A Subsumption Architecture for Robotic Soccer

Valnir Ferreira Junior, John Thornton, and Joe Leonard
School of Information Technology, Faculty of Engeneering and Information Technology
Griffith University Gold Coast Campus
Parklands Drive, Southport, Queensland, 4215, Australia
{v.ferreira, j.thornton, j.leonard}@mailbox.gu.edu.au

Abstract: The purpose of this paper is to report on our investigation about the feasibility of implementing a pure subsumption architecture for controlling our team of Mirosot Robotic Soccer players, the RoboCoasters. This extends previous work on subsumption by considering a highly dynamic and competitive environment where speed and accuracy are paramount.
The major finding of our study is that a rigid adherence to the incremental development concept of subsumption has proved impractical when applied to our single architecture approach to Robotic Soccer. This finding has led us to propose two extensions to subsumption that simplify the development process and address the need for continual improvement in lower level behaviours. These are: (a) The creation of a chief behaviour as a permanent top layer for the architecture; and (b) the development of modules that can be changed within a layer without changing the overall layer's structure.

Keywords: Subsumption architecture, Robotic Soccer, Behaviour-Based Robotics

1. Introduction

The subsumption architecture [1] is a substrate used to build Behaviour-Based mobile robot control systems, which possesses the following main characteristics: (a) It decomposes the problem into horizontal behaviour-producing layers, as opposed to the Classical AI approach of decomposing the problem into vertical functional modules; (b) The failure of a higher level layer or an unexpected change in the world should not cause the robot to collapse, since lower level layers are continuously operating in parallel, and there is no central control; and (c) It provides a modular and incremental way to build a highly reactive mobile robot control system, a characteristic also known as incremental development. The subsumption architecture has the potential to create systems that hold at least two desirable characteristics for a Robotic Soccer architecture: Robustness, and incremental development.

We begin by presenting an overview of our current system in Section 2. In Section 3 we discuss a number of issues which influenced the way we designed the final architecture. In Section 4 we present our seven-level subsumption architecture. In Section 5 the results obtained from empirically testing this architecture on a set of existing Robotic Soccer benchmark problems are shown, along with a qualitative evaluation of our implementation. Finally, Section 6 offers some conclusive remarks for our study, and defines several areas for further research.

2. System Overview

We have two teams consisting of three Mirosot compliant [2] players each, plus a spare player. Our main team, the RoboCoasters, is controlled by a Windows98 based 933 MHz Pentium III PC with 512 Mb RAM. The player's motors are controlled by an Intel 80C296SA processor which is used to translate velocity commands received by the onboard Radiometrix RF module receiver. A corresponding RF module transmitter is used to broadcast the velocity information to the robots from the host computer [3].

The original software system used to control the Robotic Soccer players is written in Microsoft's Visual C++ 6.0. This software provides a set of controlling functions that can be called from a main control loop 60 times every second. This repetition rate is dictated by the frame grabber's field acquisition rate.

We use a global vision system consisting of an overhead CCD camera, a frame grabber, a PCI MGA video card and the image processing algorithm. The Pulnix TMC-7DSP camera captures a global 640 x 480 pixel picture of the pitch at the full NTSC rate (60 Hz). These pictures are continuously relayed by our Matrox Meteor II frame grabber to video RAM.

The vision processing algorithm currently takes around 7 ms to accurately track and identify the seven objects (three home robots, three opponent robots and the ball) by using both shape and colour recognition. A set of positioning variables for the identified objects is then made available to the strategic module of our system. These variables consist of the Cartesian coordinates for the robots and the ball, as well as the orientation for the robots.

3. Design Issues

In this section, a number of practical and theoretical design issues raised during the development of our subsumption architecture are briefly presented and discussed.

3.1. Central control

We decided that in order to increase the team's effectiveness, we would need to enable the players to swap their roles dynamically during a match, so we identified three roles for our team: Goalkeeper, defender and attacker. Our problem then became one of how to allocate roles to players without having an explicit central controller within the architecture. Also, given the importance of role assignment within our strategy, we decided that these roles had to be assigned to the players at the earliest possible stage in a given sampling period. In subsumption terms, this meant that the dynamic assignment of roles would have to be taken care of within the first behaviour producing layer of the architecture. To address this problem, we implemented a ROLE DECIDER module in the first layer (or level 0) of our subsumption architecture, as we will later show in Section 4.

