336403

BIBLIOTECA

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MEXICO

07 FEB 2014

Campus Estado de México
Biblioteca

Donado Por Dr. Jaime Mora Vargas
Fecha 23 JUL 2013

230-22

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

DEFINICIÓN DE UNA ARQUITECTURA PARA LA GENERACIÓN Y SELECCIÓN DE ESTRATEGIAS EN EQUIPOS DE FÚTBOL ROBÓTICO

TESIS PARA OPTAR EL GRADO DE
MAESTRA EN CIENCIAS DE LA COMPUTACIÓN
POR

MYRIAM ZORAYA ARIAS RUIZ

Asesor:	DR. JORGE ADOLFO RAMÍREZ URESTI	
Comité de tesis:	DR. ISAAC RUDOMÍN GOLDBERG	
	DR. MIGUEL GONZÁLEZ MENDOZA	
	DR. LEONARDO GARRIDO LUNA	
Jurado:	DR. ISAAC RUDOMÍN GOLDBERG	Presidente
	DR. MIGUEL GONZÁLEZ MENDOZA	Secretario
	DR. LEONARDO GARRIDO LUNA	Vocal
	DR. JORGE ADOLFO RAMÍREZ URESTI	Vocal

Atizapán de Zaragoza, Edo. Mex., Mexico, Junio 2008

Esta tesis corresponde a los estudios realizados con una beca otorgada por la Secretaría de Relaciones Exteriores del Gobierno de México.

RESUMEN

En el presente trabajo se hace uso de técnicas de sistemas multiagentes para la comunicación, la interacción y más específicamente coordinación entre agentes, lo que permite la definición de una arquitectura de estrategias de juego, para ser aplicada en equipos de fútbol robótico. Esta arquitectura seleccionará planes de juego a medida que las condiciones del ambiente sean modificadas por los jugadores, tanto compañeros como adversarios.

El entorno de ejecución se realiza con base en robots tipo *Sony AIBO ERS-7* que son utilizados en la liga de cuadrúpedos de *RoboCup*, proyecto internacional que promueve la investigación en Inteligencia Artificial (AI) y Robótica, con el fin de desarrollar tecnologías y apoyar la resolución de diversos problemas.

Esta investigación es una continuación del trabajo realizado en el proyecto de investigación de robots cuadrúpedos, del Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México ITESM-CEM; relacionado con la aplicación de comportamientos a través del lenguaje *XML (Extensive Markup Language)* por medio de un sistema de eventos y acciones, en un ambiente de simulación.

La propuesta de arquitectura de estrategias de juego, TABA (Team Agent Behavior Architecture), se define de tal forma que pueda ser empleada en diversos ambientes que apliquen coordinación entre agentes, para lograr un objetivo en común; esto con base en la combinación de comportamientos de planeación estratégica (deliberativo) y comportamientos de acción inmediata (reactivo). Así mismo el comportamiento colaborador es definido mediante el uso de maquinas de estado finito aplicado a sistemas multiagentes.

Se hace uso del Razonamiento Basado en Casos como metodología para la resolución de problemas de toma de decisión, la cual responde a una coalición de tecnologías que en conjunto proveen las bases para el concepto, diseño, y utilización de sistemas inteligentes, y se presenta en este trabajo de investigación, como alternativa en la resolución de problemas de coordinación en sistemas multiagentes aplicado al fútbol robótico.

Dentro de las principales contribuciones del trabajo de investigación realizado se puede mencionar: la definición de una arquitectura basada en equipos de agentes, que trabaja de forma flexible, y que permite descomponer el espacio de tareas o acciones en un número finito de roles o comportamientos, que pueden ser intercambiados a medida que el juego o el estado del medio varíe. Así mismo se proporciona una simbología para la definición de estrategias denominada SSDS.

La arquitectura se divide en cuatro etapas: la selección del líder, la selección de la estrategia, asignación de roles y ejecución/terminación de la estrategia.

AGRADECIMIENTOS

A nuestro Señor por su infinito amor y eterna misericordia.

A mis padres Nemer Arias y Dolores Ruiz por su amor, comprensión y apoyo incondicional.

A mi asesor Jorge Ramirez por su tiempo y paciencia, sus útiles consejos y excelentes ideas.

A mis amigos por sus palabras de animo y cariño durante todos estos meses.

A todos los integrantes del proyecto de investigación de robótica móvil en cuadrúpedos, humanoides y rescate.

A todos los profesores que forman parte del claustro de la Maestría en Ciencias Computacionales por permitir adentrarme en el área de inteligencia artificial y el mejoramiento continuo sobre los valores profesionales, éticos y sociales.

Al Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México por fomentar la investigación y su apoyo integral.

Al gobierno de los Estados Unidos Mexicanos, por medio de la Secretaria de Relaciones Exteriores por brindarme la oportunidad de realizar estudios en este país.

ÍNDICE GENERAL

1. Introducción **11**
1.1. Descripción del problema 12
1.2. Justificación 15
1.3. Motivación 15
1.4. Hipótesis 17
1.5. Objetivos 17
1.6. Aportaciones y Beneficios 18
1.7. Alcances 18
1.8. Metodología 19
1.9. Organización de la tesis 20

2. Estado del Arte **21**
2.1. ¿Qué es una estrategia? 21
2.2. Agente y Sistemas Multiagentes 22
2.2.1. Agentes Inteligentes 22
2.2.2. Sistemas Multiagentes 25
2.2.3. Coordinación: cooperación y negociación en SMA 26
2.3. Fútbol Robótico 30
2.3.1. *FIRA* 30
2.3.2. *RoboCup* 31
2.4. SMA aplicado al Fútbol Robótico 33
2.4.1. Razonamiento basado en casos 33
2.4.2. Máquinas de aprendizaje - Selección de planes 37
2.4.3. Lógica difusa 38
2.5. Metodología para la toma de decisiones 41
2.6. Resumen del capítulo 43

3. Herramientas utilizadas por el equipo TecRams **44**
3.1. Herramientas utilizadas en TecRams 45
3.1.1. GT2004/XABSL 45
3.1.2. XML Behaviour Control 48
3.1.3. ABG 50
3.1.4. BADRL 53
3.2. Resumen del capítulo 57

4. Team Agent Behavior Architecture - TABA **59**
4.1. Introducción a la Arquitectura 60
4.2. Entorno y reglas de fútbol robótico 61
4.3. Entorno de Simulación 62
4.3.1. XML Behaviour Control y TABA 62
4.3.2. División en cuadrantes 63
4.4. Equipo y Jugadores 63

4.4.1. Estados del Equipo 64
4.4.2. Estados de los jugadores 65
4.5. Estrategia y Base de Estrategias 67
4.5.1. Representación de una Estrategia 68
4.5.2. Simbología - Estrategia Extendida 68
4.5.3. Indexación - Estrategia Simplificada 70
4.6. Definición de un caso para equipo de agentes 71
4.7. Arquitectura de estrategias 73
4.7.1. Selección del agente líder 74
4.7.2. Selección de la estrategia 75
4.7.3. Asignación de roles 79
4.7.4. Ejecución y terminación de la estrategia 80
4.7.5. Ejemplo 80
4.8. Resumen del capítulo 86

5. Experiencias y Resultados **87**
5.1. Experiencias 88
5.1.1. Estrategias de Pizarra 88
5.1.2. Simbología de términos 89
5.1.3. Entorno de Juego 90
5.1.4. Arquitectura de comportamiento en equipo 90
5.1.5. Practica vs Juego 94
5.2. Resultados 95
5.2.1. Resultados Generales de TABA 95
5.2.2. Ejecución de Estrategias sin Oponentes 97
5.2.3. Ejecución de Estrategias con Oponentes 97
5.3. Resumen del capítulo 100

6. Conclusiones y Trabajo Futuro **102**
6.1. Conclusiones 103
6.1.1. Ventajas de TABA 103
6.1.2. Limitaciones de TABA 104
6.2. Trabajo futuro 104
6.2.1. Actualización y mejora de la aplicación 104
6.2.2. Generación de estrategias 106
6.2.3. RBC y otras metodologías 107
6.2.4. Entornos de aplicación 108

Bibliografía **111**

A. Lista de Acrónimos **115**

B. Métodos Principales **116**
B.1. Triggers Game States 116
B.1.1. GoalScored 116
B.1.2. BallEndLine 117
B.1.3. Estados de *goalKeeper* 118
B.1.4. Estados de *fieldPlayer* 119

C. Manual de Usuario **120**
C.1. Requerimientos del Sistema 120
C.2. Archivos de tipo XML 121
C.3. Edición de Archivos de tipo XML 121
C.3.1. Herramienta Editix 122
C.3.2. Instalación y Ejecución de Editix 122
C.3.3. Edición de archivos XML 123
C.4. Archivo: Definición de ambiente 124
C.5. Archivo: Índice de Estrategias 126
C.6. Archivo: Definición de Estrategias 127
C.6.1. Definición de estrategia 127
C.6.2. Definición de estados 128
C.6.3. Definición de transiciones 128
C.6.4. Definición de acciones 129
C.6.5. Definición de eventos 129
C.7. Ejecución de la aplicación 131

D. Manual de Programación **134**
D.1. Requerimientos del Sistema 134
D.2. Edición de clases Java 135
D.2.1. Herramienta Eclipse 135
D.2.2. Instalación y Ejecución de Eclipse 135
D.3. XML Behaviour Control y TABA 136
D.3.1. Instalación 136
D.3.2. Modificación 137
D.3.3. Ejecución 139
D.3.4. Reportes 139

ÍNDICE DE FIGURAS

1.1. Imagen del *passing challenge* 14

2.1. Comportamiento de un agente basado en estados 25
2.2. Coordinación entre agentes 26
2.3. Robots en Categoría Estándar de RoboCup 33
2.4. Ciclo del RBC 35
2.5. Ejemplo de la definición de un caso en *RoboCat* 36
2.6. Ejemplo de la recuperación de un caso en *Humboldt* 37
2.7. Categoría *Small Size* CMDragons vs ToinAlbatross; aplicación de *playbook* 39
2.8. Aplicación de *fuzzierX* 41

3.1. Tareas identificadas por *GT2004* para el fútbol robótico 45
3.2. Definición de símbolos en *C++* `walk` 46
3.3. Implementación comportamiento básico en *XABSL* `go-to-ball` 47
3.4. Definición de comportamiento en *XABSL* `golie-position` 47
3.5. Entorno principal del simulador *XML Behaviour Control* 48
3.6. Manejador de comportamiento jugador:*atacker1* estado activo: *waitPassReply* . . . 49
3.7. Clase para manejar la comunicación con el *Sony AIBO* 50
3.8. Definición de un comportamiento en *XML* `enviarPase` 50
3.9. Definición del entorno de juego y los jugadores 51
3.10. Proceso de generación de comportamientos haciendo uso de computo evolutivo . . 52
3.11. Par de cromosoma en una generación específica, cada conjunto de comportamiento esta compuesto por un comportamiento de tres estados 52
3.12. Nueva Solución en el *XML Behaviour Control* 54
3.13. Procesos General de *BADRL* 54
3.14. Archivo de configuración `conf.xml` 55
3.15. Estado que ejecuta una acción atómica y transición a un estado principal 56
3.16. Definición de reglas en `rules.xml` para *conducción de la pelota* 56
3.17. Criterios de evaluación en `rules.xml` 56

4.1. Dimensiones de la cancha 61
4.2. Entorno general del juego 62
4.3. Entorno general del simulador con la división en cuadrantes 64
4.4. Representación simbólica de la cancha dividida en cuadrantes 67
4.5. Representación simbólica de una estrategia. Los números laterales representan los cuadrantes 69
4.6. Parte del archivo de definición de estrategias 71
4.7. Parte del archivo indexación de estrategias 72
4.8. Arquitectura de estrategias de juego - TABA 74
4.9. Selección general de una estrategia 76
4.10. Configuración del Cálculo de similitud 78
4.11. Representación simbólica de la estrategia KO3 80
4.12. Ejecución estrategia K03 parte I en XML Simulator Control 83

4.13. Ejecución estrategia K03 parte II en XML Simulator Control 84
4.14. Terminación estrategia K03 en XML Simulator Control 85
4.15. Selección de líder y definición de caso 85
4.16. Cálculo de similitud y selección de estrategia 86

