Master Thesis
Software Engineering
Thesis no: MSE-2003:20
June 2003

Situation assessment and role selection in the simulated RoboCup domain

Mattias Sjöberg
`pt99msj@student.bth.se`

Johan Sturesson
`pt99jst@student.bth.se`

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE - 372 25 Ronneby
Sweden

This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to twenty weeks of full time studies.

Contact Information:
Author: Mattias Sjöberg
Address: Ågatan 12, SE - 295 38 Bromölla
E-mail: pt99msj@student.bth.se

Author: Johan Sturesson
Address: Östrastorgatan 17A, SE - 294 31 Sölvesborg
E-mail: pt99jst@student.bth.se

University advisor:
Stefan Johansson
Department of Software Engineering and Computer Science

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE - 372 25 Ronneby
Sweden

Internet : www.bth.se/ipd
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

Abstract

In the recent world championships of the simulated RoboCup league the winning teams possessed low level behaviours, such as kick and pass, that were close to perfection. In order to improve a team's performance you will need, beside perfect low level behaviours, a good management of the team.

We present a model for managing a team in the simulated RoboCup league. The model is based on techniques used by the recent winners in the league and allows you to get a well coordinated team of agents striving for a common goal.

The model supports different formations in different situations, which contributes to a dynamic team play, where the players can adjust to their opponents and other factors like time left and goal difference. For example if the game is near the end and the team is loosing a more risky and aggressive tactic is chosen.

Keywords: Situation, Role, RoboCup, Multi-Agent system

Contents

Chapter 1

Introduction

The goal of RoboCup is defined as:

> "By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team."
>
> -RoboCup Federation [4]

The goal and timetable are aimed to advance the overall level of technology in society. There will be several technological achievements even if the goal is not completely fulfilled, ranging from improved sensor technology to advanced multi-agent coordination policies.

1.1 RoboCup

Mackworth introduced the idea of using soccer-playing robots in research [3]. Unfortunately, the idea did not get the proper response until the idea was further developed and adapted by Kitano, Asada, and Kuniyoshi, when proposing a Japanese research program, called Robot J-League, a professional soccer league in Japan [3].

During the autumn of 1993, several American researchers took interest in the Robot J-League, and it thereafter changed name to the Robot World Cup Initiative or RoboCup for short. RoboCup is sometimes referred to as the RoboCup challenge or the RoboCup domain. In 1995, Kitano et al. proposed the first Robot World Cup Soccer Games and Conferences to take place in 1997 [3].

The aim of RoboCup was to present a new standard problem for Artificial Intelligence and robotics, somewhat jokingly described as the life of AI after Deep Blue [3]. RoboCup differs from previous research in AI by focusing on a distributed solution instead of a centralized solution, and by challenging researchers from not only traditionally AI-related fields, but also researchers in the areas of robotics, sociology, real-time mission-critical systems, etc.

To co-ordinate the efforts of all researchers, the RoboCup Federation was formed. The goal of RoboCup Federation is to promote RoboCup, for example by annually arranging the world cup tournament. Members of the RoboCup Federation are all active researchers in the field, and represent a number of universities and major companies. As the body of researchers is quite large and widespread, local committees are formed to promote RoboCup-related events in their geographical area.

In order for a robot team to actually perform a soccer game, various technologies must be incorporated including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup is a task for a team of multiple fast-moving robots in a dynamic environment.

1.1.1 Simulated league

The RoboCup simulator league is based on the RoboCup simulator, the soccer server [3]. The soccer server supports competition among multiple virtual soccer players in an uncertain multi-agent environment, with real-time demands as well as semi-structured conditions.

One of the advantages of the soccer server is the abstraction made, which relieves the researchers from having to handle robot problems such as object recognition, communications, and hardware issues, e.g. how to make a robot move. The abstraction enables researchers to focus on higher level concepts such as co-operation and learning. Since the soccer server provides a challenging environment, i.e. the intentions of the players cannot be mechanically deduced, there is a need for a referee when playing a match. The included artificial referee is only partially implemented and can detect trivial situations, e.g. when a team scores. However, there are several situations that are hard to detect in the soccer server, e.g. deadlocks, which bring the need for a human referee.

As to now, there have been five world cups and one pre-world cup event [3].

1.1.2 Environmental issues

The type of environment in RoboCup is one of the hardest to deal with according to the definition of environments in "Artificial Intelligence, A modern approach" [9].

- It is inaccessible since the field of view is limited to the view angle and it is only possible to see a part of the soccer field, i.e. the agent has to maintain an internal state of the soccer field.
- It is nondeterministic from the agents point of view[1] because the next state of the environment cannot be determined by its current state and the actions selected by the agent. For instance, there are 21 other players the agent can only guess what actions they will select, and it is not possible to calculate the exact trajectory of the ball.

[1]It is deterministic from the viewpoint of the system, since even the noise can be predicted given that the randomization key and the movement model is known.

- It is nonepisodic because the agent has to plan its actions several cycles ahead, and every action the agent does has impact on the subsequent cycle.
- It is semi-dynamic because the environment will not change while the agent is deliberating if the agent comes to a decision and acts within passage of the cycle time (currently 100ms). If the agent does not send a command to the server before the ending of the cycle, the server calculates the next state without any action from the agent.
- It is discrete in the sense that the agent knows what flags and number of players he can see, and what actions the he can do. But it is continuous in the sense that the possible perceptions of the flags and the players are neither limited nor clearly defined.

1.1.3 CRaPI, a RoboCup API

CRaPI - Correct Robust and Perfect API for simulated RoboCup, is our attempt to provide an API that is well documented and simple to use.

The responsibilities of CRaPI is to provide an up-to-date model of the current state of the environment, a means of communication with the soccer server and a set of utility functions to relieve the user of tedious calculations not related to agent design. In order keep CRaPI independent of the agent architecture chosen by the user, it is driven by events. The current size of CRaPI is 7228 LOC. See the CRaPI manual concerning the details of how to set up a team using CRaPI [12].

An architectural overview of CRaPI is shown in Figure 1.1. The main parts are;

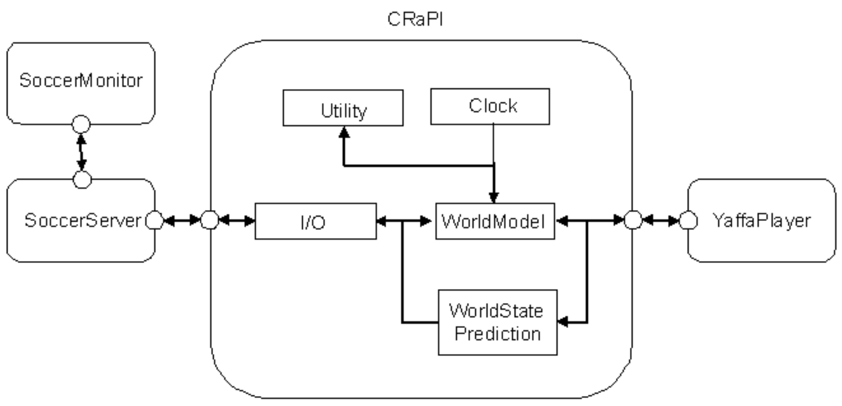


Figure 1.1: Architectural overview of CRaPI

WorldModel The WorldModel is the agent's internal representation of the current state of the environment. The responsibilities of the WorldModel is to calculate the positions, velocities and directions of all objects in the environment, based on perceptual inputs.