3.2. Multi-agency versus single-agency

In so far as this implementation of the subsumption architecture is concerned, we see our team as a single agent, sensing the environment through its sensors (the CCD camera), and acting upon the environment through a set of effectors (the three robots on the pitch). Our agent has a number of goals, of which the most relevant one is to place the ball inside the opponent's goal while not allowing the opponent to place the ball inside our goal.

When we looked at having a separate subsumption architecture for each of the players, as opposed to our choice of having a single subsumption architecture to control a team of three players, the suitability of a subsumption architecture to our domain came into question. This is mainly due to the fact that, as already pointed out by others [4], it is not possible to implement a subsumption architecture to control a truly Multi-Agent team without some modifications to the architecture's original specifications.

As far as our domain is concerned, in order to accommodate role-swapping in a truly Multi-Agent system, every player's architecture would have to be able to produce the behaviours that are expected from the three different roles in our team, which would result in what we refer to as *redundant parallelism*. Furthermore, such approach would also require the existence of a central controller to integrate the functionality of the three single architectures. This idea is discussed in more detail in Section 3.4, where we introduce the idea of a *subsumption of subsumptions*.

3.3. Scalability

The exercise of adding a new layer to our system was not complicated. We attribute this to the inherent simplicity of the subsumption architecture, brought about by having increasingly competent independent layers competing for the right to output a certain behaviour during a given sampling interval.

3.3.1 Evolutionary ceiling

We put forward the idea that a control system should be expected to have its applicability and effective usage bound by what we refer to as an *evolutionary ceiling*.

Our architecture is intended to efficiently control a three-a-side Mirosot team, and not a five-a-side or even an eleven-a-side team, if such a league existed. The experience gained from the implementation part of this study has led us to believe that our architecture is highly scalable, but like in evolution, every system eventually needs to be fully replaced when a completely new level of competency (one that we are unable to either think of or make an allowance for at the present) is to be achieved.

3.3.2 Improving existing behaviours

Another characteristic that assisted us to identify scalability as one of the great strengths of the subsumption architecture was the practical issue of having to modify existing modules in order to improve their effectiveness. During our tests, it became clear that the effectiveness of the overall team depended significantly on the actual output of the modules contained within the architecture. In the domain of Robotic Soccer, we must be able to modify a module's internal structure (its code) many times in order to achieve optimal performance. This situation can take place during ongoing research, or during a competition, in which case we must be able to perform the necessary modifications to a module, and then *plug* it back into the architecture as quickly and as easily as possible.

We believe this would not contravene the incremental development concept contained in the original subsumption architecture, as we would only be modifying (or adding to) a module's internal code, and not to any of its input/output lines or inhibition/suppression nodes. Furthermore, this ability to easily modify a given part of the controller provided for in our proposed architecture is surely a highly desirable characteristic within the Robotic Soccer domain.

3.3.3 Electing a chief behaviour

During our development, we found that in all circumstances if a robot is about to hit a wall we would want a boundary avoidance layer to subsume all other behaviours. This caused us to elect boundary avoidance as the *chief behaviour*, and to place it as the topmost layer in the architecture, rather than implement a boundary avoidance behaviour in each layer, which would be unnecessarily complex.

A problem arises whenever we need to add a new layer to enhance the system's performance, as according to subsumption theory, such a layer should be placed on top of the existing architecture, thus demoting our former topmost layer to the status of second topmost layer. This situation is undesirable, as it is imperative that our chief behaviour remains on the very top of the architecture if the system is to work efficiently.

In practice, this problem can be circumvented by inserting the new layer immediately below the topmost layer, as shown in Figure 1. In the figure, step one represents an abstract subsumption architecture with n layers, augmented with the topmost layer representing the chief behaviour. In step two, the need arises to introduce a new layer to the system, however, for efficiency reasons, the chief behaviour must be kept as the highest level of competence (or on top of the architecture). The new layer is inserted immediately below it. Step three shows the resulting architecture with the new layer placed as the second topmost layer.

This clearly violates the incremental development concept, and it represents a serious limitation to the suitability of implementing it to the domain of Robotic Soccer, where this situation tends to arise frequently.

This does not, however, represent a significant hindrance to the use of the subsumption architecture in Robotic Soccer at the practical level. As we discovered, the violation operation described in Figure 1 can be performed quite easily to achieve the goal of maintaining the chief behaviour on top, and to improve the system's performance by adding a new layer. This situation would not be possible without the violation.