5.1. Representación simbólica de una estrategia - primeras versiones 89
5.2. Representación simbólica de una estrategia - versión final 90
5.3. Cancha de juego 91
5.4. Resultado competencia Red de Contratantes 100
5.5. Resultado competencia TABA y Red de Contratantes 100

6.1. Reglas cuadrúpedos 2008 106
6.2. Robot de Rescate y Robots de navegación autónoma 109
6.3. Robot Humanoide 110

B.1. Ejemplo de verificación de goles 116
B.2. Ejemplo pelota fuera 117
B.3. Ejemplo estados del *goalKeeper* 118
B.4. Ejemplo estados del *fieldPlayer* 119

C.1. Herramienta Editix 122
C.2. URL para descargar Editix 123
C.3. Archivo de Definición de Ambiente 125
C.4. Archivo de Definición de Índice de Estrategias 127
C.5. Archivo de Definición de Estrategia 128
C.6. Archivo de Definición de Estados 128
C.7. Archivo de Definición de Transiciones 129
C.8. Archivo de Definición de Acciones 130
C.9. Archivo de Definición de Eventos 131

D.1. URL para descargar Eclipse 136
D.2. Creación del proyecto robocup 137
D.3. Proyecto RoboCup en Eclipse 138
D.4. Cálculo de distancia entre dos casos 139
D.5. Ejecución de la herramienta de simulación 140
D.6. Ejemplo del archivo 'CBR Report File' 140

ÍNDICE DE CUADROS

2.1. Perspectiva de taxonomías de agentes 29
2.2. Categorías de *Robo Soccer* en FIRA 31
2.3. Categorías de *RoboCup* 32
2.4. Comparación de tecnologías 42

3.1. Probabilidades de Mutación de un Cromosoma 53

4.1. Representación textual de una estrategia K02 69
4.2. Atributos y valores de un caso 72
4.3. Definición y condiciones de la estrategia de KICKOFF K03 81
4.4. Ejecución y terminación de la estrategia KO3 82

5.1. Competencia entre Red de Contratantes 98
5.2. Competencia entre TABA (Azul) y Red de Contratantes(Rojo) 99

C.1. Acciones Complejas 130
C.2. Eventos del Módulo de Visión 132
C.3. Acciones del Módulo de Locomoción 133
C.4. Eventos del Módulo de Comunicación 133

1. INTRODUCCIÓN

"Mucha sofisticación ha sido desarrollada sobre la utilización de máquinas para objetivos complejos, y no tiene sentido no usarlo; si no se usa, es como si piensas en una buena pregunta y no la haces"
Noam Chomsky

Hoy en día, las organizaciones modernas, sin importar el tipo de misión y visión que persigan, llevan a cabo una carrera contra el tiempo para alcanzar y mantener el éxito, tomando en cuenta que se requieren talentos imposibles de encontrar en un solo individuo, sobre todo en organizaciones con estructura lineal y con menos niveles jerárquicos, en las cuales se requiere una interacción entre individuos, de forma cooperativa y no individualista.

Lo anterior se aplica en gran manera a todos los aspectos de la sociedad, la familia, el trabajo, los estudios, los deportes. Es en este último, los deportes, en el cual, la importancia de la planeación estratégica, es vital para el logro de objetivos, principalmente cuando se requiere corregir o mejorar el desempeño a corto o largo plazo, con el fin máximo de ganar.

Las ciencias computacionales, en especial el estudio de Inteligencia Artificial, la Robótica y Sistemas Multiagentes, han despertado gran interés en las últimas décadas por su importancia en la representación o modelado de las interacciones humanas con fines académicos, de negocio, militares, de salud, entre otros. Esto se debe a la capacidad de estas ciencias de manipular altos y complejos volúmenes de información o realizar procesamientos específicos en dependencia de los dominios de aplicación.

El término agente, en un sistema de computo, posee características como autonomía, habilidad social, es reactivo y está situado en un ambiente donde percibe, razona y actúa, es día a día, más

cercano a un agente humano y su interrelación cobra una particular importancia para la resolución de problemas y toma de decisiones, sobre todo en ambientes dinámicos y bajo incertidumbre. Para efectos del presente documento se hará uso del término agente para referirnos a un robot tanto en un ambiente de simulación (software) como en un entorno real (robot).

Gracias a diversas iniciativas de organismos internacionales que persiguen el fin de promover la investigación, creación e innovación de nuevas y mejores tecnologías, se han analizado distintos enfoques sobre el estudio de los agentes, así como sistemas multiagentes y sus principales aspectos como: la comunicación, la interacción, la planeación y el aprendizaje.

En el contexto de fútbol robótico, un ambiente complejo, dinámico y de alto grado de incertidumbre, los equipos de robots deben enfrentar diversos retos, como por ejemplo localización los jugadores, locomoción eficiente y toma de decisiones; dentro de esta última, la coordinación efectiva entre los jugadores, para llevar a cabo una meta en común bajo una configuración altamente competitiva. Entre las técnicas de estrategia cooperativa para los sistemas de fútbol robótico se destaca el uso de métodos estadísticos, algoritmos de aprendizaje, razonamiento basado en casos, entre otros.

El razonamiento basado en casos, una coalición de tecnologías que en conjunto proveen las bases para el concepto, diseño, y utilización de sistemas inteligentes, se presenta, en este trabajo de investigación, como alternativa en la resolución de problemas de coordinación en sistemas multiagentes aplicado al fútbol robótico. En [Pal and Shiu, 2004] se puntualiza que diferentes técnicas pueden ser usadas y aplicadas en alguna de las fases del ciclo de vida del razonamiento basado en casos. Entre estas se pueden mencionar: técnicas del vecino mas cercano, algoritmos de inducción, lógica difusa, redes neuronales, computo probabilístico, y máquinas de aprendizaje.

El presente trabajo es un propuesta que tendrá como fin diseñar, implementar y probar una arquitectura en etapas para la definición, selección y aplicación de estrategias en el ambiente de fútbol robótico, que maximice las acciones de cooperación y coordinación entre los agentes para lograr un objetivo común y con éxito.

1.1. DESCRIPCIÓN DEL PROBLEMA

Los sistemas multiagentes, aplicados a dominios complejos y en tiempo real, requieren de agentes que actúen de forma estratégica tanto de forma individual como en equipo. Un ejemplo de este dominio complejo, es el ambiente del fútbol robótico, en el cual, es más inherente la necesidad de emular el comportamiento humano, mediante la aplicación de estrategias de juego, que les permita obtener mejores resultados.

A pesar que los robots son diseñados con características que le permiten analizar el entorno, mediante dispositivos de entrada como cámaras y sensores, aún sigue siendo una limitante, la capa-

cidad de abstraer completemente el ambiente, y construir planes de acción efectivos, que superen las adversidades del medio, dinámico y bajo incertidumbre.

Robocup es una iniciativa internacional que promueve la investigación y educación sobre inteligencia artificial, hace uso del deporte de fútbol como base en la representación de una serie desafíos, para alcanzar el objetivo de modelar las interacciones humanas[RoboCup, 2008]. El proyecto se divide en categorías, entre las que se destaca la liga de robots cuadrúpedos, cuyo principal objetivo es hacer uso de robots de tipo *SONY AIBO ERS-7* para llevar a cabo una competencia de fútbol, sobre un terreno de juego con reglas definidas[AiboSoccer, 2007].

La liga de robots cuadrúpedos, en voz inglesa *Four-Legged League*, posee especial desafío, debido a que el reto se centra en el desarrollo efectivo de software, a diferencia de otras categorías, no se puede realizar modificaciones al hardware de los robots utilizados, de esta forma, la información que se obtiene proviene directamente de la recepción de datos, de los dispositivos de entrada de los robots, fija e inalterable, lo que convierte una limitante dado, que no se posee una percepción global del entorno, como ocurre en la liga de robots pequeños.

La liga de robots pequeños, en voz inglesa *Small-Size League*, dispone de una cámara central que capta todo el campo de juego y una unidad de procesamiento externa que permite realizar operaciones y computo centralizado y, posteriormente, enviar ésta información a los robots durante el juego, tal como se refiere en [Bowling et al., 2004].

Dado lo anterior, la aplicación de un curso de acción, en la liga de cuadrúpedos, depende únicamente de la percepción limitada de los robots a un entorno o visión local y propensa a ruidos y errores, lo cual implica no obtener el máximo rendimiento de los jugadores como equipo.

Actualmente el Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México, forma parte de la red de instituciones de educación superior y de investigación, que participan en la competencia y congresos promovidos por *RoboCup*, por medio de la Cátedra Agentes Virtuales y Robóticos en Ambientes de Realidad Dual, de la Rectoría Zona Centro, y el proyecto de robot cuadrúpedos.

Uno de los principales problemas que se tiene en el proyecto de investigación, es que no se cuenta con una herramienta que permita modelar el comportamiento de los robots en estrategias de cooperación y coordinación de forma eficiente. A pesar que trabajos previos de investigación han propuesto soluciones de interacción, como lo es el uso de la tecnología de red de contratantes en voz inglesa *Contract Net*, en un entorno simulado, es necesario mejorar y aplicar otras técnicas para una coordinación exitosa tanto en el entorno simulado como en los robots reales.

Esto limita en gran medida la emulación de un juego real que parte de las premisas básicas de definición de estrategias tanto ofensivas como defensivas, así como de otros tipos, como de eva-

sión, bloqueo, pases, los cuales requieren de la participación de dos o mas jugadores.

Otro aspecto que implica un desafío, tanto en un entorno simulado como en el real, es la asignación de roles, en los trabajos previos de investigación, se define desde que los agentes son creados y no varían a lo largo del juego, por ejemplo en el entorno de simulación se definen portero, atacante, defensa y soporte, cada uno con su respectiva especificación de comportamientos en una máquina de estados que evoluciona a medida de ciertos cambios en el entorno y a su rol dentro del juego, como lo es, manejo del balón, posición con respecto a la portería, evasión de obstáculos entre otros, esto implica que el comportamiento se aplica de forma individual y no en grupo, todos los agentes trabajan para lograr una meta, pero no colaboran entre sí para alcanzarla, lo hacen por que es lo que tienen que hacer.

Lo anterior impulsa la motivación de diseñar mecanismos alternos para una comunicación efectiva, precisa y de menor carga computacional que permita a determinados agentes tomar una decisión acerca del proceder del equipo en el entorno de juego y seleccionar estrategias para determinados eventos con aplicación de roles dinámicos.

En la Figura 1.1 se muestra un ejemplo de comportamiento que requiere de coordinación entre los agentes, es el llamado reto de pases, en voz inglesa *passing challenge* de la categoría de robots cuadrúpedos de *RoboCup*, su objetivo es, en forma general, alcanzar el mayor número de pases efectivos entre tres jugadores en un tiempo límite; un pase efectivo se refiere a que el jugador A envía un pase al jugador B y este al jugador C, de forma continua durante el límite de tiempo.