WorldState Prediction Since perceptual inputs are not synchronized with the server cycles, a prediction of the world state is needed to maintain high coherence with the actual state of the world.

Clock The internal clock is needed to keep track of when a cycle is about to end, and hence when a command has to be sent to be executed by the server in the current cycle.

Utility The utility module provides a set of useful tools, mostly concerned with geometrical calculations.

I/O The I/O module is responsible for maintaining the communication with the soccer server.

1.1.4 Yaffa, a RoboCup player

Yaffa - Yet Another Force Field Approch, is our RoboCup player. The development of Yaffa has been the single largest task of the project. Yaffa is built on top of CRaPI and uses the electric field model for action selection. The current size of Yaffa is 8630 LOC.

1.1.5 EFA Electric Field Approach

In autonomous robotics, artificial potential fields are often used to plan and control the motion of physical robots [7]. The *Electric Field Approach* [5] is proposed as a generalization of traditional potential field approaches, which allows for both motion control and object manipulation. The main concepts of the electric field approach are artificial charges and probes, where strategically important positions of the probe are positively charged and “dangerous” positions negatively charged. The electric field is then used as a heuristic for action selection by simulating the potential at the probed position for each action. The action resulting in the highest potential in the probed positions is regarded as the most suited for execution in the current situation.

1.2 Problem description

Can a model, representing a team configuration in the simulated RoboCup domain, be built with consideration on maintainability?

1.3 Delimitations

The scope of this thesis is to construct a model for the configuration of a team of agents in the simulated RoboCup domain. The model will link players to specific roles in different situations, and will not focus on the specific low level behaviours of each agent.

1.4 Method

Two research methods were utilized in the project; a literature study and an evaluation through experimentation.

1.4.1 Literature survey

The first task was to conduct a literature survey of architectures for RoboCup multi-agent systems, searching for different role and situation assessment techniques. Knowledge about what others already had achieved would be a guideline to us.

1.4.2 Series of tests

The second task was to construct a model, design and implement a team using it, and by performing series of tests, insure that it is accurate.

1.5 Outline of the thesis

This thesis begins in chapter 1 with a description of the simulated RoboCup league and the pre-developed modules needed to build our application. Chapter 2 begins with a description of some of the different situation assessment techniques developed by the recent winners in the world championships of the league, followed by the model we constructed. In chapter 3 we present our results, we discuss them in chapter 4 and conclude the thesis in chapter 5 with a suggestion on further work in the area.

Chapter 2

Modelling in RoboCup

2.1 Contemporary research

We started with a literature study, looking at the most recent winners in the RoboCup simulated league. There is an obvious pattern, the winner always wins big overall, and hardly ever lets the opponents score. In table 2.1 we present a collection of the statistics about the simulated league that we collected from RoboCup's official homepage[1].

Winning team	Year	Place	Goals	Games played	No. teams
CMUnited98	1998	Paris	66 - 0	8	34
CMUnited99	1999	Stockholm	110 - 0	8	37
FC Portugal	2000	Melbourne	94 - 0	8	43
Tsinghuaeolus	2001	Seattle	90 - 1	12	44
Tsinghuaeolus	2002	Fukuoka	151 - 1	13	45

Table 2.1: Statistics from the simulated league.

2.1.1 CMUnited98

CMUnited98 was the team that introduced the terms formation and role into the world of RoboCup. By doing so, they added flexibility to the team. The players have flexible positioning within their roles, different roles have specific behaviours, they can make dynamic switches between formations and they have set plays(a formation in a certain situation) defined to know how to act in certain situations, eg. corner, kickoff, etc.

They use a *locker room agreement*, a mechanism for defining pre-determined multi-agent protocols accessible to the entire team, to specify an initial formation, an initial map from agents to roles and run-time triggers for dynamically changing the formation.

The flexible teamwork structure allows for task decomposition and dynamic role assignment. The formation is chosen based on goal difference and time left in the game, this is information that all players for certain have knowledge of. The role selection is a more complex task, since it may be difficult to determine if a role already has been taken by another agent. This is solved by implementing the behaviours according to the fact that one or more agents can possess the same role at the same time.

The team uses communication between the agents, but since the communication in RoboCup is not reliable in all situations, they can not have communication as the only option, they need the agents to be able to be totally autonomous [11].

2.1.2 CMUnited99

The changes that were made to CMUnited99 compared to CMUnited98(see 2.1.1) were mostly in the low level skills. The behaviours were improved upon and an online coach was added to shout out the exact ball position whenever there were an interruption in the game, such as kick in, kick off, offside, etc. None of these changes were of interest to this thesis since we focused on the management of the team [10].

2.1.3 FC Portugal

FC Portugal is the result of a cooperation project between the university of Averio and Porto in Portugal. FC Portugal is built upon CMUniteds low-level source code. Only small modifications were performed on the shooting and dribbling abilities of the players. The concept of formation and positioning introduced by CMUnited was extended by introducing strategy that included tactics, situations and player roles.

A Strategy includes a set of Tactics, predefined rules for activating the tactic, a set of roles that a player can possess and information concerning opponent modelling strategy, teammate modelling strategy and protocols for Communication.

A Tactic is defined by a set of formations, predefined rules for activating the formation. The valid tactic is selected from the global information, including: information about time current result and game statistics. For example if the game is near the end and the team is loosing a more risky and aggressive tactic is chosen.

A Formation is defined by the player positions inside the formation and the player role. The valid formation is chosen from situation information, including: ball possession, ball information and player information.

A Role is defined by a set of characteristics(behaviours) for players that include: strategic, ball possession and ball recover characteristics [6] [8].

2.1.4 Tsinghuaeolus

Team Tsinghuaeolus explains the importance of the basic skills a RoboCup team should possess, such as kick, dribble, etc. In the world championship in Seattle

2001 many teams had behaviours that were close to perfection. The main effort in building a team should be above the basic skills, and focus more on the management of the team. Team Tsinghuaeolus built their model with CMUnited (see 2.1.1, 2.1.2) as an initial reference. Since we did not find any report about Tsinghuaeolus' model, we started to review their source code and we found classes like formation and role, like all other teams we reviewed [13].