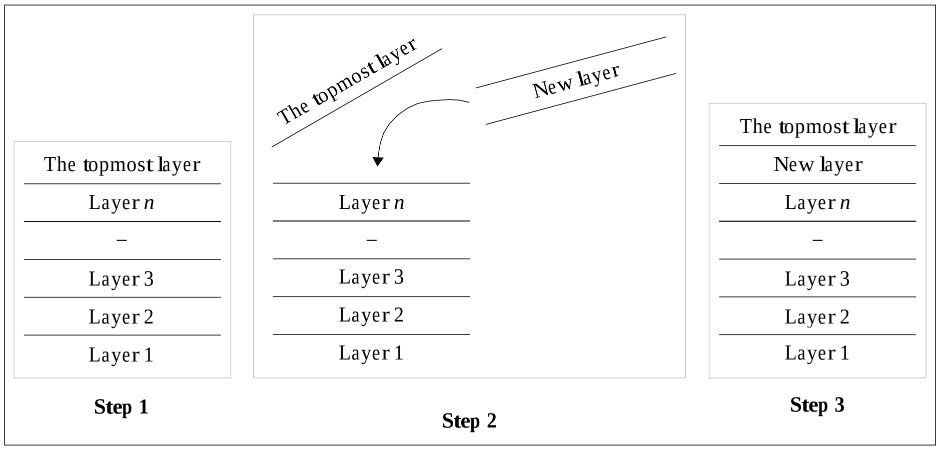

Fig. 1. Violating the incremental development concept.

3.4. Alternative approaches

During our evaluation, we encountered two other approaches that could represent alternatives to the implementation of a subsumption architecture to our team. Behaviour selection is based on the work of Ronald Arkin [5], and was successfully implemented by the MASKARO team [6], the Mirosot league champions of FIRA's Robot World Cup 2001.

MASKARO's approach consists of a number of predetermined behaviours, which can be assigned to the robots in the team according to the current state of the world, and with respect to a given task. Central to this approach are the concepts of actions and behaviours. Actions are the low level components of the controller, while behaviours are made up of one or more actions. The behaviour selector runs the algorithm to, at every sampling interval, dynamically allocate one behaviour to each one of the robots on the team. This therefore represents the central controller that is not permitted if one is to implement a pure subsumption architecture. The existence of a central controller, together with the dynamic role assignment are the main features of this approach. The MASKARO team's recent successful campaign in the Robot World Cup 2001 serves to illustrate the desirability of dynamic role assignment for achieving success in the Mirosot league.

We also investigated the possibility of having three subsumption architectures (one for each robot) being controlled by a master subsumption architecture. This means that we would have a set of three robots, each controlled by a subsumption architecture, along with a subsumption architecture to integrate the single architectures; rather than a soccer system controlled by a subsumption architecture. We refer to this alternative as a subsumption of subsumptions. It soon became apparent that it would not be possible to implement this approach without adding some extensions to the original subsumption architecture. This view is shared in [4]. To achieve dynamic role assignment under a subsumption of subsumptions, we realised that every robot's architecture would need to compute the calculations required for the output of each one of our team roles. This would mean that having a subsumption architecture for each robot would result in redundant parallelism, which clearly represents an undesirable characteristic of any software system.

4. Implementation

In this section we present a detailed description of the final architecture, focusing on the incremental development process of building one behaviour-producing layer at a time to achieve higher levels of competence. The architecture presented here was implemented in C within the Microsoft Visual C++ environment. All concepts contained within the original subsumption architecture description [1] were maintained in order to guarantee the scientific validity of our study.

The architecture presents seven behaviour-producing layers, which correspond to the levels 0 (POSITION), 1 (GOALKEEPER), 2 (CHASE), 3 (DEFEND), 4 (ATTACK), 5 (PASS), and 6 (AVOID). This architecture is shown in Figure 2. We use the terms *layer*, *level*, and *level of competence* interchangeably.

Each layer can have one or more modules. For example, layer 0 has five modules (namely VISION, PLACE, MOTOR, ROLE DECIDER, and ROLE ENFORCER), layer 1 has two modules (BALL PREDICTION, and GOALKEEPER CONTROL), and so on. There are four main wires in the architecture: *positions*, *ball prediction*, *roles*, and *command*. Also present are the wires *role history*, and *Broadcast(command)*. As shown in Figure 2, in compliance with subsumption architecture theory, a lower level layer never modifies the state (or any variable values) of any of the modules in higher level layers.

The ovals with an S inside their lower portion represent suppression nodes. The dashed lines represent the end of one layer and the beginning of a new one. ROBOTS represents the physical robots on the pitch receiving the final wheel velocity commands contained in *command*.