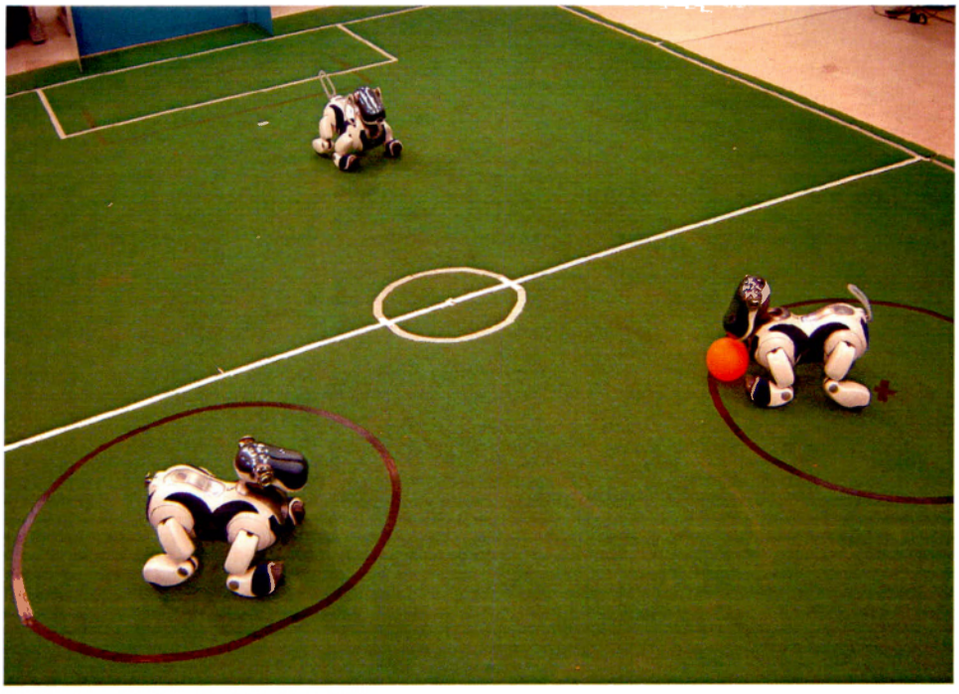


Figura 1.1: Imagen del *passing challenge*

1.2. JUSTIFICACIÓN

La comprensión, desarrollo y uso de los sistemas multiagentes, como respuesta a la definición y resolución de problemas variados, ha sido una constante en las últimas decadas. La iniciativa de organismos internacionales, dedicados a la investigación y desarrollo, las entidades educativas así como la estrategia de acción de gobiernos y compañías privadas, propician un ambiente para la invención y creación, proyectos que tenían un objetivo específico inicial han sido ampliados y aplicados a otros dominios, ejemplo de esto es la historia que ha seguido *RoboCup*, que nació de la propuesta de investigadores independientes en Canadá y Tokio, de usar el juego de fútbol para promover la ciencia y tecnología, por los retos que este juego implica [RoboCup, 2008]. Actualmente se han desarrollado otras ligas con diferentes objetivos y reglas, las cuales se abordará en los próximos capítulos.

Es por esto que este proyecto, no se limita a dar respuesta a una problemática definida, si no mas bien, continuar un trabajo de investigación, aportar en la búsqueda continua de conocimiento y aplicarlos en proyectos de gran relevancia, así mismo instar a otros a participar y aportar ideas con el fin de aplicar el extenso campo de la Inteligencia Artificial, en la resolución de problemas diversos, ya sea de orden comercial, educativo, de salud en general en beneficio de la humanidad.

La definición de una arquitectura de estrategias especificada en etapas, permite dividir el problema en subproblemas, lo cual es de utilidad para resolver, de forma integral y escalable, las diferentes adversidades que se presentan en un entorno de juego, con dominios complejos, limitados recursos y en un ambiente competitivo.

En respuesta a la problemática expuesta en la sección anterior, el presente proyecto propone que la asignación de roles sea de forma dinámica con base en los cambios del entorno, a pesar que al momento inicial del juego, se les asignará un rol por defecto a cada uno de los agentes, al momento de seleccionar una estrategia los roles cambiarán a medida de las necesidades ofensivas o defensivas a excepción del portero, que no cambiará su rol a lo largo del juego, pero su comportamiento se ajustará dinámicamente a los cambios del mismo.

1.3. MOTIVACIÓN

Los objetivos que persigue la Inteligencia Artificial (IA) han sido variados y algunos definidos con base en la perspectiva de los investigadores, pero estos objetivos se pueden concentrar en uno solo: entender, explorar, modelar y estimular la inteligencia humana tal como se refiere en [Raymond, 2006]. Así mismo, la automatización de actividades relacionadas al pensamiento humano, como la toma de decisiones, la resolución de problemas, el aprendizaje artificial, presentado por [Bellman, 1978] y más especificamente el desarrollo de agentes inteligentes, que interactúan dentro de un ambiente determinado y se relacionan con otros agentes.

Un agente humano hace uso de sus facultades mentales, motrices y sentidos para comprender el ambiente y actuar sobre éste, en cambio un agente robótico percibe su entorno por medio de sensores y lo modifica por medio de actuadores tal como se refiere en [Russell and Norvig, 2003], siendo más efectivo que los humanos en algunas tareas específicas, pero limitados en acciones de corte general.

Se debe destacar que entre las características que posee un agente, es el hecho de que en la mayoría de los sistemas computacionales donde se aplique un agente, este actúa en conjunto con otros agentes, con quienes comparte objetivos y metas, ya sea comunes o contrarias.

Un ejemplo de agentes que actúan en equipo, es aplicado al caso de fútbol robótico, implementado en las competencias internacionales de *FIRA*[1] y *RoboCup*[2]. De ahí nació la necesidad de simular el comportamiento humano, al emplear técnicas de un equipo de fútbol humano a uno ejecutado por sistemas multiagentes.

Actualmente el proyecto de investigación de robots cuadrúpedos del Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México, ha desarrollado un equipo, *TecRams* [TecRams, 2008], que hace uso de los robots *SONY AIBO* modelo *ERS-7* [AIBO, 2007] para competir en la categoría de cuadrúpedos de *RoboCup*, a pesar que se ha obtenido buenos resultados a nivel latinoamericano, es necesario mejorar el desempeño a nivel internacional.

Se debe tomar en cuenta que en comparación con otros equipos de mayor trayectoria, el *TecRams* ha logrado en poco tiempo, desde su fundación en 2002, organizar un equipo capaz de competir de forma ágil y eficaz bajo un enfoque de multiagentes, con roles individuales y comportamientos completamente reactivos. El presente trabajo propone cambiar el enfoque de comportamientos individuales a uno más cooperativo por medio de la aplicación de una arquitectura de estrategias y selección de planes de juego, que permita la asignación de roles de forma dinámica y autónoma haciendo uso de razonamiento basado en casos.

Además de mejorar el desempeño en competencia del equipo *TecRams*, la aplicación de conceptos de inteligencia artificial, robótica y sistemas multiagentes es de vital importancia para resolución de problemas de la vida real, y no se limitan a un aspecto particular como lo es el fútbol robótico, por el contrario, este trabajo presenta un propuesta escalable y puede ser aplicada a otros entornos o ambientes como el caso de los robots de rescate, diseñados para ser usados en lugares donde ha ocurrido algún tipo de catástrofe, como apoyo para reconocimiento del lugar y rescate de víctimas, acciones que requieren un trabajo coordinado y colaborador.

Se debe considerar de forma significativa que parte de la motivación es aportar con el sueño de

[1] *Federation of International Robot Soccer Association* (Federación Internacional de la Asociación de Fútbol Robótico), `http://www.fira.net`

[2] *Robot World Cup Soccer Games and Conferences* (Juegos y Conferencias de la Copa Mundial de Fútbol Robótico), `http://www.robocup.org`

lograr una interacción efectiva entre individuos y agentes, por el bien de la humanidad, asemejando los sistemas multiagentes a la forma en como opera un sistema humano y social.

1.4. HIPÓTESIS

El diseñar e implementar una arquitectura para la selección y ejecución de estrategias de juego, aplicada en fútbol robótico, que proveerá capacidad a los agentes de trabajar en forma colaborativa y coordinada, para llevar a cabo acciones ofensivas y defensivas que maximicen su desempeño individual y en equipo.

1.5. OBJETIVOS

El objetivo general de la presente propuesta de investigación es, diseñar, implementar y validar, la utilidad de aplicar una arquitectura para la ejecución de estrategias de juego, en equipos de fútbol robótico, que sea fácil de comprender y aplicar, y que pueda ser extendida a otros entornos de sistemas multiagentes, al considerar los principales retos del entorno dinámico y de incertidumbre que implica el ambiente de juego.

Los objetivos específicos se detallan a continuación:

a) Investigar y comparar las diferentes tecnologías en el campo de cooperación y estrategia en sistemas multiagentes, de forma genérica y aplicada a un entorno de juego. A su vez, analizar comportamientos del fútbol robótico y humano con el fin de descubrir patrones que permitan formular estrategias de juego potenciales y de esta forma clasificarlas por su uso y aplicación en el entorno de juego de fútbol robótico.

b) Definir y fundamentar la selección de estrategias con base en el tipo, parametros de localización y número de agentes.

c) Aplicar la metodología de razonamiento basado en casos para la selección de cursos de acción.

d) Proponer la arquitectura en capas para la aplicación de estrategias de juego de fácil comprensión e implementación.

e) Probar la arquitectura en un entorno de prueba sin oponentes y posteriormente, con adversarios.

f) Comparar el comportamiento de un equipo con arquitectura de estrategia *versus* un equipo sin arquitectura de estrategias.

1.6. APORTACIONES Y BENEFICIOS

A pesar de que el trabajo sobre cooperación y coordinación en entorno de sistemas multiagentes, ha sido reconocido en muchos trabajos de investigación, aún no se han establecido lineamientos firmes y definitivos sobre este campo; es un hecho que el trabajo de colaboración entre agentes es un reto que enfrenta muchas variantes y diversos aspectos que considerar, un ejemplo de esto es la competencia anual de *RoboCup* en el desafío de *pase entre tres robots*, mencionado anteriormente, el reto es propuesto de forma constante cada año con algunas variantes, a pesar que han habido actuaciones aceptables en el mismo, aún no se ha logrado completar eficientemente, nuevamente la sincronía es un factor de gran relevancia en conjunto con limitantes propias del diseño del robot.

En esta propuesta de investigación se consideran las siguientes aportaciones y beneficios:

a) Desarrollar una propuesta integral de arquitectura para la definición, selección y aplicación de estrategias de juego para fútbol robótico que sea fácil de comprender y de uso intuitivo, capaz de brindar flexibilidad a mejoras y expansibilidad a otros ambientes.

b) Definir y aplicar una simbología para la descripción de estrategias aplicados en el fútbol robótico.

c) Dar continuidad a trabajos previos de investigación, llevados a cabo en el Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México, relacionados a la administración de comportamientos de agentes, mediante máquinas de estado jerárquica, simulador *XML Behaviour Control.*

d) Mejorar el módulo de comunicación del *XML Behaviour Control* con el fin de proveerle nuevas performativas de comunicación que coadyuven en el proceso de aplicación de estrategias.

e) Incorporar la facilidad de separar comportamientos complejos en acciones atómicas, que puedan ser utilizadas de forma recurrente y de fácil comprensión.

f) Aplicar metodología de "razonamiento basado en casos", en voz inglesa CBR (*Case-Based Reasoning*) por sus siglas en inglés, para la selección de estrategias de acuerdo al estado del juego.

1.7. ALCANCES

Los alcances de este proyecto se centran en la definición de una arquitectura de estrategias con orientación en el entorno de fútbol robótico que pueda ser implementada en el simulador *XML Behaviour Control* y que permita mejorar el desempeño de los jugadores robótico mediante un trabajo colaborador y coordinado.

Es importante considerar los siguientes aspectos para establecer el alcance del trabajo de investigación:

a) Las estrategias a implementar son de tipo pizarra con características de secuenciador, incremento y oportunista. Secuenciador implica que una se ejecutará posteriormente a otra; incremento pueden realizarse modificaciones a la estrategia previo análisis y no en estado de ejecución; oportunista se activarán mediante los cambios de estado del ambiente de juego.

b) Se hará uso de una base de datos de estrategias previamente definidas que se probarán en el ambiente simulado, para proveer al programador la capacidad de rediseñar la estrategia.

c) La selección de estrategias, en tiempo de ejecución, se realizará mediante cambios de estado o eventos. Ejemplo de estos eventos: oportunidad de ataque por la izquierda, distracción del portero, obstrucción de visibilidad de la portería.

d) De la misma forma, la selección de las estrategias estará limitada a la visión del entorno que tengan los agentes; la selección de una estrategia determinada, la realizará el agente que posea el rol del *líder*.

e) Solo una estrategia puede ser ejecutada a la vez.

f) La selección de la estrategia estará determinada por las oportunidades comunicadas por los agentes, quienes informarán sobre su posición con respecto a la cancha y de los adversarios adyacentes, esto con el fin de identificar, de la base de estrategias, la más adecuada de acuerdo a un margen o rango de posiciones.

1.8. METODOLOGÍA

Para lograr los objetivos previamente planteados se define la siguiente metodología de trabajo:

a) Identificación de un dominio de aplicación del modelo. El dominio de aplicación es el entorno de juego de la categoría de robots cuadrúpedos del proyecto internacional *RoboCup*.

b) Identificación de la metodología para la resolución de problemas de toma de decisión. Se hará uso del razonamiento basado en casos.

c) Selección de una plataforma de desarrollo para simulación. La plataforma de desarrollo está basado sobre *JAVA* y *XML*.

d) Aplicación del diseño, implementación y pruebas de la solución propuesta.