2.2 Our model

Modelling in RoboCup is usually done by altering the code. Our implementation allows you to alter in a database and achieve specific behaviours in certain situations. You can simply build a whole team from scratch, without having to write a single line of code, and let them play according to your tactic, with your formations and with the behaviours that you specify. If you feel you need extra rules or behaviours, then you need to implement those according to superclasses. The objects identified through our research are represented in figure 2.1 and explained in the following section.

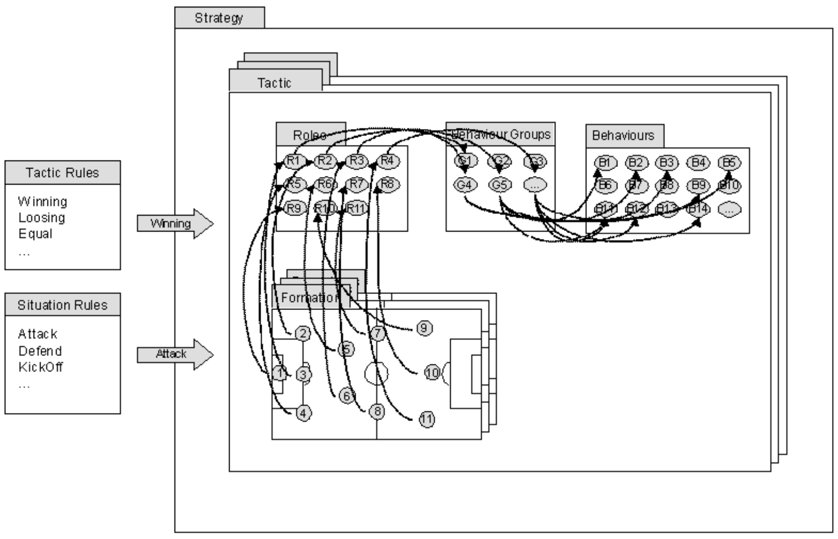


Figure 2.1: A visual representation of the model.

Planner The main part of the decision maker. Based on the current Strategy, Tactic and Situation a Formation is chosen. A set of Behaviours is then evaluated depending on which Role the individual player has in the current Formation. The Behaviour found to be most appropriate according to the evaluation is chosen for execution in the current cycle.

Strategy The Strategy is a high-level configuration of the team, allowing for different set-ups when meeting different teams. It is the wrapper class for all other classes, i.e. it contains all the tactics and is the only class known by the planner.

Tactic The Tactic is a configuration of Formations depending on the current situation in a game. Different Tactics can be used to customize the play when the team is at a disadvantage, compared to when it is leading. Each Tactic is controlled by a Tactic rule, which validates it and ensures that the whole team has the same Tactic. A Tactic rule should be based on information that is known, without any doubt, to every agent in the team, such as goal difference and the time left. To base the rules on ballposition and other more insecure data might result in different Tactics for different agents in the same team.

Situation A Situation consists of a various number of factors, e.g time left, goal difference, playmode, position of player, position of ball, position of opponent, etc. These factors need to be taken into consideration when the player chooses what action to perform. It is important to make sure that there is always one Situation available and only one, otherwise there will be a conflict and the right Formation might not be chosen. To make the Situation less complex in our model, we extracted basic information about time left and goal difference from the Situation rules and into the Tactic rules.

Formation A Formation is a set of Roles, that form the set-up of the team in a Situation, e.g. 4-3-3, 3-3-4, etc. It should have a distinct set of Roles. The Formation is chosen based on the current Situation and Tactic rule.

Role A Role is a set of Behaviours and a home position. A player should have one Role, and only one, at all times. The homeposition is the position on the playfield where the player starts, and holds close to, to keep his place in the Formation.

Behaviour Group A Behaviour Group is a collection of behaviours that a player can take in action.

Behaviour A Behaviour is the actual action taken by the players. The Behaviour class is the super class of all Behaviours and forces the subclasses to implement certain abstract methods for evaluation. We use *Electric Field Approach*, see Section 1.1.5, for evaluating the different Behaviours.

EFA A heuristic for action selection, used to evaluate the Behaviours. See section 1.1.5 for more information.

Database Used to store the configuration of Strategies, Tactics, Situations, Roles and Behaviours.

2.2.1 Configuration process

To be able to create a good team, you should have at least a situation rule for each of the different playmodes in the RoboCup server. You can add a situation rule for anything that is represented in the worldmodel, you just have to make sure that there is no conflict between the different rules, in which case it is not certain you will achieve the desired behaviour.
To simplify the process, we assume that all the behaviours, situations and tactic rules are implemented.

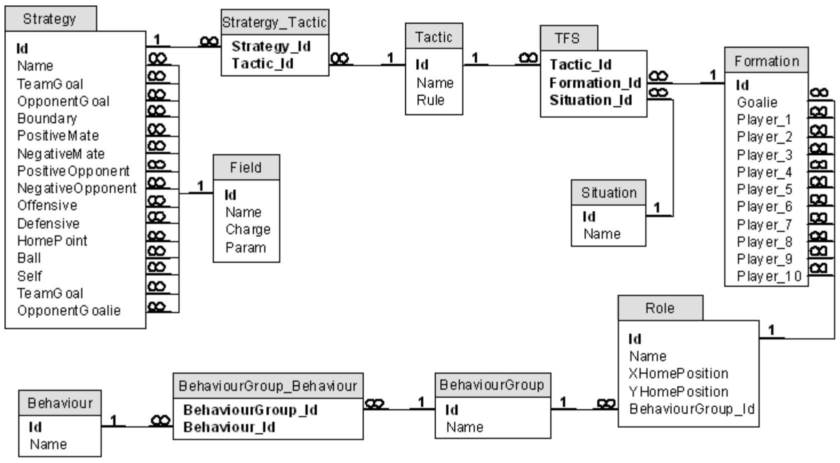


Figure 2.2: A visual representation of the database.

2.2.1.1 Use Case - Add a rule

This Use Case is specific for the Yaffa player. The model requires only that the name of the implemented rule has the same name in the database to be fired.

There a two types of rules, situation and tactic rules. They both inherit from the class `TeamYaffa.Yaffa.Rules.Rule` and they need to be in the correct path, `TeamYaffa.Yaffa.Rules.Rule.Situation` and `TeamYaffa.Yaffa.Rules.Rule.Tactic`. An example of the difference between the two could be that the tactic rules evaluate if your team is winning, losing or playing equal. The situation rules define what current playmode the server has and in what fieldhalf the ball is located.

1. Implement the class, in the right directory, inheriting from `TeamYaffa.Yaffa.Rules.Rule`.
2. Compile the new class.
3. In case of a situation rule, add the name of the class to the database table 'Situation' (without "`.cs`") and connect it to a formation by inserting the id of the situation in the 'TFS' table along with the corresponding formation and tactic id.
4. In case of a tactic rule, insert the name of the class (without "`.cs`") in the desired tactic row.

2.2.1.2 Use Case - Add a behaviour

This Use Case is specific for the Yaffa player. The model requires only that the name of the implemented behaviour must be the same as the name in the database.