The *positions* wire represents the x and y coordinates for the team robots and for the ball, as well as the orientation for the robots. These are extracted by the vision system every sampling interval (approximately every 16 ms). The wire is implemented as two arrays, one of size 4 x 2 used for the coordinates, and another of size 3 used for the orientations.

The *ball prediction* wire represents the predicted x and y coordinates for the ball n fields ahead, where n is an integer number representing the number of fields ahead we would like to predict the position of the ball.

The *roles* wire carries the information that assigns mutually

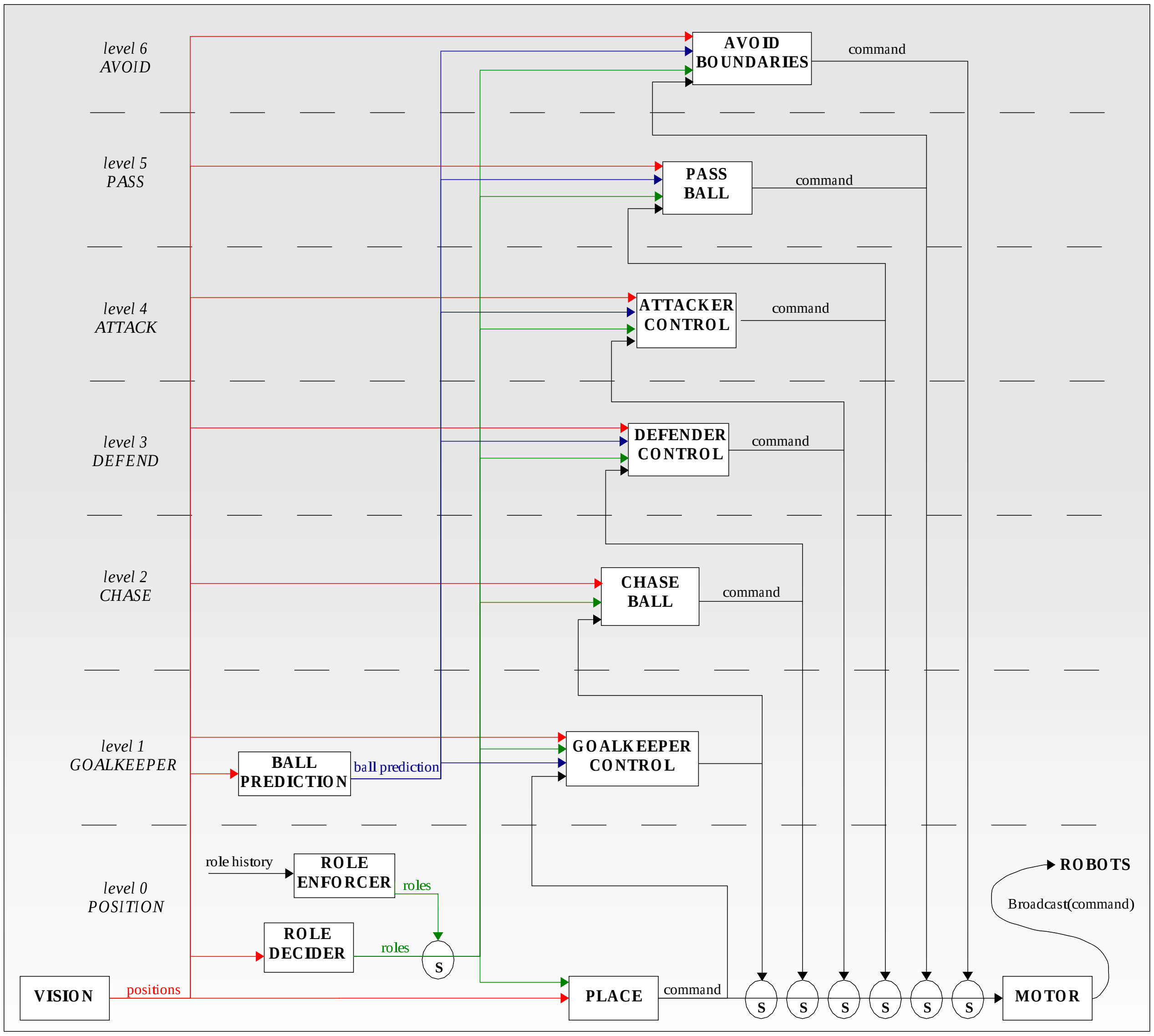

Fig. 2. The proposed architecture.

exclusive roles (either goalkeeper, defender, or attacker) to each of the robots.