1.9. ORGANIZACIÓN DE LA TESIS

El presente documento esta organizado de la siguiente forma:

El capítulo 2 aborda a detalle lo relacionado al Estado del Arte, el dominio de aplicación del modelo, breve descripción del proyecto internacional *RoboCup*, el proyecto de investigación del Instituto Tecnológico y de Estudios Superiores de Monterrey, campus Estado de México *TecRams*; definiciones sobre la metodología de análisis y diseño del sistema multiagentes, entre las que destacan agente y sistemas multiagentes y trabajos relacionados con su aplicación; aspectos relacionados a la metodología aplicada para la resolución de problemas de toma de decisión, razonamiento basado en casos y trabajos relacionados con su aplicación y finalmente, información relevante sobre la plataforma de desarrollo del entorno de simulación del sistema multiagentes *JAVA* y *XML*.

El capítulo 3 trata los trabajos previos de investigación realizados en el proyecto de investigación del ITESM, campus Estado de México así como la metodología de trabajo actual y la confirmación de la importancia de la incorporación de una arquitectura de estrategias para el trabajo colaborador entre agentes.

El capítulo 4 presenta la propuesta de investigación, al incluir el análisis y diseño del modelo de sistema multiagentes, estructura de la arquitectura de estrategias, aplicación del razonamiento basado en casos como mecanismo para la toma de decisiones con base en el principio de semejanza.

El capítulo 5 plantea la generación de experimentos y evaluación de resultados.

El capítulo 6 resume las conclusiones del trabajo de investigación y trabajos futuros propuestos.

El apéndice A presenta un listado de los principales acrónimos utilizados en este documento; el apéndice B lista algunos de los métodos implementados en la aplicación propuesta tanto en *JAVA* como en *XML*; el apéndice C provee el manual de usuario y el apéndice D suministra el manual de programación.

336403
BIBLIOTEC
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MEXICO

2. ESTADO DEL ARTE

"La mayoría de las ideas fundamentales de la ciencia, son esencialmente sencillas y, por regla general, pueden ser expresadas en un lenguaje comprensible para todos"

Albert Einstein

Los agentes habían sido visualizados, en un inicio, como un sistema, que definido por objetivos, podría llevar de forma apropiada las operaciones en computadora, pero esta idea se ha desarrollado de tal manera que a los agentes y los sistemas multiagentes se les ha dotado de capacidades de razonamiento, aprendizaje y habilidad social, entre otras. En este capítulo se abordará aspectos relacionados al marco teórico que incluye definiciones básicas de estrategia, agentes y sistemas multiagentes, así mismo un breve recorrido por el fútbol robótico y las diferentes tecnologías que han apoyado ya sea por sí mismas o una coalición de las mismas, a dar solución a los diferentes retos propuestos por esta competencia.

2.1. ¿QUÉ ES UNA ESTRATEGIA?

Según la *Real Academia Española* una estrategia es el arte de dirigir operaciones militares, así mismo, el arte de trazar para dirigir un asunto, un conjunto de reglas que aseguran una decisión óptima; proviene del latín *strategîa* y del griego $\sigma\tau\rho\alpha\tau\eta\gamma\iota\alpha$, en [Oxford, 2002] se define como un plan para tratar con circunstancias futuras inciertas; un juego de reglas por las cuales las acciones a ser tomadas dependen de las circunstancias actuales, incluso acontecimientos naturales y las acciones de otras personas. Es un término que se aplica en distintos contextos entre ellos militares, comerciales, políticos y de juego.

En la teoría de juego presentada en [Oxford, 2003], se marca una diferencia entre *estratégico* y *táctico*. Una estrategia es un plan para tratar cada movimiento posible del jugador o jugadores en cada etapa del juego. El número de estrategias de apertura de un jugador en un juego de cualquier complejidad es astronómico; incluso en un juego trivial como el tres en raya (tic-tac-toe) el primer jugador tiene nueve movimientos de apertura legales. Sin embargo, el análisis de estrategias buscar la forma de eliminar algunas de ellas, o de forma habitual seleccionar la mejor estrategia posible. Así mismo en [Britannica, 2008] se define táctica como la disposición o alineación de los jugadores o contendientes en el terreno de juego o batalla.

Para que el uso de estrategias y tácticas sea exitoso, es necesario aplicar métodos de *coordinación* a fin de concertar medios y esfuerzos para conseguir un objetivo común entre los miembros de un equipo de cualquier índole. A este punto se introducirá el termino de *agente* como integrante de un equipo, así mismo, el grupo de agentes que comparte un objetivo a largo plazo se les denominará *equipo* y los agentes que participan en un mismo entorno pero poseen metas contrarias a nuestro *equipo* se llamarán *oponentes*.

Tal como se comenta en [Stone, 2000] los sistemas multiagentes (SMA), en voz inglesa MAS (*Multiagent System*), como subcampo de la inteligencia artificial (IA), en voz inglesa AI (*Artificial Intelligence*), proveen los principios para la construcción de complejos sistemas que involucran *agentes* y mecanismos para la *coordinación* de *comportamientos* de agentes por medio de la adopción de procesos cognitivos y acciones de interacción entre ellos. Estos temas serán abordados en las secciones posteriores.

2.2. AGENTE Y SISTEMAS MULTIAGENTES

Es común que los sistemas de computo que trabajan en nuestro beneficio deban relacionarse con otros sistemas de computo que representan el interés de otro u otros, estamos conectados bajo un red invisible de colaboración que no siempre funciona de la mejor forma, es por esto que es necesario dotar a los sistemas de computo de capacidad de cooperar y alcanzar acuerdos con otros sistemas de computo, tal como sucede en la vida real. Esta capacidad ha propiciado el desarrollo de uno de los mas novedosos campos de las ciencias computacionales: agentes y sistemas multiagentes. Estos temas serán analizados a continuación.

2.2.1. AGENTES INTELIGENTES

El concepto de agente computacional se remonta a los principios de la informática. Alan Kay, un científico americano, declaró en 1984 acerca de los orígenes de este concepto.

> *"La idea de una agente se originó con John McCarthy a mediados de los 50's y el termino fue aplicado por Oliver G. Selfridge pocos años después, cuando ambos asistían al Instituto de Tecnología de Massachusetts. Ellos habían visualizado un*

sistema que, definido un objetivo, podría llevar a cabo de forma apropiada las operaciones en computadora y, este sistema, podría pedir ayuda, en términos humanos, cuando lo necesitara. Un agente podría ser un 'robot de software' viviendo y haciendo negocios dentro del mundo de las computadoras"

Tomado de [Ye and Churchill, 2003]

Características de los Agentes

Diversos autores han proporcionado una definición de lo que es un *agente*, hasta ahora se ha logrado llegar a un consenso sobre diversas características o cualidades que deben ser inherentes en un agente, como por ejemplo la *autonomía*, que significa que los agentes deben ser capaces de actuar sin la intervención humana o de otros sistemas de computo. En [Wooldridge and Jennings, 1995] se realiza la siguiente definición:

"Un agente es un sistema de computo que esta situado en algún ambiente, y que es capaz de actuar de forma autónoma en este ambiente a fin de alcanzar los objetivos de diseño"

Así mismo se confirma en [Wooldridge, 2001] otros aspectos deseables en un agente inteligente:

a) *Reactivo*: los agentes inteligentes son capaces de percibir su ambiente, y responder en una manera oportuna a cambios que ocurren en este, a fin de satisfacer sus objetivos de diseño.

b) *Iniciativa*: los agentes inteligentes son capaces de exponer el comportamiento dirigido por objetivo, tomando la iniciativa a fin de satisfacer sus objetivos de diseño.

c) *Habilidad social*: los agentes inteligentes son capaces de interactuar con otros agentes, para satisfacer sus objetivos de diseño.

Es muy común confundir a un agente con un objeto. Un *objeto* es una entidad computacional que encapsula algún estado, es capaz de realizar acciones o métodos en ese estado y comunicarse mediante el envío de mensajes. Pero una de las diferencias mas notables es el grado de autonomía que un agente posee con respecto a un objeto, este último puede manifestar control sobre el estado en el que se encuentra, más no control sobre su comportamiento. Una frase que distingue la diferencia entre objeto y agente es:

"Los objetos lo hacen de gratis, los agentes lo hacen por que quieren o por dinero"

Un aspecto importante en cuanto agente inteligente es identificar cómo decirle a un agente lo que debe hacer y cómo adquiere intenciones, creencias, conocimientos, deseos y distintas capacidades. Una forma sencilla de hacerlo es escribir un programa que el agente ejecute, pero con esto no preparamos al agente para responder a circunstancias imprevistas, las metas en sí, no son suficientes para generar un comportamiento eficiente. En otras palabras es deseable decirle al agente qué hacer sin decirle cómo.

Un modo de lograr que el agente ejecute alguna acción sin decirle cómo, es definir tareas de forma indirecta, por medio de una especie de medida de interpretación; una de ellas es la función de *utilidad*, que permita la comparación entre diferentes estados del mundo, dependiendo de la mayor utilidad que el cumplimiento de una meta, pueda generar para un agente. Una *utilidad* es una representación numérica de lo 'bueno' que el estado es: entre más alta la utilidad, mejor. La tarea del agente es procurar estados que maximicen la utilidad.

$$u : E \rightarrow R$$

De esta forma se puede afirmar que el agente inteligente ideal es aquel que, luego de analizar su entorno, maximiza la medida de interpretación, con base en la evidencia provista por la percepción y el conocimiento previo con el que cuente. Para implementar tal especificación, existen dos posibilidades: tratar de desarrollar un algoritmo que *sintetice automáticamente* a tales agentes, con las especificaciones de ambiente de tarea; o podemos tratar de desarrollar un algoritmo que *ejecute directamente* especificaciones de agente a fin de producir el comportamiento apropiado.

Arquitecturas de los Agentes

En [Weiss, 1999a] se consideran cuatro clases de agentes, que consecuentemente son identificados por muchos autores, como las arquitecturas de agentes; estas son:

a) Agentes lógicos: la toma de decisión se realiza mediante deducción lógica. Apego a la representación simbólica del medio y del comportamiento. Complejidad computacional para la comprobación de teoremas.

b) Agentes reactivos: la toma de decisión se realiza con la implementación de alguna forma de correlación directa, de la situación a la acción. Rechazo a la representación simbólica. El comportamiento inteligente emerge de la interacción de varios comportamientos más sencillos

c) Agentes creencia-deseo-intención: la toma de decisión se realiza en dependencia de la manipulación de estructuras de datos, que representan las creencias, deseos, e intenciones del agente. Basado en el razonamiento práctico de la vida diaria, incluye dos procesos importantes: deliberación (decidir que metas se desean) y la planeación (decidir cómo llevar a cabo las metas); y finalmente,

d) Arquitecturas híbridas: la toma de decisión es realizada por medio de varias capas de software, en los cuales el razonamiento se realiza sobre el ambiente en niveles diferentes de abstracción. Comportamiento reactivo y deliberativo.

Cada una de las arquitecturas posee características que las diferencian una de la otra, pero de forma general se puede abstraer el comportamiento de los agentes mediante el análisis de los cambios de estados por los que transita, a medida que el medio esta cambiando; de esta forma los estados de un agente pueden ser definidos por un conjunto $S = s_1, s_2, ...$ de *estados del ambiente*. En cualquier

momento dado, el ambiente se encuentra en algún estado definido por el conjunto S. Las capacidades de los agentes esta representados por el conjunto $A = a_1, a_2, ...$ de *acciones*. De esta forma se puede intuir que un agente puede ser visto como una función de:$ACCION : S* \rightarrow A$.

En la Figura 2.1 se muestra un análisis más detallado del comportamiento de un agente basado en estados. El agente inicia en algún estado interno inicial i_0. Luego realiza observaciones del estado del ambiente S, mediante $VER(S)$. El estado interno del agente, es actualizado por medio de la función `SIGUIENTE`, de esta forma se realiza $SIGUIENTE(i_0, VER(S))$. El comportamiento seleccionado por el agente, esta dado entonces por $ACCION(SIGUIENTE(i_0, VER(S)))$. Una vez que el agente ejecuta la acción, el agente ingresa al ciclo, percibe el mundo a través de `VER`, actualiza su estado por medio de `SIGUIENTE`, y selecciona su estado vía `ACCION`.