The created behaviour must inherit from the class `TeamYaffa.Yaffa.ActionModel.Behaviours.Behaviour` and needs to be in the correct path, `TeamYaffa.Yaffa.ActionModel.Behaviours`.

1. Implement the class, in the right directory,inheriting from `TeamYaffa.Yaffa.ActionModel.Behaviours.Behaviour`.
2. Override the method `canHandle()`, preconditions to the behaviour.
3. Override the method `Evaluate()`, evaluation of the behaviour.
4. Override the method `Perform()`, what action to perform of this behaviour.
5. Override the method `Continue()`, continue the action of the behaviour.
6. Override the method `Reset()`, reset global values of the behaviour.
7. Compile the new class.
8. Add the name of the class to the database table '`Behaviour`' (without "`.cs`")

2.2.1.3 Use Case - Build a team

1. Build Strategy
   (a) Define all the field weights in the table '`Field`'.
   (b) Create a new row in table '`Strategy`' and point the field values to the corresponding field value.
2. Build Tactic
   (a) Define the tactic rule it will correspond to (see 2.2.1.1).
   (b) Define all the tactics with corresponding rules in the table '`Tactic`'
   (c) Link the strategy to the tactic(s) by entering the Tactic Id and Strategy Id in the table '`StrategyTactic`'.
3. Define Behaviour
   (a) Define the behaviour (see 2.2.1.2).
   (b) Define all the behaviours the team will use in the table '`Behaviours`'.
   (c) Define all the different behaviour groups the team will use, eg. defender, forward, goalie, etc, in the table '`BehaviourGroups`'.
   (d) Link the behaviours to the behaviourgroup in the table '`BehaviourGroupBehaviour`'.
4. Build Roles
   The x coordinate of the homeposition must be between -52.5 and 52.5, the y coordinate of the homeposition must be between -34 and 34. The behaviourgroup must be specified in the table `BehaviourGroup`.
   (a) Insert the x and y coordinate for the homeposition in the table '`Role`'

   (b) Link the role with the wanted behaviourgroup by entering the BehaviourGroup Id in the table '`Role`'.

5. Build Formation
   (a) Link wanted roles for the formation by entering the Role id for each player in the formation in the table '`Formation`'.

6. Define Situation
   (a) Define the situation rule that the formation will correspond to (see 2.2.1.1).
   (b) Define all the situations that the team will consider in the table '`Situation`'.

7. Link the model together
   'TFS' (Tactic Formation Situation), you must link a formation to a situation for each tactic you have created.
   (a) Link the formation with the tactic and the situation it will correspond to by entering the Formation Id in the table '`TFS`'.
   (b) Link the formation with the tactic by entering the Tactic Id in the table '`TFS`'.
   (c) Link the formation with the situation by entering the Situation Id in the table '`TFS`'.

8. Start your team by giving the strategyname you created as an input parameter.

2.2.1.4 Use Case - Alter a team

To make changes to the configuration of a team, you simply alter it in the database. To make the correct changes you follow the appropriate procedure in 2.2.1.3.

Chapter 3

Result

3.1 Architecture

Our application Yaffa (see 1.1.4) is built upon CRaPI (see 1.1.3). Yaffa starts by connecting a CRaPI player to the soccer server [3], and then continues by loading the team data from the database. A strategy object is created, which holds all the values of the fields to be used in the EFA and a reference to all the tactics that are included in this strategy. Each tactic has a rule (tacticrule) that indicates if it is currently valid or not. A Tactic holds a set of formations that also has rules (situationrules) to indicate whether they are to be used or not. A formation consists of eleven roles, each player is assigned to one specific role. By having all the roles in each agent, each player has knowledge about the other teammates in the current situation and can make more sophisticated predictions about how teammates will act. Each role has a set of behaviours and a home position. The home position is used to keep the formation together and the behaviours are evaluated by the EFA, see Section 1.1.5, each soccer server cycle to find out what we can do and what we do best. In Figure:3.1 we show the flow in the program.

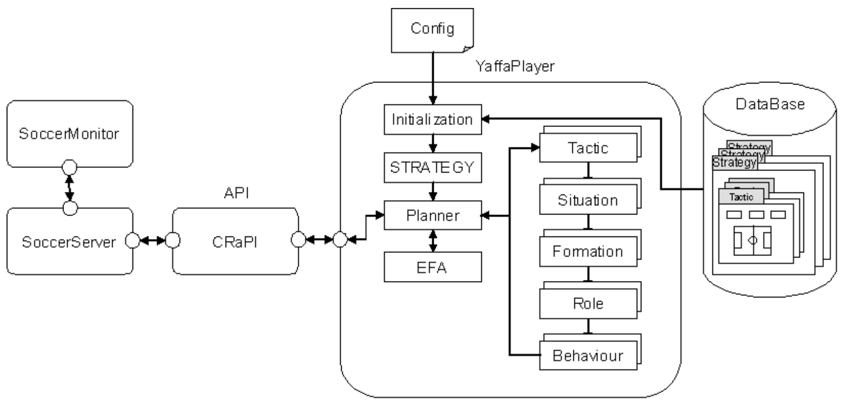


Figure 3.1: Yaffa Architecture

3.2 Series of tests

To be able to perform tests on the model, we used a database containing the situations presented in Appendix B.

The first task was to conduct a set of tests to make sure that the formation of the team worked and each player had a unique role. To do this we filled the database with different roles connected to different formations, and linked the formations to different situations.

We used a soccermonitor with support for coach messages, which allowed us to change playmode whenever suitable and made it possible to drag and drop the ball all over the field. The players had two behaviours each, LocalizeBall and RunHome (see Appendix A). This would make the players move according to where the ball was located on the field but not trying to intersect it. The players would also position themselves according to the current playmode, such as corner_kick, kick_off, etc. The experiment went as expected and the players ran as supposed (see figures: 3.2, 3.3, 3.4, 3.5, 3.6).

The second task was to verify that the formations held together during a regular game, with all the different behaviours the different roles included. This is a non trivial test, as always in the RoboCup domain, due to the many factors of uncertainty. For instance, a minor error in a behaviour or a miscalculated judgement of the importance of the behaviours in the EFA can cause strange performance of the player, and the sources of the problem can be many. However, we played against the world champions 2002 (see 2.1.4) and the configuration worked as expected. We did not perform well, due to our poorly trained low level behaviours, e.g. kick, pass, etc.

3.3 Evaluation

3.3.1 Maintainability assessment

To asses the model a Software Engineering process based on industrial practice, called Scenario-based assessment by Jan Bosch, was chosen.

Scenario based assessment is particularly useful for development of quality attributes. Quality attribute such as maintainability can be expressed very naturally through change scenarios.

To perform this process a typical change profile for the model is created. This profile will then be used to evaluate the effort required to adapt the model to the new situation, a so called impact analysis (see Table 3.1).