The *command* wire carries the left and right wheel velocities for each of the three robots on the pitch. It is interesting to note that the architecture is finally only concerned with modifying (or not) the contents of *command*. Regardless of the amount of computations, predictions, or planning done within a given sampling interval, the strategic exercise is reduced to updating the contents of *command*, which is implemented as an 8-byte array. The first and last bytes are used for delimiting the RF message, while every other pair of bytes is used to store a left and a right wheel velocity to be broadcasted to the robots. This allows us to only modify the part of *command* we are interested in, leaving the other portions of the array unchanged.

The *role history* wire carries the ten previous role assignments for each of the robots in the team, while *Broadcast(command)* represents the function that takes *command* as a parameter, and is used to broadcast the RF signal to the robots on the pitch at the end of every sampling interval. For a thorough explanation on the subsumption theory, including inter-module communication and diagram notations, please refer to [1].

When the system is at the POSITION level of competence, we expect the robots to move to predetermined positions on the pitch.

When the system is at the GOALKEEPER level of competence, we expect that the robots assigned with the roles of defender and attacker will remain in the same position as they were when the system was at the first level of competence. The robot assigned with the goalkeeper role, however, will now display a complete goalkeeper behaviour, moving up and down in front of the home goal's mouth, attempting to stop the ball from being placed inside the goal. Also at this level, the system utilises for the first time a predicted position for the ball, to enable the goalkeeper to more efficiently track a fast moving ball.

When the the system is at the CHASE level of competence, we expect it to display a slightly more elaborate behaviour than the one demonstrated at the GOALKEEPER level of competence. The robot assigned with the goalkeeper role remains operating as the goalkeeper, while the robot assigned with the attacker role chases the ball around the pitch. At this level of competence, the system displays the emergent behaviour of moving the ball around the pitch, without any human intervention (i.e. the ball is constantly being pushed around by the attacker robot). The third robot (assigned with the role of defender) remains standing in its position, just as it was in the POSITION level of competence.

When at the DEFEND level of competence, the system's behaviour is improved by having not only an active goalkeeper and a *ball chaser*, but also by now having the robot assigned with the role of defender displaying a defending behaviour. This behaviour consists of vertically tracking the ball up and down the pitch, attempting to prevent it from approaching the defense line. The system is now manifesting a cooperation behaviour, by having some of its parts (the robot assigned with the goalkeeper role, and the robot assigned with the defender role) working towards the same overall goal of stopping the ball from being placed inside the home goal. Also at this level, we start to see the swapping of roles more often than before, due to the more dynamic environment.

When the system is at the ATTACK level of competence, it is for the first time able to pursue its main goal of placing the ball inside the opponent's goal, while not allowing the opponent to place the ball inside its own goal. This is achievable by having the DEFEND level of competence improved by enabling the robot assigned with the role of attacker to proactively kick the ball towards the opponent's goal, while avoiding hitting the ball towards its own goal.

When the system achieves the PASS level of competence, we expect it to improve on ATTACK, by enabling the attacker robot to kick the ball towards the defender robot whenever a certain predetermined scenario takes place. For instance, one of these scenarios occurs whenever the defender is in a better position to kick the ball towards the opponent's goal than the attacker is. This behaviour is crucial for performing the Mirosot benchmark problem that deals with passing the ball between players, and then shooting the ball towards the opponent's goal. However, this level is only relevant (or efficient) when we are recognising the position of the opponent robots on the pitch. Otherwise, our attacker may well end up passing the ball to the enemy instead.

The AVOID layer corresponds to the seventh level of competence of our system. The behaviour expressed by this layer has been previously referred to as our chief behaviour. This level improves on the previous one by enabling the robots to turn away from any of the pitch's boundaries, should the issuing of the current content of *command* (as modified by the lower level layers) cause any of the robots to collide with these boundaries. One of the reasons why we consider boundary avoidance to be our chief behaviour is the practical matter of having to maintain the physical integrity of our players. We cannot afford to have our robots slamming into the walls frequently as this may cause serious damage to them.

The architecture at this level of competence can be used to further exemplify the limitation for the application of a pure subsumption architecture to Robotic Soccer. This limitation is represented by having the necessity to elect a chief behaviour. It is not possible to simultaneously expand the architecture further than the AVOID level of competence and maintain AVOID as the chief behaviour without violating the rule of incremental development.

5. Results and Evaluation

In this section we present the results obtained by our architecture on the three Mirosot benchmark problems [7], along with some qualitative evaluation of our implementation.