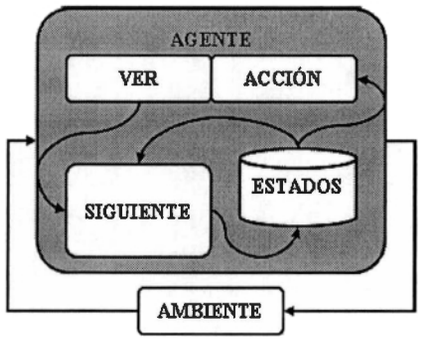


Figura 2.1: Comportamiento de un agente basado en estados

2.2.2. SISTEMAS MULTIAGENTES

La habilidad social, definida anteriormente, como la capacidad del agente inteligente de interactuar con otros agentes, para cumplir sus objetivos de diseño, propicia el entorno de sistemas computacionales integrados por muchos agentes que colaboran entre sí, para solucionar una diversidad de problemas. Así mismo el establecimiento de muchas disciplinas, incluyendo la inteligencia artificial, ciencias informáticas, sociología, economía, filosofía entre otras, favorecieron el fortalecimiento del estudio de la inteligencia artificial distribuida (IAD); En [Weiss, 1999b] se hace la siguiente introducción al termino:

> *"La IAD es el estudio, construcción, y aplicación de sistemas multiagentes, es decir sistemas en los cuales los agentes inteligentes interactúan con la finalidad de perseguir objetivos o realizar algún tipo de tareas en conjunto"*

Posteriormente, el mismo autor, hace una distinción de dos tipos primarios de sistemas de IAD y SMA: los sistemas multiagentes en los cuales varios agentes coordinan su conocimiento, actividades y razón sobre procesos de coordinación; y los sistemas de solución de problemas distribuidos en los cuales el trabajo de solucionar un problema particular es dividido entre varios nodos que se dividen, incluyendo el conocimiento sobre el problema y la solución en muchas vías.

Los SMA se diseñan e implementan para permitir la interacción entre varios agentes. La comunicación es una forma de interacción en la cual se permite el intercambio de mensajes y realizar conversaciones entre los agentes, lo cual favorece la ejecución de las metas de cada uno de los agentes. Es necesario hacer notar que estas metas no necesariamente son comprendidas por todos los agentes y la comunicación permite la coordinación de las acciones y comportamientos.

2.2.3. COORDINACIÓN: COOPERACIÓN Y NEGOCIACIÓN EN SMA

La *coordinación* definida en [Weiss, 1999c] es una propiedad de los sistemas de agentes que ejecutan alguna actividad en un medio compartido, entre mejor sea la coordinación mejor estarán los agentes preparados para reducir riesgos ante cualquier eventualidad. La *cooperación* es coordinación entre agentes no antagónicos, mientras que la *negociación* es la coordinación entre agentes en un medio competitivo o simplemente para satisfacer los intereses de cada uno de los agentes. En una cooperación exitosa, cada agente debe mantener un modelo de los otros agentes y de la misma forma, construir modelos de interacciones futuras. En la Figura 2.2 se muestra diferentes vías en las cuales los agentes se coordinan.

Figura 2.2: Coordinación entre agentes

En [Ferber, 1999] se hace la siguiente definición de cooperación:

> *"Se dice que varios agentes están cooperando, o están en una situación de cooperación, si una de las dos condiciones siguientes es verificada: (1) la adición de un nuevo agente hace posible incrementar los niveles de ejecución de un grupo de forma diferencial y (2) la acción de los agentes ayuda a evitar o resolver conflictos potenciales o actuales"*

Comunicación

Los protocolos de comunicación permiten que las conversaciones e interacciones se lleven a cabo de forma organizada y efectiva, tal como en la vida real. Algunos tipos de mensajes son: proponer una acción, aceptar un acción, rechazar un acción o realizar una contrapropuesta a una proposición. En [Ferber, 1999] se afirma que los lenguajes de comunicación constituyen la 'clave' que hace posible que los agentes heterogéneos puedan coordinar sus acciones, cooperar para un objetivo común y en conjunto formar un grupo integrado de agentes. La comunicación puede realizarse de uno a uno o de uno a muchos.

En la búsqueda de un ideal en cuanto a lenguajes de comunicación entre agentes se refiere, en voz inglesa *Agent Communication Language (ACL)*, se han destacado entidades como FIPA [1] y el Esfuerzo de Compartimiento de Conocimiento en voz inglesa *Knowledge Sharing Effort*, quienes han preparado la plataforma por medio de la cual todos los agentes serían capaces de relacionarse sin tener en cuenta su ambiente de ejecución. Así mismo han definido un modelo para ACL que incluye tres capas de representación: la primera especifica actitudes proposicionales, la segunda especifica proposiciones (conocimiento) y la tercera especifica la ontología de estas proposiciones.

De esta forma entre los protocolos y lenguajes más comunes utilizados para el intercambio de información y conocimiento entre agentes, se destacan el *KQML* (Knowledge Query and Manipulation Language) como lenguaje para el intercambio de información y conocimiento por medio de performativas (actitudes proposicionales) y el *KIF* (Knowledge Interchange Format) propuesto como un estándar de uso para describir aspectos relacionados a los sistemas expertos, base de datos y sistemas inteligentes, especificamente para servir como "interlingua," o mediador en la traducción de otros lenguajes (proposiciones) tal como se refiere en [Weiss, 1999b].

KIF expresa el contenido de una proposición mientras que *KQML* expresa la actitud del agente hacia la proposición. De forma básica el protocolo *KQML* esta definido por la estructura compuesta por: performativas de *KQML*, emisor, receptor, lenguaje, ontología. A continuación se presenta la estructura básica y de la misma forma un ejemplo.

```
(KQML-performativas
      :emisor<palabra>
      :receptor<palabra>
      :lenguaje<palabra>
      :ontología<palabra>
      :contenido<palabra>
...)
```

```
(reply
      :emisor<agente 2>
      :receptor<agente 1>
      :lenguaje<java>
      :ontología<fútbol robótico>
      :contenido<pos: -140,-180>
...)
```

[1] *Foundation for Intelligent Physical Agents* (Fundación para los Agentes Físicos Inteligentes `http://www.fipa.org`

Similar a KQML, *FIPA-ACL* es otro lenguaje de comunicación que se basa en la teoría del acto del discurso, la especificación consiste en un juego de tipos de mensaje y la descripción de su pragmática, es decir, los efectos en las actitudes mentales de agentes de receptor y el remitente. La especificación describe cada acto comunicativo, tanto con una forma de narrativa, como con una semántica formal basada según la lógica modal. Esto también proporciona la descripción normativa de un juego de protocolos de interacción de alto nivel, incluso la solicitación de una acción, red de contrato, y varias clases de subastas tal como se explica en [Labrou and Finin, 2000].

Red de Contratantes

Red de Contratantes en voz inglesa *Contract Net* es un protocolo de interacción donde los agentes cooperan para la resolución de problemas. Se aplican roles de agente como el de *administrador*, quien supervisa la ejecución de una tarea y procesa el resultado; y el de *contratado*, el responsable de la ejecución de una tarea. Cualquier agente puede ser *administrador* al anunciarse las tareas, de la misma forma, *contratado*, al responder al anuncio de las tareas.

El proceso básico de *Red de Contratantes* se realiza de la siguiente manera:

1) Un administrador anuncia a todos los agentes la existencia de una tarea vía un mensaje multidifusión, en voz inglesa *multicast*.

2) Los agentes evalúan el anuncio y pueden seleccionar el abandonar la solicitud o aplicar a esta.

3) El administrador recibe y evalúa las licitaciones de los posibles contratados.

4) El administrador concede un contrato al agente mas apropiado.

5) El agente contratado realiza la tarea.

6) El agente contratado reporta resultados al administrador.

Existen otros protocolos de interacción entre agentes además de la red de contratantes tales como los sistemas de pizarra, la negociación y las subastas (mecanismos de mercado), pero para efectos de esta investigación se hace énfasis en la red de contratantes, por ser parte funcional del esquema implementado en la herramienta de simulación que se analizará con mayor detalle en el siguiente capítulo. Junto a este protocolo de interacción también se abordara los mecanismos de coordinación y cooperación vitales para la herramienta propuesta.

Interacción Personas y Agentes

Además de las atributos mencionados, en lo que respecta a los agentes, como son *autonomía, reactivo, iniciativa y habilidad social*, se deben destacar otras cualidades como son: comportamiento colaborador, habilidad de comunicación, personalidad, movilidad, habilidad de aprendizaje, adaptabilidad, entre otros. Un número de autores han propuesto varias esquemas de clasificación

y taxonomías, tomando en cuenta todas estos atributos, haciendo énfasis en algunos y restando importancia a otros.

Un listado de taxonomías o clasificación incluye: *Agentes Colaboradores*, *Agentes de Interfaz*, *Agentes Móviles*, *Agentes de Información/Internet*, *Agentes Reactivos*, *Agentes Híbridos*, *Agentes Inteligentes*; estas complementan el listado de arquitecturas visto con anterioridad. Es imposible crear taxonomías de agentes sin que algunos de los atributos se traslapen entre las categorías.

En [Ye and Churchill, 2003] se propone un esquema de categorización basado en personas, agentes y su interacción; donde en cada categoría, los agentes podrían poseer varios grados de atributos, como grado de autonomía, movilidad, iniciativa y así sucesivamente. En el Cuadro 2.1 se presenta una perspectiva de esta categorización y algunos ejemplos.

	Sin Usuario	Un Usuario	Usuarios Múltiple
Un Agente	Un Robot	Agente de Interfaz	Agente de Apuestas en Internet
	Agente de Software sencillo	Agente de Entretenimiento	Agente de soporte a usuarios múltiple en juegos por Internet
	Agente de Software orientado a tarea	Agente de Búsqueda en Internet	Agente que organiza calendario y agenda
	Agente de Aprendizaje	Agente para compras	Agente coordinador
		Agente interfaz Usuario-Máquina	
		Agente de Microsoft	
		Agente de aprendizaje y tutoría	
Agentes Múltiples	Agentes Múltiples de Software sencillo	Agentes de viaje por Internet paquetes vacacionales	Agentes que coordinan actividad humana
	Agentes Múltiples de Software orientado a tareas	Agentes que proveen servicios centralizado al usuario	Agentes de negocio, agentes múltiples negocian convenios múltiples
	Equipo de Robots		Agentes que se adaptan a la organización real y virtual
	Agentes de Vida Artificial		Ecosistemas Agente-Humano

Cuadro 2.1: Perspectiva de taxonomías de agentes

La combinación *un agente - sin usuario*, se refiere a los agentes que no tienen interacción directa con las personas; el trabajo que se ha realizado en relación a esta interacción, se constituye en el diseño de agentes que llevan a cabo acciones dentro de la IA y comunidad de agentes. Así mismo la mayor parte del trabajo realizada a la fecha en IAD se ha enfocado en la interacción *agentes múltiples - sin usuario*; ejemplo de esto son los agentes robóticos que juegan en un entorno de fútbol robótico, como se verá a continuación.

2.3. FÚTBOL ROBÓTICO

El fútbol robótico es una creciente área de interés para la robótica y la inteligencia artificial; esto se debe a diversas razones entre las cuales se puede mencionar la complejidad del dominio, dinámico y de incertidumbre, junto al conjunto definido de reglas que gobiernan el comportamiento de los jugadores en dicho dominio; al mismo tiempo, es una temática representativa para la aplicación de sistemas multiagentes, debido a que representa situaciones donde se pueda modelar el mundo real.

Así mismo en [Malec, 2000, Groen et al., 2002] se mencionan algunos de los principales elementos que se presentan en aplicaciones de los SMA, tales como: percepción de ambientes dinámicos, comportamiento reactivo, coordinación de equipos, comunicación y fusión de información, toma de decisiones, entre otros. En el área de robótica e inteligencia artificial pueden ser explotados aspectos como, diseño e implementación de robots móviles, arquitecturas de control, arquitecturas de agentes, cooperación, comportamiento colectivo y oportunismo.