The measurement of the effort will be the number new lines of code to the adaption divided by the total lines of code for the system (15858). Low effort implies high maintainability for the model [2].

3.3.1.1 The Profile

A selected profile [2] was selected for the assessment. A detailed description of how the tests where conducted, see 2.2.1.3.

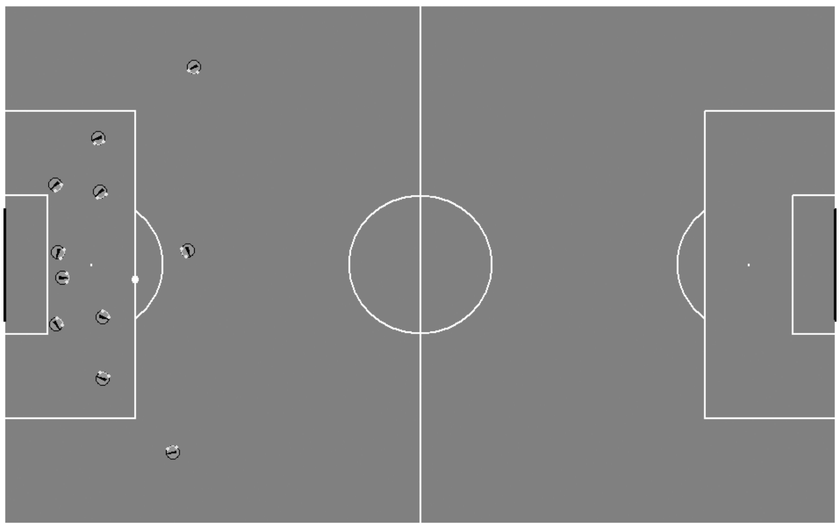


Figure 3.2: When the situation rule Ball_Left_Left is true, we configured the players to be very defensive.

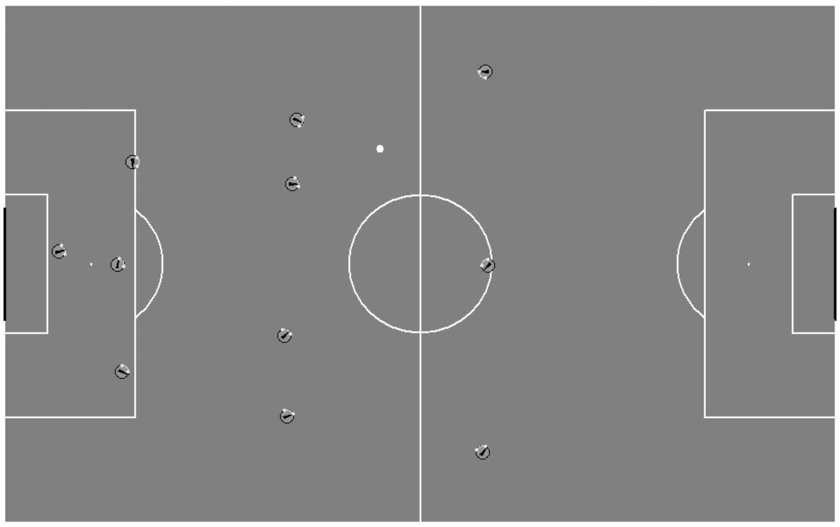


Figure 3.3: When the situation rule Ball_Left_Right is true, the defenders stay behind while the forwards run in offensive mode and wait for passes from the middlefielders.

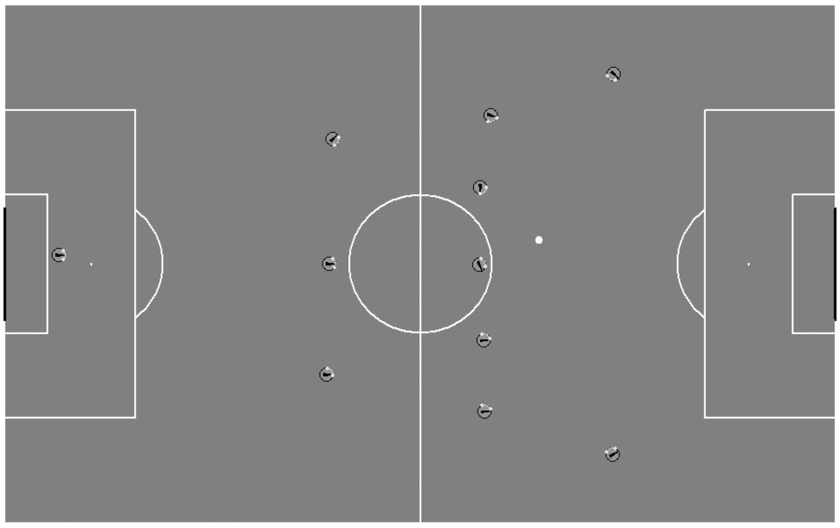


Figure 3.4: When the situation rule Ball_Right_Left is true, the players in this configuration change formation from 3-4-3 to 3-5-2.

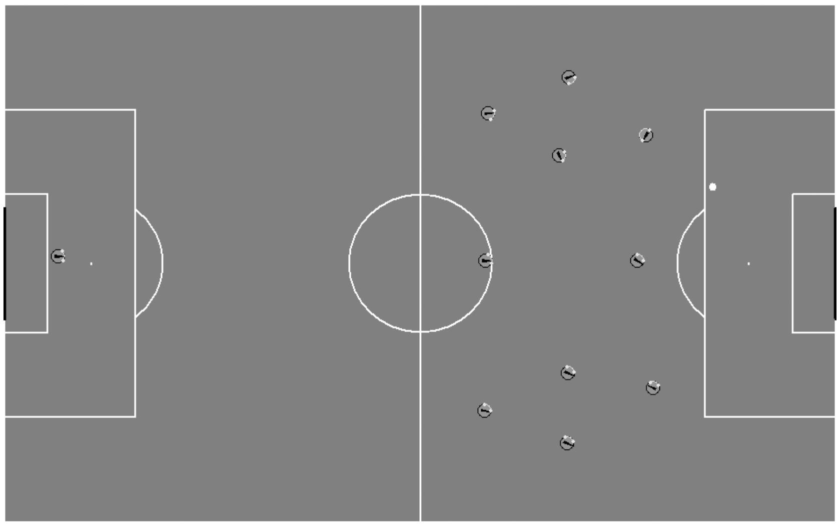


Figure 3.5: When the situation rule Ball_Right_Right is true, the players go back to 3-4-3.

Profile	LOC	Effort
Add new Tactic	0	0
Add new Tactic Rule	12	0,0004
Alter Tactic Rule	1	0,00006
Add new Formation	0	0
Add new Role	0	0
Add new Behaviour	70	0,0044
Add new Situation Rule	12	0,0004

Table 3.1: Statistics from scenario-based assessment.