The hardest problem (problem C) was attempted by having the PASS layer suppressing the output of the ATTACK layer by replacing ATTACK's output with a new *command* (modified with a pair of new right and left wheel velocities for the attacker). This new pair of velocities was used to direct the attacker robot (the passing robot) to kick the ball towards the x and y coordinates of the receiving robot. Table 1 shows a summary of the results obtained by our subsumption architecture while performing the Mirosot benchmark problems.

Table 1. Overall results for benchmark problems.

Benchmark problem	Maximum score	Score achieved	Success rate (%)
A	10	10	100.00
B	20	14	70.00
C	20	11	55.00
Overall	50	35	70.00

As expected, the architecture displayed the highest level of success (100.00 %) for the easiest of the problems (problem A). The performance deteriorated, however, as the problems got harder (70.00 % for problem B), and involved some explicit cooperation between robots (55.00 % for problem C). The concept of incremental development had, in our case, a positive impact in the development of a Robotic Soccer architecture. We found, however, that the benefits provided by the incremental development concept are weakened by the presence of two characteristics in the Robotic Soccer domain. First, there is the need to elect a chief behaviour for the team. Whenever the chief behaviour is identified, its layer must always remain on top, as previously discussed in Section 3.3.3. Second, Robotic Soccer is a domain where existing behaviours must be revisited many times, if the overall performance of the team is to be improved. Addressing these characteristics either by inserting new layers into the existing architecture, or by changing existing behaviours would clearly contravene the incremental development rule.

The team cooperation achieved through the dynamic role assignment to the players is a positive outcome of our implementation. Team cooperation is also enhanced by the PASS layer, which enables the team to capitalise on certain passing opportunities that may arise during a match.

6. Conclusion and Future Research

In this paper we have evaluated the feasibility of implementing a pure subsumption architecture to control our Robotic Soccer team. We began by presenting an overview of our current system in Section 2. In Section 3 we presented and discussed a number of issues which influenced the way the final architecture was designed. In Section 4 we introduced our seven-level subsumption architecture. Finally, in Section 5 the results obtained from empirically testing this architecture on a set of existing Robotic Soccer benchmark problems were shown, along with a qualitative evaluation of our implementation. In this section we present our conclusive remarks regarding this study, along with some areas that are open for future research.

The major finding of this paper is that strict observance of the incremental development concept within the Robotic Soccer domain is possible, although it leads to unnecessary and avoidable complexity. Our major contribution is the proposal of a new design methodology that improves on the subsumption's incremental development concept and eliminates its inheriting complexities by allowing for: (a) The election of a chief behaviour as the topmost layer of the architecture; and (b) the modification of existing control modules within a layer without having to concurrently modify other parts of the architecture.

We believe it is only feasible to continue the work on the architecture presented here if the rule of incremental development is paid no regard to, and our new methodology is adopted.

As future research issues, we believe it is worthwhile to evaluate the feasibility of implementing a behaviour selection approach to control our team for two reasons. First, the approach has been proved successful by its implementation to the current world champion team of FIRA's Mirosot league. Second, the use of a behaviour selector accommodates the need for a chief behaviour, which cannot be done with a pure subsumption architecture. We would also like to formally define the subsumption approach to Robotic Soccer presented in this paper, if this development paradigm is adopted. This could be done through the development of a framework for automating the design and deployment of new behaviour-producing layers and low level control modules.

References

[1] R. A. Brooks, "A robust layered control system for a mobile robot". *Technical report A.I., memo 864*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, September 1985.

[2] "Mirosot rules", `http://fira.cqu.edu.au/rwc2000/mrules.htm`, 2001.

[3] J. Thornton, "Robot soccer in 21 days". Unpublished, School of Information Technology, Griffith University Gold Coast, Southport, QLD, Australia, December 2000.

[4] L. E. Parker, "Adaptive heterogeneous multi-robot teams". *Neurocomputing, special issue of NEURAP '98: Neural Networks and Their Applications*, volume 28, 1999.

[5] R. C. Arkin, "Motor schema-based mobile robot navigation". *International Journal of Robotics Research*, 1987.

[6] D. M. Shin, "Behavior selection strategy for soccer robots". In *Journal of Harbin Institute of Technology*, Vol 8, No. 3, 2001.

[7] J. Johnson, J. L. de la Rosa, and J. H. Kim, "Benchmark tests of robot soccer ball control skills". `http://fira.cqu.edu.au/rwc2000/pages/firabenc.htm`, 2001.