A continuación se describen dos de las principales organizaciones que aplican la robótica e inteligencia artificial, para solventar problemáticas del mundo real, utilizando el ambiente del fútbol robótico como entorno para el estudio y la investigación.

2.3.1. *FIRA*

FIRA es la asociación internacional de fútbol robótico, establecida en junio de 1997 durante el torneo de micro robot *MiroSot'97* en Daejon, Corea, desde entonces organiza un torneo anual de escala mundial, promueve el *FIRA Cup* y el *FIRA Robot World Congress*, en el cual los participantes presentan artículos de investigación sobre sus robots y las metodologías utilizadas, con el fin de compartir las experiencias y tecnología, en la construcción de robots que participan en un enfrentamiento de fútbol robótico, ver enlace a sitio web en [FIRA, 2008].

Dentro de los objetivos que persigue *FIRA* están: motivar el espíritu científico y tecnológico en las nuevas generaciones; promover el desarrollo de sistemas autónomos multiagentes, que puedan cooperar mutuamente y contribuir al desarrollo del estado del arte en campos especializados de la tecnología; proveer el trabajo colaborador entre investigadores y estudiantes de las distintas ramas como: robótica, fusión sensorial, control inteligente, comunicación, procesamiento de imágenes,

mecatrónica, tecnología de la computación, vida artificial, y el nuevo y creciente campo de los robots autónomos inteligentes que juegan fútbol robótico; organizar anualmente el *FIRA Robot World Cup*; y trabajar de forma conjunta para establecer el *FIRA Robot World Cup* como una copa mundial de ciencia y tecnología.

A la fecha se han definido seis categorías que compiten en *FIRA Robo Soccer*. Para cada categoría se definen las reglas del juego así como los lineamientos sobre el diseño del robot. En la mayoría de los casos dos equipos compiten sobre un cancha de juego, con la finalidad de obtener el mayor puntaje. En el Cuadro 2.2 se presenta el listado de las categorías que compiten bajo *FIRA Robo Soccer*, esta información ha sido tomada del sitio web de FIRA.

Categoría	Descripción
HuroCup	Un robot humanoide aplica siete pruebas de forma autónoma
KheperaSot	Un robot de forma cilíndrica compite de forma autónoma
MiroSot	a) Middle - cinco robots b)Large - once robots, compiten de forma autónoma
NaroSot	Cinco robots compiten de forma autónoma
AndroSot	Tres robots compiten a control remoto (humano)
RoboSot	De uno a tres robots compiten de forma autónoma o semi autónoma (computadora)

Cuadro 2.2: Categorías de *Robo Soccer* en FIRA

2.3.2. *ROBOCUP*

RoboCup es una iniciativa internacional que promueve la investigación en el campo de la inteligencia artificial y robótica con el fin de desarrollar una amplia gama de tecnologías que puedan ser integradas y evaluadas en la resolución de diversos tipo de problemas ya sea de orden social o industrial.

De la misma forma, el proyecto impulsa la creación e innovación en materia educativa al concentrar sus actividades en el desarrollo de conferencias técnicas, competencias internacionales, programas de retos, programas educativos y desarrollo de infraestructura, los dos primeras aspectos, conferencias y competencias, conforman las principales actividades de *RoboCup*, ver enlace a sitio web en [RoboCup, 2008].

Los orígenes de *RoboCup* se remontan al año 1997, consecuentemente ese año sería recordado como un punto importante en la IA y robótica al presenciar eventos de gran magnitud y relevancia, como el enfrentamiento del destacado jugador de ajedrez, Gary Kasparov contra *IBM Deep Blue*,

así mismo, en julio de ese año *NASA*[2] realizó el descenso exitoso en Marte con el primer sistema robótico autónomo *PathFinder*.

La primera competencia oficial de *RoboCup* se llevó a cabo en agosto de 1997 en Nagoya, Japón con una participación de 40 equipos entre simulación y robots reales.

El objetivo principal de *RoboCup* es formar un equipo de robots humanoides que pueda competir con el equipo humano ganador de la copa mundial de fútbol en el año de 2050 bajo las mismas reglas de la FIFA[3]. Para lograr esto último, *RoboCup* aplica novedosos principios de diseño de agentes autónomos, colaboración de multiagentes, estrategia, razonamiento en tiempo real y robótica en general, bajo ambientes dinámicos. A la fecha *RoboCup* cuenta con cuatro divisiones con sus correspondientes categorías, ver Cuadro 2.3 para mayor detalle.

Tipo	Categoría	Descripción
RoboCupSoccer	*Standard Platform*	a)Robots humanoides (*Aldebaran Nao*) b)Robots cuadrúpedos (*Sony AIBO*)
	Humanoid	a) Kid Size (30-60cm de altura) b) Teen Size (80-130cm de altura)
	Middle Size	Robots medianos con sensores a bordo
	Small Size	Robots pequeños, cámara superior
	Simulation	a) Simulation 2D b) Simulation 3D c) Mixed Reality d) 3D Development
RoboCupRescue	*Rescue Simulation*	a) Agent and Infrastructure b) Virtual Robots
	Rescue Robot	Robots exploran un área construida como sitio de desastre
RoboCupJunior	*Dance*	Uno o mas robots que danzan al compás de la música
	Soccer	Robots que juega fútbol robótico
	Rescue	Robots que identifican víctimas al recrear escenarios
	Demo	Variado, puede ser competencia de fútbol
RoboCup@Home	*General*	Aplicaciones del mundo real e interacciones hombre-máquina

Cuadro 2.3: Categorías de *RoboCup*

En la Figura 2.3 se muestra dos de los robots que participan en *RoboCup* en la categoría estándar.

[2]*National Aeronautics and Space Administration* (Administración Nacional de Aeronáutica y del Espacio), `http://www.nasa.gov/`

[3]*Fédération Internationale de Football Association* (Federación Internacional de Fútbol), `http://www.fifa.com/`

(a) Nao (b) AIBO

Figura 2.3: Robots en Categoría Estándar de RoboCup

2.4. SMA APLICADO AL FÚTBOL ROBÓTICO

Los sistemas multiagentes aplicados al fútbol robótico son un gran reto el área de inteligencia artificial y han sido desarrollados como representación de fútbol humano al tomar como base múltiples disciplinas de investigación, a saber: robótica, control de inteligencia, visión computacional, entre otros. En un sistema de múltiples robots, la estrategia de acción, es objeto de estudio relacionado con el desempeño en los sistemas y grados de inteligencia.

Coordinación y adaptación son dos elementos críticos para el desarrollo de equipos de robots que ejecutan tareas útiles, especialmente cuando el ambiente no está bajo control o actúan otros agentes. A continuación se abordará algunas aspectos propuestos en algunas metodologías para resolver de forma integral o específica el trabajo colaborador o coordinado en un sistema multiagentes.

2.4.1. RAZONAMIENTO BASADO EN CASOS

El razonamiento basado en casos (RBC), en voz inglesa CBR (*Case-Based Reasoning*), puede ser definido como un modelo de razonamiento que incorpora la resolución de problemas, entendimiento y aprendizaje, además, puede integrar todos estos aspectos en procesos de memoria. Según [Pal and Shiu, 2004, Pal et al., 2001] éstas tareas son realizadas usando situaciones típicas llamadas *casos*, experimentados previamente por un sistema.

Un *caso* puede ser definido como una pieza dentro de un contexto de conocimiento que representa una experiencia que muestra una lección fundamental para lograr los objetivos del sistema. Este se vuelve más eficiente y competente como resultado del almacenamiento de la experiencia

pasada y haciendo referencia a ésta en un proceso posterior.

Los sistemas RBC son una alternativa, en muchos casos, de los *sistemas basados en reglas*, en voz inglesa *Rule-Based System*, pero aspectos como la adquisición de conocimiento y representación formal de un dominio de aplicación, puede llegar a consumir muchos recursos entre ellos el tiempo.

En un sistema basado en reglas, la base de una regla consiste en un conjunto de condiciones de la forma: `IF A, THEN B`, donde `A` es una condición y `B` es una acción, si `A` es verdadera *entonces* `B` se lleva a cabo. Los sistemas basados en casos usualmente requieren menos adquisición de conocimiento; además es útil en las aplicaciones donde los dominios cuentan con poca información o pocos casos pueden ser definidos. Esto nos lleva a puntualizar en uno de los beneficios más sobresalientes de RBC presentados en [Pal et al., 2001].

> *" Un sistema puede ser creado con una pequeña o limitada porción de experiencia y posteriormente desarrollarla de forma incremental, agregando mas casos a la base de casos y haciendo estos disponibles para su aplicación"*

Como se muestra en la Figura 2.4, el ciclo de vida de la resolución de problemas en un RBC consiste en cuatro procesos principales:

a) Recuperar: casos similares experimentados anteriormente cuyo problema es juzgado como similar.

b) Reusar: copiando o integrando las soluciones de los casos recuperados.

c) Revisar: o adaptar la(s) solución(e)s recuperadas para resolver de forma tentativa un nuevo problema.

d) Retener: la nueva solución una vez que ha sido confirmada o validada.

Diferentes técnicas pueden ser usadas y aplicadas en cualquiera de las fases del ciclo de vida del RBC. Por ejemplo dos de las técnicas más ampliamente usadas en la fase de recuperación corresponden a: *vecino mas cercano*, en voz inglesa *nearest-neighbor* y técnicas de inducción, en voz inglesa *inductive retrieval*, como *ID3 y C4.5*, así mismo otras técnicas de clasificación como *c-medias*, lógica difusa, bases de datos, redes neuronales, entre otros pueden utilizarse en alguna de las fases de RBC. La ecuación:

$$similitud(p,q) = \sum_{j=1}^{n} f(p_i, q_i) \times w_i \tag{2.1}$$

representa una técnica típica de la técnica del *vecino mas cercano* que describe una situación para la cual `p` y `q` son dos casos en los que se compara similitud o proximidad, `n` es el número de atributos en cada caso, `i` es un atributo individual de 1 a `n`, y w_i es el peso característico del atributo `i`, f es la función del atributo i en casos p y q. Generalmente, el cálculo de similitud, continua hasta que todos los casos en la base de casos sean comparados, y son normalizados para caer en un rango de

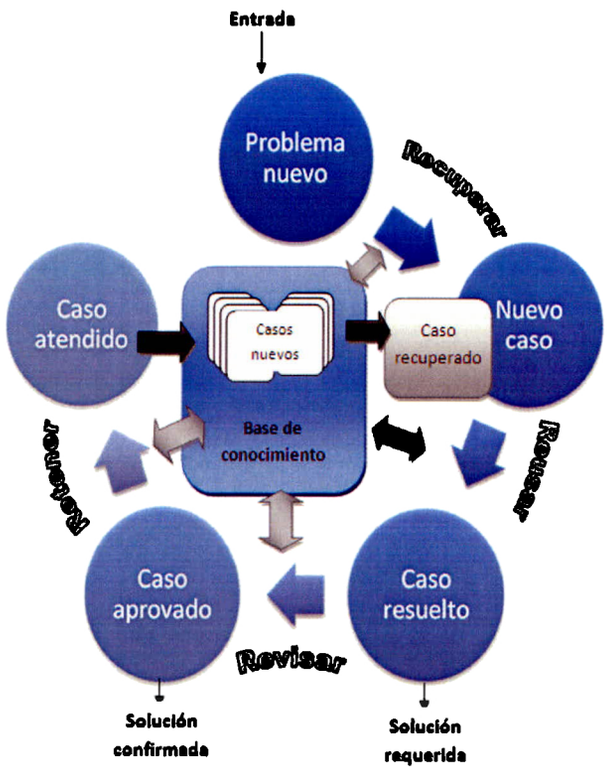


Figura 2.4: Ciclo del RBC

0 a 1 (donde 0 es total diferencia y 1 es una coincidencia exacta).

Así mismo en [Balaraman, 2004] se estima que la medida de similitud es una de las principales medidas para asegurar la calidad de los sistemas RBC. La mayoría de las medidas de similitud también incorporan elementos de configuración que permiten realizar afinamientos o ajustes para obtener resultados mas eficientes.