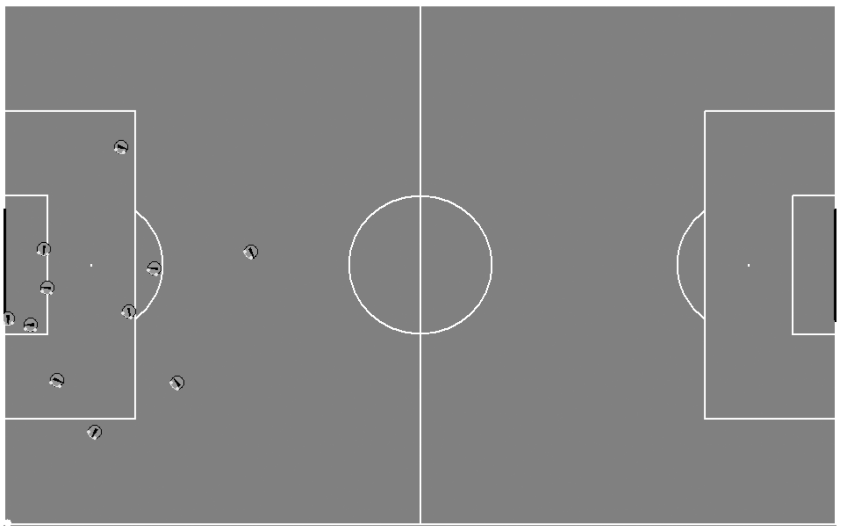


Figure 3.6: The team's formation when defending a corner kick, of course the opponents would cause the players to move around and block them.

Chapter 4

Discussion

4.1 Results

The experiments were successful, if we see to the functionality of the model. The agents behaved as expected and held the formations together. A disappointment is our poorly developed low-level behaviours, if compared to the world champions. To be able to contest in the higher division you will need perfectly developed behaviours, trained by machine learning algorithms due to the inconsistent data in the perception from the soccer server.

We used CRaPI as a platform for Yaffa since we were a part of the CRaPI-team. We did not believe that the behaviours would be as far behind the champions as they were.

The model is relaying upon the user to assure that there is no conflict between the different rules. Even though the model accepts conflicts, it might not be a good solution when the team configuration grow big.

The model is following the development of the recent winners of the world championship in the simulated RoboCup league. When CMUnited won in 1998 they introduced formations and roles into the league, and since then you will find similar objects in a team.

Role selection in our model is non-functional, the agents get a role id upon startup and play in that role the entire game. We have no communication protocol implemented, which we believe is a vital part for a successful implementation of role selection. However, we considered it during the design phase, if a communication engine and role selection logic are added, the implementation will not need much alteration.

4.2 Problems

- Our agents seemed slower than the champions when we ran the teams under the same circumstances. We believe the main reason for this to be that our implementation is in C# while the champions uses C++.

Another reason might be that our implementation is quite extensive and demands a lot of memory to be able to have a dynamic play.

- Due to lack of computing power, we had a hard time testing. The synchronization between the client and the server was not always syncronized while testing, which led to misleading behaviours of the agents.
- Building a team configuration in the database becomes quite complex when it grows big, due to all the relationships among the entities in the database.
- To test and evaluate a RoboCup simulated league team are hard since there are a lot of factors that can interfere. E.g.
  - A team has eleven different agents with their own perception of the world.
  - Situation assessment might differ between agents in the same situation.
  - Certain situations can be hard to invoke, e.g. if the situation is depending on the positioning of the opponents.

Chapter 5

Conclusion

A model can be built, with consideration on maintainability, to represent a team configuration in the simulated RoboCup domain.

The teams in RoboCup are taking on values from real life soccer as the development proceeds. In order to be on top, fighting for a title in the world championships, you need not only perfect behaviours but also a good management of the team.

A situation is everything an agent perceive. You need to 'divide and conquer' the situation into different levels to decrease the complexity. Based on these levels the agents should be able to extract how to perform. In our model, we present one general solution of how to represent domain knowledge and its relationship to the agent.

5.1 Future work

- A graphical user interface to configure the database. You need a visual representation of the data when the complexity grows, otherwise there is a great chance of faults in the configuration that might be difficult to trace.
- Find a method to avoid conflicting rules.
- Optimize the low-level skills, such as kick, pass, etc.
- Enhance the precision of the world model.
- Add memory to the world model, and a model of how to decrease the confidence value for an object depending on the time since it was last perceived.
- Add role selection logic.
- Add communication protocol.
- Add execution plans for 'ahead' planning in the agents.
- Apply our model to the RoboCup four-legged league [4].

Chapter 6

Acknowledgements

Stefan Johansson, Błażej Żak, Martin Hylerstedt, Anders Olofsson, Kenneth Henningsson, Markus Bergkvist, Tobias Olandersson.

Bibliography

[1] http://www.robocup.org/games/31.html.

[2] Jan Bosch. *Design & Use of Software Architectures Adopting and evolving a product-line approach.* Addison-Wesley, 2000.

[3] Mao Chen, Ehsan Foroughi, Fredrik Heinz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. *User Manual, RoboCup Soccer Server*, 2002.

[4] The RoboCup Federation. http://www.robocup.org/. last checked 2003-05-22.

[5] Stefan Johansson and Alessandro Saffiotti. Using the electric field approach in the RoboCup domain. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, number 2377 in LNAI, pages 399–404. Springer-Verlag, Berlin, DE, 2002. Online at http://www.aass.oru.se/˜asaffio/.

[6] Nuno Lau Luís Paulo Reis and Eugénio Costa Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents, 2001.

[7] David W. Payton, J. Kenneth Rosenblatt, and David M. Keirsey. Plan guided reaction,. *IEEE Transactions on Systems, Man and Cybernetics, No. 6, November/December 1990*, 20:1370–1382, 1990.

[8] Luís Paulo Reis and José Nuno Lau. Fc portugal team descripton: Robocup 2000 simulation league champion paper, 2001.

[9] Stuart Russel and Peter Norvig. *Artificial Intelligence a Modern Approach.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1995.

[10] Peter Stone, Patrick Riley, and Manuela Veloso. The CMUnited-99 champion simulator team. In Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Berlin, 2000. Springer Verlag.

[11] Peter Stone and Manuela M. Veloso. Task decomposition and dynamic role assignment for real-time strategic teamwork. In *Agent Theories, Architectures, and Languages*, pages 293–308, 1998.

[12] Team Yaffa. http://crapi.sourceforge.net/. last checked 2003-05-24.

[13] Cai Yunpeng Yao Jinyi, Chen Jiang and Li Shi. Architecture of tsinghuaeolus. Technical report, State Key Lab of Intelligent Technology and System, Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P.R.China, 2001.

Appendix A

Behaviours in Yaffa

This appendix describes the behaviours we have used during our experiment. Mate and Opponent charges can have either positive or negative charge. The charges for the Ball, the HomePoint and the OpponentGoal charges are always positive, whereas the TeamGoal and the Boundary charges are always negative.

