fact that even in the case of a new initialization of the whole planning process very often the same steps are chosen because the situation has not radically changed. That means that the same goal and intention are created, too. The player must prevent that missing knowledge or only slightly better other intentions destroy the former behavior. Indeed, a simple implementation of our strategy would have serious drawbacks. As already mentioned in Section 4.4, simple plans could result in only repeated initial actions. To avoid this the agent for example lets out minor turns on the way to a certain position which would cost too much time. This could be called implicit persistence.

The explicit persistence of goals and intentions in our implementation can be exemplified by the realization of the goal "Go to home position". To decide whether to intercept the ball or to go home the agent has calculated the *minimal* distance of all team-mates and opponents to the ball (cf. section 4.3) and has stored these values. If the decision is to go home, the agent will use these values to determine a time interval in which it must not care for the ball because no other player will be able to change the ball's known movement. The decision tree usually strongly relies on sight of the ball but in this case the agent won't turn for the ball in the calculated "don't care"-interval on its way to its assigned position. This results in a straight run to the designated position until the "don't care"-time will be over. In general that means that the old goal and intention is kept as long as the calculated interval lasts.

We found this to be a very interesting implementation of the stability principle for committed intentions without explicitly using the old intention. Our agents are able to adapt a plan to new situations if necessary (e.g. a turn-command with a greater change would not be dropped). It might be the case that future implementations would need an explicit treatment of previous intentions (e.g. if there was a commitment given to team-mates in explicit cooperation).

6 Future Work

The next steps to improve our players concern various forms of training and learning as well as other methods of cooperative play:

- Some decisions and skills use individual parameters which values were found by testing evaluation. These parameters shall be tuned by the agent itself.
- Methods from Case Based Reasoning can help to reuse "good" decisions in equal situations.
- Learning means training of behavior with off-line learning as well as adaption to the opponents' behavior in the match (on-line learning).
- The team play shall be improved by a special behavior in well-known situations, explicit cooperative skills and use of the provided communication on the field.

7 Conclusion

The following lessons have been learned during our discussions and implementations:

- An architecture which makes the agent processing transparent is important for development of concepts, implementation and fast changes.
- An efficient implementation needs the integration of methods from different fields, especially Mathematics, Software Engineering and Artificial Intelligence.
- Methods from Artificial Intelligence are efficient if applied to well performing basic behavior.

The last experience was very important to us. It showed us that "pure" Artificial Intelligence may be insufficient. In our development phase sometimes a better solution from AI view has led to less performance. The reason were basic mistakes which were increased by the sensitive AI techniques. As a result we think that learning can only be based on a strong and correct conventional foundation.

We have published our code to make transparent our ideas (we have to apologize for some "unorthodox" program parts, we hope to present a better version in the future).

We want to express our thanks to all people which are engaged in RoboCup for giving us so much fun. Special thanks are due the Japanese colleagues for organizing RoboCup 97!

References

[BRATMAN, 1987] M. E. Bratman: Intentions, Plans and Practical Reason. Harvard University Press, 1987.

[BROOKS, 1990] R. A. Brooks: Elephants don't play chess. In P. Maes (ed.): Designing Autonomous Agents. MIT press, 1990.

[BURKHARD, 1996] H. D. Burkhard: Abstract goals in multi-agent systems. In W. Wahlster (ed.): 12th European Conf. on Artificial Intelligence (ECAI96). 524 – 528. John Wiley & Sons, 1996.

[BURKHARD, 1997] H. D. Burkhard. Cases, Information, and Agents. In P. Kandzia, M. Klusch (eds.): Cooperative Information Agents. Proc. First Int. Workshop CIA'97. 64–79. LNAI 1202, Springer, 1997.

[KITANO ET AL., 1997] Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, Minoru Asada. The RoboCup Synthetic Agent Challenge 97. In M. E. Pollack (ed.): Proc. IJCAI-97. 24–29. Morgan Kaufmann, 1997.

[LENZ/BURKHARD, 1996] M. Lenz and H. D. Burkhard. Lazy propagation in case retrieval nets. In W. Wahlster (ed.): 12th European Conf. on Artificial Intelligence (ECAI96). 127–131. John Wiley & Sons, 1996.

[RAO/GEORGEFF, 1995] A. S. Rao and M. P. Georgeff: BDI agents: From theory to practice. In V. Lesser (ed.): Proc. of the First Int. Conf. on Multi-Agent Systems (ICMAS-95). 312–319. MIT Press, 1995.

[SHOHAM, 1993] Y. Shoham: Agent oriented programming. 60:51–92. Artificial Intelligence, 1993.

[WOOLDRIDGE/JENNINGS, 1994] N. R. Jennings and M. Wooldridge: Proceedings of the ECAI-94-Workshop on Agent Theories, Architectures, and Languages. LNAI 890, Springer, 1994.