Algunas aplicaciones de RBC, presentadas en [Watson, 1997] corresponden a una variedad de demostraciones académicas como son: diagnóstico (de enfermedades, de mecánica automotriz, defectos de construcción), planeación (de vuelos, de asuntos bélicos, de aspectos de manufactura), razonamiento legal (de sentencia criminal, de ley de patentes, de regulaciones en la construcción), diseño (de áreas de servicio, de menús de restaurante, de automotores), razonamiento análogo (entre situaciones, relaciones entre las muerte de animales y humanos), arbitraje (resolución de disputas), adaptación (crear nuevas recetas a partir de las anteriores), de planes (a partir de las situaciones nuevas, nuevas jugadas de fútbol a partir de las anteriores) y, tutoría (resolución de problemas pedagógicos, generación de nuevos casos para estudiantes de leyes).

Así mismo existe una diversa cantidad de aplicaciones industriales que hacen uso del razonamiento basado en casos, muchas de las principales compañías de todos los sectores han aplicado esta tecnología para dar solución a variados problemas entre estos: hardware (soporte técnico, manejo

de eventos de error), software (soporte técnico), finanzas y seguros (evaluación de créditos, análisis de datos), telecomunicaciones (administración de red, soporte técnico), manufactura y transporte (sistemas de reservación, mantenimiento de vehículos, procesos de control), venta al por menor (soporte técnico), entre otros.

Aplicación en Fútbol Robótico

El RBC aplicado al fútbol robótico no ha sido ampliamente explotado, existen pocos recursos de información que se relacionan a esta temática.

Una aplicación presentada en [Marling et al., 2003] describe el desarrollo, en simulación, de tres prototipos que aplican razonamiento basado en casos para *RoboCats*, un equipo de cinco jugadores que participan en la categoría de *Small-Size* de *RoboCup*. Los prototipos son: *a)- localización del portero, b)- selección de la formación del equipo* y, *c)- reconocimiento del estado del juego.* En la liga de *Small-Size* de *RoboCup*, la visión se provee mediante una cámara que se localiza en la parte superior del terreno de juego, esto permite obtener una información global del entorno, posteriormente esta información en enviada a cada uno de los jugadores.

En el primer prototipo de *RoboCats*, un *caso* se deriva de la información que se abstrae acerca de posición y orientación del portero, así mismo la información general del entorno y las acciones llevadas a cabo por el portero, como respuesta a un comportamiento de bloqueo; en la Figura 2.5 se observa la posición inicial y final del portero luego de aplicar un comportamiento de bloqueo, las líneas discontinuas representan la poción inicial y las líneas continuas la posición actual.
Sin embargo las primeras implementaciones de RBC aplicado al fútbol robótico, como apoyo en

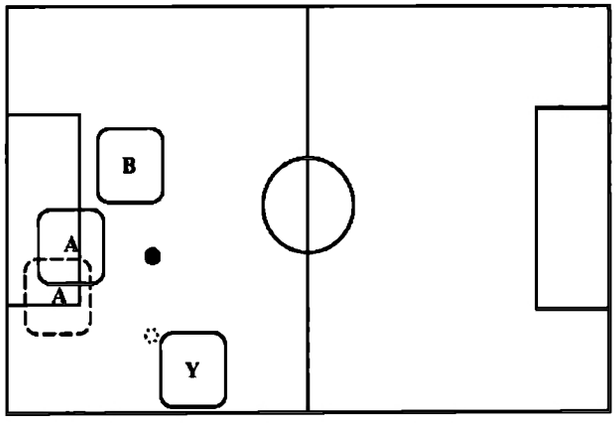


Figura 2.5: Ejemplo de la definición de un caso en *RoboCat*

los procesos de decisión, fueron aplicados por el equipo de simulación de fútbol de Humboldt-Universität Berlín tal como se refiere en [Wendler and Lenz, 1998], en la cual un jugador individual determinaba el siguiente movimiento, basado en un movimimiento anterior realizado en una situacion de juego similar. El equipo alemán ha continuado los trabajos sobre la base de esta

metodología, tal como lo reporta la descripción técnica del equipo disponible en [Humboldt, 2007].

El trabajo realizado presenta una arquitectura de "pase-doble" capaz de seleccionar e iniciar jugadas complejas para varios jugadores, a través de experiencias pasadas. En la Figura 2.6 se presenta un ejemplo de como luce de forma general la recuperación de un caso en la implementación del equipo Humboldt.

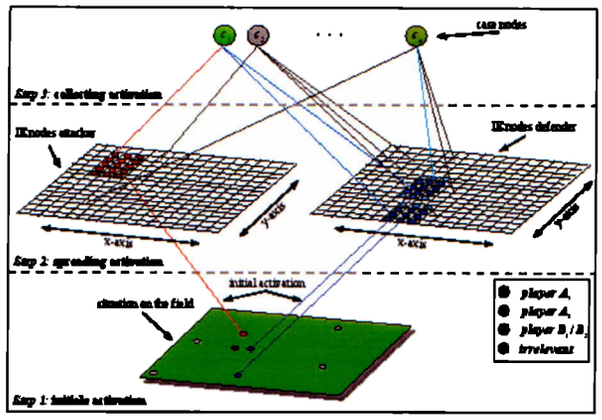


Figura 2.6: Ejemplo de la recuperación de un caso en *Humboldt*

2.4.2. MÁQUINAS DE APRENDIZAJE - SELECCIÓN DE PLANES

Es un subcampo de la inteligencia artificial que se ocupa del diseño y desarrollo de algoritmos y técnicas, que permiten a las computadoras 'aprender'. Según [Alpaydin, 2004] las máquinas de aprendizaje permiten por medio de la programación en computadora, optimizar el criterio de ejecución usando una porción de información o experiencias previas.

A pesar que las maquinas de aprendizaje son usadas ampliamente en problemas relacionados a bases de datos como *minería de datos*, también ha sido de mucha utilidad en la búsqueda de soluciones en problemas de visión, reconocimiento de voz y la robótica.

Aplicación en Fútbol Robótico

La aplicación de estrategias de juego es un verdadero reto, especialmente en ambientes no predecibles como en el fútbol robótico. Una jugada es un plan de equipo que combina comportamientos y capacidades ante un oponente o como mecanismo de ataque o defensa. En [Bowling et al., 2004, Browning et al., 2005] se propone el uso de *libro de jugadas* en voz inglesa *playbook*, éste es un método que combina múltiples jugadas, base para el proceso de adaptación del juego. A pesar de la impredecibilidad, los equipos de fútbol robótico desempeñan una serie de acciones sencillas, estáticas y monolíticas, si éstas no son combinadas adecuadamente no se logrará una estrategia consistente contra los oponentes.

Una solución para construir una estrategia de juego consistente, es usar modelos de acción definidos antes o durante la competencia. Un modelo puede ser usado en un comportamiento reactivo que haga que una estrategia estática (definida previamente), sea ejecutada de forma deliberada en dependencia del uso de un planeador.

El objetivo de hacer uso de modelos de acción previamente definidos, es reemplazar un equipo de estrategias monolíticas por uno de planes de equipo con múltiples jugadas. Un *playbook* encapsula la jugada que el equipo puede usar, además permite la evaluación de la misma, para determinar las mejores jugadas y descartar las débiles.

Una perspectiva de estrategia determina que un robot puede desempeñar un rango de habilidades individuales que son encapsuladas como tácticas y configuradas para proveer un rango de comportamiento. Las tácticas pueden ser clasificadas en activas (manipular el balón de alguna manera) y no activas (defensa, o hacer pases). Cada una de estas tácticas debe ser informada a la capa de control del robot, para mantener información específica sobre la navegación y movimientos de este.

Dos aspectos esenciales en el sistema de estrategia del *playbook* es la selección y adaptación de jugadas de acuerdo a la experiencia obtenida.

a) Selección del *playbook*: el esquema de selección de *playbook* usa las condiciones de aplicabilidad para cada jugada y así, forma una lista de jugadas candidatas entre las que se selecciona una de forma aleatoria. La probabilidad de seleccionar alguna se hace por medio de un sistema de ponderación; ver Figura 2.7, aplicación de estrategia con un *playbook* presentado en [Browning et al., 2005]

b) Adaptación del *playbook*: para facilitar la compilación de resultados pasados dentro del proceso de selección se le asocia un peso a cada jugada.

$$w_p i \epsilon [0, \infty) \tag{2.2}$$

A: set de jugadas aplicables, los pesos son normalizados para definir una probabilidad de distribución masiva.

$$Pr(selectingpi) = \frac{w_p i}{\sum_{pj} \epsilon_A pj} \tag{2.3}$$

Se aplica un regla de adaptación que determina, de un vector de pesos, la siguiente jugada. Esta regla de adaptación multiplica el peso de la jugada anterior por una constante.

$$W(w, p_i, 0) = C_0 w p_i \tag{2.4}$$

2.4.3. LÓGICA DIFUSA

Los sistemas difusos se han mostrado y probado por diferentes autores como aproximadores universales, en [Ross, 2004] se menciona que parte de estas pruebas se basan en la correspondencia

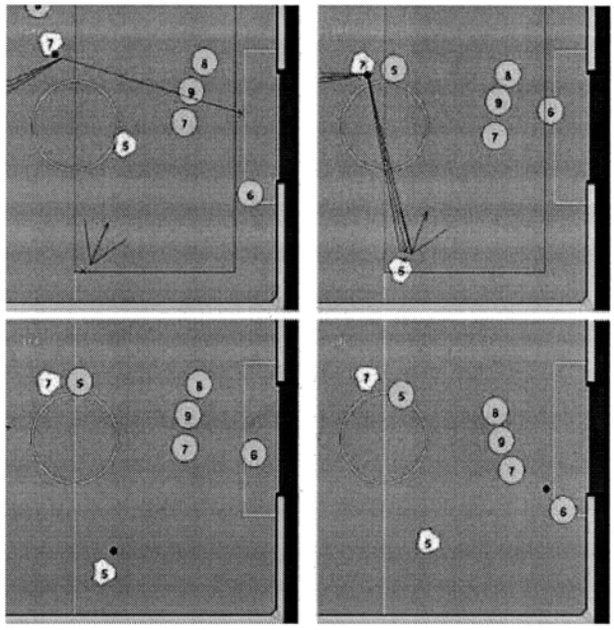


Figura 2.7: Categoría *Small Size* CMDragons vs ToinAlbatross; aplicación de *playbook*

biunívoca entre dos estructuras algebraicas: álgebra abstracta (transacciones con grupos, campos y anillos) y álgebra lineal (transacciones con espacios vectoriales, estados vectoriales y matrices). La relación entre el álgebra y los sistemas difusos se centra en la correlación entre elementos de dos o mas dominios, de esta forma una función algebraica traza un mapa de una variable de entrada a una variable de salida y un sistema difuso, traza un mapa de un grupo de entrada en un grupo de salida.

Se debe destacar que una de las ventajas primordiales de la teoría de los sistemas difusos, radica en que ésta brinda un acercamiento o resolución a un problema dentro de un sistema, donde las funciones analíticas o las relaciones numéricas, no siempre están disponibles o no son tan exactos; de esta forma los sistemas difusos tienen un alto potencial de ser más útiles en sistemas donde son carentes de formulaciones analíticas: los sistemas complejos. Los sistemas complejos son los nuevos sistemas que no han sido probados, que se involucran con funciones humanas tales como sistemas médicos o biológicos, o que pueden ser de carácter social, económico o político.

Sin embargo, es siempre deseable un apoyo adicional, un valor que podamos medir y explotar, por tanto es necesario algunas herramientas adicionales que nos apoyen en los modelos poco profundos, normalmente usados en el razonamiento deductivo (de lo específico a lo general en sistemas) hasta llegar al razonamiento inductivo (de lo general a lo específico), un ejemplo de esto se aplica a los sistemas que se involucran con algún tipo de competencia, juegos.

Como se mencionó en la sección 2.1 en el juego de *tres en raya* solo hay unas cuantas movidas posible en todo el juego, se puede deducir la siguiente movida tomando en consideración una

movida anterior y del conocimiento del juego en sí; este es un ejemplo de razonamiento inductivo, donde poseemos un conocimiento profundo y sustancial.