Offensive and Defensive are special purpose charges, with linear spread functions. The Offensive field has a negative charge in the team goal and a positive charge in the opponent goal, and the opposite for the Defensive field.

BlockOpponent Interfere an opponent when the other team is in possession of the ball.

> **Charges** Mate[-], Opponent[+], Ball, HomePoint, Defensive.
>
> **Probes** Predicted positions of the player, next to the blockable opponents.

DribbleWithBall Advance without losing control of the ball.

> **Charges** Mate[+], Opponent[-], Self, Offensive, Boundary TeamGoal, OpponentGoal.
>
> **Probes** A set of dribble points on the offensive side of the player, which are predicted possible positions of the ball, when the behaviour is done executing.

GetPassable Make sure to be positioned where there is a free pass-path from the ball-keeper, when the ball is in our possession.

> **Charges** Opponent[-], HomePoint, Offensive, Ball Boundary.
>
> **Probes** Predicted possible positions of the player, when the behaviour is done executing.

IntersectBall Run to the closest possible intersection-point with the ball-trajectory.

> **Charges** Mate[-], Opponent[+], Ball, HomePoint, Defensive.
>
> **Probes** Predicted position of the player next to the ball.

LocalizeBall *This behaviour is currently not incorporated in the EFA-model.* It localizes and focuses on the ball. Chosen for execution when no other behaviour is applicable.

Charges None

Probes None

LocalizeMate *This behaviour is currently not incorporated in the EFA-model.* It localizes and focuses on the last seen team mate. Chosen for execution when no team mate is seen and no other behaviour than LocalizeBall is applicable.

Charges None

Probes None

PassTheBall Passes the ball to a team mate.

Charges Mate[+], Opponent[-], Self, Offensive, TeamGoal, Opponent-Goal, Boundary.

Probes Predicted position of the ball when the behaviour is done executing.

RunHome Runs to the players home-position.

Charges Mate[-], Opponent[+], Ball, HomePoint, Offensive.

Probes The player at the home-position.

ShootAtGoal Shoots the ball to the most appropriate position within the goal.

Charges Mate[+], Opponent[-], Self, Offensive, OpponentGoal.

Probes The Ball.

KickOut *This behaviour is currently not incorporated in the EFA-model.*Take care of the play modes corner kick and kick in. Kicks the ball to a team mate.

Charges None

Probes None

BeforeKickOffMove *This behaviour is currently not incorporated in the EFA-model.*Moves the player to its start position.

Charges None

Probes None

GoaliCatch Catch the ball.

Charges Mate[-], Opponent[+], Ball, Defensive, HomePoint, Self.

Probes The Ball.

GoalieIntersectBall Run to the closest possible intersection-point with the ball-trajectory.

Charges Mate[-], Opponent[+], Ball, Defensive, HomePoint.

Probes The Ball.

GoalieKickOutMove Make sure to be positioned where there is a free area of players.

Charges Mate[-], Opponent[-], Ball, Defensive, HomePoint.

Probes Self.

GoaliePassTheBall Passes the ball to a team mate

Charges Mate[+], Opponent[-], Offensive, TeamGoal, OpponentGoal, Self.

Probes Predicted positions of the ball when the behaviour is done executing.

GoalieStandStrategic Passes the ball to a team mate

Charges HomePoint, Ball, Defensive.

Probes Predicted position of the player when the behaviour is done executing.

Appendix B

Situations in Yaffa

This appendix describes the situations we used during our experiments. The situation name and the conditions are mentioned, where Side is either the character 'l' or 'r'. If the playmode equals free_kick_l and my side equals 'l' then free_kick_OurSide becomes true.

Ball_Left_Left Ball.X $\leq$ -26 $\wedge$ playmode $\equiv$ play_on

Ball_Left_Right -26 $<$ Ball.X $\leq$ 0 $\wedge$ playmode $\equiv$ play_on

Ball_Right_Left 0 $<$ Ball.X $\leq$ 26 $\wedge$ playmode $\equiv$ play_on

Ball_Right_Right 26 $<$ Ball.X $\wedge$ playmode $\equiv$ play_on

Before_Kick_Off playmode $\equiv$ before_kick_off

Corner_Kick_Our_Upper Ball.Y $<$ 0 $\wedge$ playmode $\equiv$ corner_kick_OurSide

Corner_Kick_Our_Lower Ball.Y $>$ 0 $\wedge$ playmode $\equiv$ corner_kick_OurSide

Corner_Kick_Opponent_Upper Ball.Y $<$ 0 $\wedge$ playmode $\equiv$ corner_kick_OpponentSide

Corner_Kick_Opponent_Lower Ball.Y $>$ 0 $\wedge$ playmode $\equiv$ corner_kick_OpponentSide

Goal_Kick_Our playmode $\equiv$ goal_kick_OurSide

Goal_Kick_Opponent playmode $\equiv$ goal_kick_OpponentSide

Kick_In_Our_LL Ball.X $\leq$ -26 $\wedge$ playmode $\equiv$ kick_in_OurSide

Kick_In_Our_LR -26 $<$ Ball.X $\leq$ 0 $\wedge$ playmode $\equiv$ kick_in_OurSide

Kick_In_Our_RL 0 $<$ Ball.X $\leq$ 26 $\wedge$ playmode $\equiv$ kick_in_OurSide

Kick_In_Our_RR 26 $<$ Ball.X $\wedge$ playmode $\equiv$ kick_in_OurSide

Kick_Off_Our playmode $\equiv$ kick_off_OurSide

Kick_Off_Opponent playmode $\equiv$ kick_off_OpponentSide

Offside_Our playmode $\equiv$ offside_OurSide

Offside_Opponent playmode $\equiv$ offside_OpponentSide

Time_Over playmode $\equiv$ time_over

Kick_In_Opponent_LL Ball.X $\leq$ -26 $\wedge$ playmode $\equiv$ kick_in_OpponentSide

Kick_In_Opponent_LR -26 $<$ Ball.X $\leq$ 0 $\wedge$ playmode $\equiv$ kick_in_OpponentSide

Kick_In_Opponent_RL 0 $<$ Ball.X $\leq$ 26 $\wedge$ playmode $\equiv$ kick_in_OpponentSide

Kick_In_Opponent_RR 26 $<$ Ball.X $\wedge$ playmode $\equiv$ kick_in_OpponentSide

Free_Kick_Our_LL Ball.X $\leq$ -26 $\wedge$ playmode $\equiv$ free_kick_OurSide

Free_Kick_Our_LR -26 $<$ Ball.X $\leq$ 0 $\wedge$ playmode $\equiv$ free_kick_OurSide

Free_Kick_Our_RL 0 $<$ Ball.X $\leq$ 26 $\wedge$ playmode $\equiv$ free_kick_OurSide

Free_Kick_Our_RR 26 $<$ Ball.X $\wedge$ playmode $\equiv$ free_kick_OurSide

Free_Kick_Opponent_LL Ball.X $\leq$ -26 $\wedge$ playmode $\equiv$ free_kick_OpponentSide

Free_Kick_Opponent_LR -26 $<$ Ball.X $\leq$ 0 $\wedge$ playmode $\equiv$ free_kick_OpponentSide

Free_Kick_Opponent_RL 0 $<$ Ball.X $\leq$ 26 $\wedge$ playmode $\equiv$ free_kick_OpponentSide

Free_Kick_Opponent_RR 26 $<$ Ball.X $\wedge$ playmode $\equiv$ free_kick_OpponentSide

Goal playmode $\equiv$ Goal_AnySide

Our_Goalie_Caught_Ball playmode $\equiv$ goalie_catch_ball_OurSide

Opponent_Goalie_Caught_Ball playmode $\equiv$ goalie_catch_ball_OpponentSide

Free_Kick_Fault_Our playmode $\equiv$ free_kick_fault_OurSide

Free_Kick_Fault_Opponent playmode $\equiv$ free_kick_fault_OpponentSide

Back_Pass_Our playmode $\equiv$ back_pass_OurSide

Back_Pass_Opponent playmode $\equiv$ back_pass_OurSide

Appendix C

Implementation of EFA

Our implementation is based on two main components, the Engine and the FieldGenerator. The responsibility for generating the fields is assigned to the FieldGenerator, whereas all electric field calculations are handled by the Engine.

The main aspect of our design is to let the FieldGenerator pre-calculate as much as possible before an actual game begins. Since the interaction between electric fields is only a matter of superposition[1], electric fields for all specified types of charges can be pre-calculated. To improve the performance of the superposition, the fields are implemented as layers of matrices in a *composite electrical field*, hence no complete superposition of the fields are made, only the probed position is superimposed.

The generation of a field surrounding a charge is made in three steps. The first step is to calculate the size of the matrix that will represent the field. This is done by calculating the radius around the charge where the potential reaches a predefined threshold value, all values below this threshold value is ignored in further calculations. The radius is calculated by applying a distribution formula, which loosely mimics the potential distribution around real electric charges[2]:

$$p = \frac{p_{center}}{d^k} \tag{C.1}$$

where p is the potential at distance d from p_{center} and k is the decrease constant. Extracting d from equation (C.1) we get the formula for the radius:

$$r = \sqrt[k]{\left(\frac{p_{center}}{p_{threshold}}\right)} \tag{C.2}$$

where r is the radius of the charge and $p_{threshold}$ is the threshold potential. The second step of the field calculation is to calculate the potential for each cell in the field matrix. These calculations uses equation (C.1) with distance d set to the distance in squares from the center multiplied with the size of each square.

[1]Superposition of electric fields means that the potential in a point p is the sum of the potentials for all fields in point p.

[2]The potential distribution around real electric charges is described by Coulumbs law. Equation (C.1) is equivalent to Coulumbs law when $k = 2$

The third and final step is to use the concept of superposition to add together all the charges in the current field.

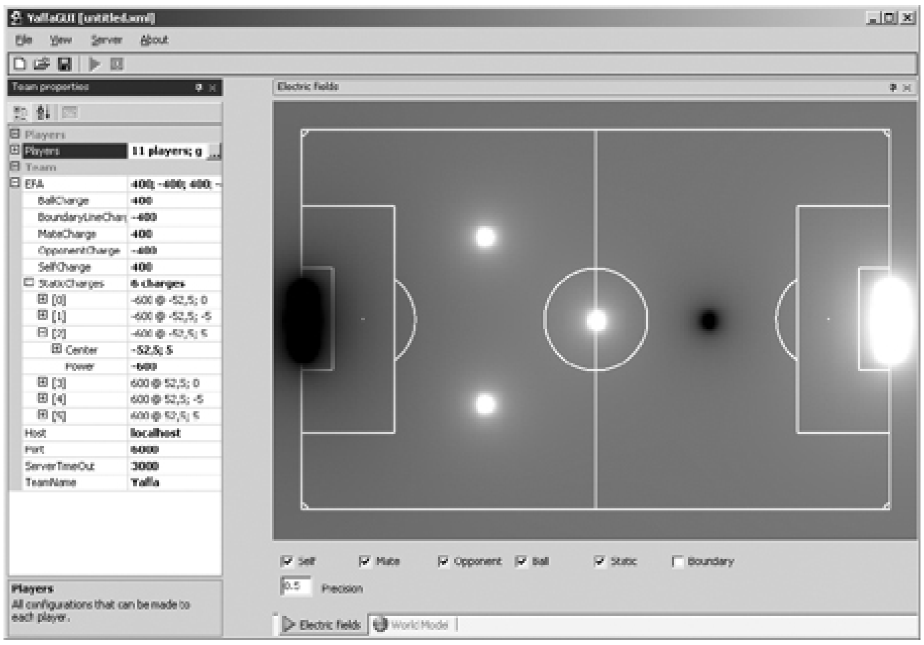
Apart from field generation based on potential distribution around artificial charges, we also utilize a special purpose field generation of the boundary lines. The purpose of this field is to defer the players from running outside the field, but without setting an absolute condition. This field uses the same distribution formula as equation (C.1). The difference between this calculation and the standard charge-based calculation is the way in which the potentials are distributed. Since the boundary line field is designed to simulate a set of charges placed with infinite density around the field, the potentials are decreasing only inwards on the field, without using superpositioning.

Visualization of EFA

During the development of *CRaPI* [12], we learned that a tool to visualize the view of the agent can be very useful. With such a tool we can confirm that the agent has the view it is supposed to have. The tool we developed to visualize the Electric Field Engine offers the following functionality:

- Configure the charges of the predefined fields that are generated for the boundary line, a mate, an opponent, the ball and the agent.
- Position and configure static charges, e.g. the goal-charges.
- Compare the strength of the charges. You should be able to see both the relative strength and the propagation of the charge.

In Figure C.1 some of the fields that can be configured with the Electric Field Generator are visualized.



Players	
Players	11 players; g
Team	
EFA	400; -400; 400; -
BallCharge	400
BoundaryLineChar	-400
MateCharge	400
OpponentCharge	-400
SelfCharge	400
StaticCharges	6 charges
[0]	-600 @ -52,5; 0
[1]	-600 @ -52,5; -5
[2]	-600 @ -52,5; 5
Center	-52,5; 5
Power	-600
[3]	600 @ 52,5; 0
[4]	600 @ 52,5; -5
[5]	600 @ 52,5; 5
Host	localhost
Port	6000
ServerTimeOut	3000
TeamName	Yalla

Players
All configurations that can be made to each player.

[✓ Self] [✓ Mate] [✓ Opponent] [✓ Ball] [✓ Static] Boundary

Precision: 0.5

Figure C.1: Visualization of the Electric Field Generator