En [Kecman, 2001] se hace énfasis en el hecho que los sistemas difusos son útiles cuando se posee un conocimiento humano relacionado a la solución de algún problema, en sí, es considerado una herramienta eficiente para estructurar y fijar conocimiento humano en algoritmos útiles, precisos y de significado. Tal como es el razonamiento humano, la verdad de cualquier declaración, medida, u observación es un aspecto de *grado*. Este grado esta expresado por una función de membresía que cuantifica (mide) un grado de pertenencia de alguna entrada sobre algún conjunto difuso. El campo de la lógica difusa es amplio y cubre diversos conceptos matemáticos y lógicos que pueden ser aplicados en muchos entornos.

Aplicación en Fútbol Robótico

La lógica difusa también es usada en el fútbol robótico especialmente en la planificación de caminos [Meng et al., 2000], como apoyo en la definición de estrategia de juego. Un robot debería conocer su posición entorno al área de juego, de esta forma el sistema podría determinar su rol en la siguiente jugada acorde a una estrategia, sin embargo, ésta posición es variable a medida que avanza el juego. Por lo anterior, es necesario partir el área de juego en zonas especiales que permita localizar al jugador, robot, en el terreno de juego; una solución que se plantea es hacer uso de la lógica difusa que apoye en la localización de los agentes, y de esta manera poder tomar decisiones sobre la actuación del robot de acuerdo a su localización.

En principio el área de juego será dividido en zonas, acorde a la información recibida y a la estrategia de juego. El área central será dividida en un plano cartesiano `X` y `Y` y para esto se usarán dos *fuzziers*, uno para para el eje `X` y otro para el eje `Y`. El *fuzzier* `X` tomará un valor universo del discurso por ejemplo `X[-76-,5 7, -38, -19, 0, 19, 38, 57, 76]` y de la misma forma el *fuzzier* `Y`, tomará `Y[-137-. 96,-64, -32, 0, 32, 64, 96, 137]`.

Después de definirse la formación de los dos *fuzziers*, se definirán un colección de reglas difusas para el sistema de planeación de caminos, así, una sencilla regla puede considerar factores como posición del balón y de robots, y una compleja regla difusa, además de incluir estos factores, puede incorporar velocidad y orientación de los robots con respecto al balón. Relacionada a una estrategia de juego, dos listas ordenadas de reglas difusas podrán ser definidas para un robot, una para atacar y otra para defender, en la Figura 2.8 se muestra el ejemplo del *fuzzier X*.

Por ejemplo en el caso de una estrategia ofensiva, la reglas difusas diseñarán el plan de rutas para el robot que domina la jugada, y posteriormente, el plan para los otros robots involucrados en la estrategia, estas reglas consideran la trayectoria a tomar así como la evasión de obstáculos.

Así mismo, en una aplicación para fútbol robótico de simulación [Meng et al., 2000], se hizo uso

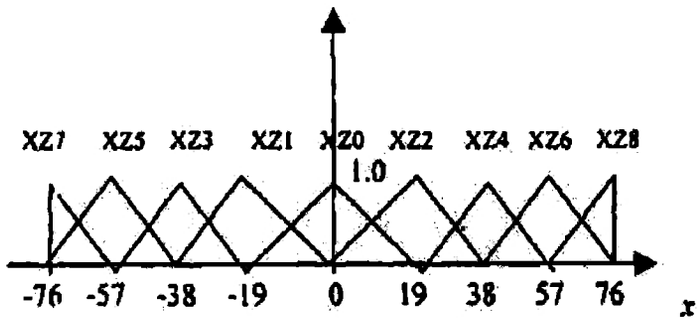


Figura 2.8: Aplicación de *fuzzierX*

de la lógica difusa para adoptar un control inteligente de planeación y cooperación entre los agentes, basado en capas, con el objetivo de lograr una aproximación del conocimiento humano basado en un modelo controlador.

De esta forma el modelo posee tres niveles: la capa inferior regula las acciones primitivas, la capa intermedia controla el comportamiento de un alto nivel y la capa superior toma decisiones sobre un plan de juego estratégico y táctico. En [Meng et al., 2000] la lógica y expresiones difusas fueron implementadas para manejar la transferencia de información entre capas y administrar los conflictos. Por ejemplo la capa intermedia controla un comportamiento de movidas tácticas de acuerdo a reglas de lógica difusa, las movidas tácticas consisten en una secuencia de comandos; el conocimiento de las reglas tácticas esta dado por:

$$R_i^{sj} : ifCONDICIONisC_ithendoM_i, i = 1...n \tag{2.5}$$

donde `CONDICION` es un vector de control y la situación del juego en un momento dado, así la aplicación de reglas se realiza de forma secuencial en dependencia de la situación especificada en el vector de `CONDICION`.

2.5. METODOLOGÍA PARA LA TOMA DE DECISIONES

En las secciones anteriores se ha hecho referencia a algunas de las metodologías usadas para la resolución de problemas en la toma de decisiones, enfocado a la plataforma de competencia de fútbol robotico. Sin embargo, es importante destacar que existen más tecnologías que se han implementado y han resuelto de forma parcial o total algún aspecto de dicha problematica. En el Cuadro 2.4, detallado en [Watson, 1997], se realiza una breve enumeración de tecnologías y así mismo una comparación simplificada sobre cuándo usar y cuándo no usar algunas de estas técnicas computacionales.

Lo anterior se complementa con la mención de algunas características, tomadas de [Pal and Shiu, 2004], que fueron de mucha utilidad en la selección de RBC como metodología para la resolución de problemas de toma de decisiones en esta investigación.

1) *Dominio Incompleto*: el dominio es imposible de comprender completamente o si los factores

Tipo de Tecnología	Cuándo usar	Cuándo no usar
Base de Datos	Datos bien estructurados y estandarizados; aplicación de consultas simples	Datos complejos y con poca estructura; consultas difusas son requeridas
Recuperación de Información	Altos volúmenes de datos de tipo texto	Tipos de datos complejos y no textuales; disponible base de conocimiento
Estadística	Altos volúmenes de datos con alta comprensión e hipótesis bien formulada	Análisis exploratorios de datos con dependencia en muchas variables
Sistemas Basado en Reglas	Problema bien conocido, estable y justificable por medio de un seguimiento de reglas aceptable	Poca comprensión del problema y cambios constantes
Máquinas de aprendizaje	Son requeridas reglas que pueden ser generalizadas de un conjunto de entrenamiento y justificable por medio seguimiento de reglas aceptable	Reglas no son requeridas y la justificación por medio del seguimiento de reglas no es aceptable
Redes Neuronales	Datos numéricos dispersos provenientes de reconocimiento de patrones o procesamiento de señales	Datos simbólicos complejos o cuando se requiere justificación
Razonamiento Basado en Casos	Poca comprensión del ambiente con datos estructurados complejos que cambia conforme al tiempo y la justificación es requerida	Base de casos no está disponible, o se requiere adaptación compleja o si una respuesta óptima es requerida

Cuadro 2.4: Comparación de tecnologías

que determinan el éxito o fracaso de una solución no pueden ser modelados explicitamente. *RBC* permite trabajar con base en la experiencia pasada, no es necesario una comprensión detallada.

2) *Excepciones o Casos Nuevos*: en una situación donde nuevas experiencias y excepciones son encontradas con frecuencia, es difícil mantener la consistencia por medio de un sistema basado en reglas. *RBC* es una alternativa a este tipo de sistemas, gracias a un aprendizaje de casos de forma incremental.

3) *Ocurrencia de Casos*: cuando las experiencias no son lo suficientemente similares para ser comparadas y adaptadas de forma exitosa, es mejor construir un modelo del dominio que se derive de la solución.

4) *Adaptación de Soluciones Pasadas*: *RBC* considera si es beneficioso, en cuanto a costo computacional - tiempo de desarrollo y procesamiento, ya sea crear una solución por medio de modificar un caso similar o crear una nueva solución cuando no aplique ningún caso.

5) *Relevancia de Casos Previos*: es posible obtener datos que almacenen las características necesarias de casos pasados y que puedan sea adaptados a nuevos casos.

Finalmente el razonamiento basado en casos, ha emergido como una metodología que puede llegar aplicar diversas tecnicas en algunas de las fases de su proceso, y tal como se muestra en el Cuadro 2.4 es útil para dar respuesta a situaciones en las cuales el ambiente es complejo o su comprensión requiere de un procesamiento de datos muy estructurados. Es importante destacar que el uso del razonamiento basado en casos aplicado en la definición o selección de estrategias, es un campo poco explorado, por tanto, es uno de los alcances de esta investigación explotar las bondades de esta metodología para alcanzar los objetivos de diseño.

2.6. RESUMEN DEL CAPÍTULO

La interación entre agentes y personas es un aspecto actual y no del futuro, cada día se generan nuevas tecnologías y metodologías que se proponen "humanizar" a este tipo de sistema especial, en este aspecto son variadas las plataformas que se especializan en los elementos "fisicos" de los agentes y otros en los aspectos del "comportamiento"; especialmente los que se relacionan con la toma de decisiones. Ya sea que el razonamiento se genere por medio de algoritmos genéticos, de aprendizaje por refuerzo, métodos estadísticos entre otros, al final el objetivo es que el agente adquiera capacidades similares al ser humando, como lo es, el grado de autonomía, movilidad, iniciativa, sociabilidad entre otros. Este último elemento cobra gran importancia cuando se requiere de sistemas capaces de cooperar y coordinar acciones para lograr un fín en común, como en el fútbol robótico.

En este capítulo se colocó en perspectiva la información relacionada a los sistemas multiagentes y algunas de las tecnologías que se han desarrollado para dar respuesta a su praxis en los diferentes entornos de aplicación. Así mismo, se fundamenta las razones que determinaron la adopción del razonamiento basado en casos como metodología para para la resolución de problemas en este trabajo de investigación.

3. HERRAMIENTAS UTILIZADAS POR EL EQUIPO TECRAMS

"Si el arte tradujera nuestros sueños las computadoras los realizarían"

Harold Abelson

En este capítulo se abordará el trabajo realizado hasta la fecha en el proyecto de investigación de robots cuadrúpedos del ITESM-CEM tanto a nivel de simulación como en la plataforma de robots reales. Se describirán las principales herramientas utilizadas en el proyecto, tanto las implementadas y adaptadas de otras fuentes de colaboración, como lo es el *GT2004* y las desarrolladas de forma local como lo es el *XML Behaviour Control*. A pesar que el trabajo sobre robots tanto en plataforma de simulación como en el de robots reales se divide principalmente en tres áreas estrechamente relacionadas, a saber, *i. Locomoción*, *ii. Visión* y *iii. Decisión*, es objetivo del presente documento enfocar el estudio hacia el área de toma de decisiones, por lo tanto no se realizará un detalle exhaustivo sobre los otros dos aspectos.

Es importante destacar que el objetivo de este capítulo es establecer las bases sobre las que se desarrolla la arquitectura propuesta en el siguiente capítulo, haciendo énfasis en las limitaciones y oportunidades que se han abstraído de las herramientas desarrolladas y utilizadas hasta la fecha, como lo es contar con una herramienta de simulación que permita el desarrollo, prueba y evaluación de una arquitectura para la definición y selección de estrategias, y de esta forma, implementar la coordinación entre múltiples agentes para alcanzar un objetivo en común, aspecto no logrado de forma integral hasta el momento.

3.1. HERRAMIENTAS UTILIZADAS EN TECRAMS

TecRams es el nombre del equipo representativo que ha liderado el proyecto de investigación de robot cuadrúpedos del ITESM-CEM desde su fundación en el 2002 a la fecha se han integrado y desarrollado diversas herramientas para la generación de comportamientos en agentes autónomos, como se analizó anteriormente, se hace uso principalmente del robot *Sony AIBO ERS-7* para la aplicación de estas herramientas. Entre los apoyos que se han utilizado en el equipo *TecRams* se cuentan: *GT2004/XABSL*, *XML Behaviour Control*, *ABG* y recientemente *BADRL*. A continuación se analizará las principales características de estas herramientas.

3.1.1. GT2004/XABSL

El *GermanTeam* es el equipo alemán de fútbol robótico que participa en la liga de cuadrúpedos de la competencia internacional de *RoboCup*, es un trabajo conjunto de varias universidades alemanas entre estas: Humboldt-Universität Berlín, Universität Bremen y la Technische Universität Darmstadt tal como se detalle en[GermanTeam, 2008]. Desde su primera participación en *RoboCup* 2001, se ha constituido como uno de los equipos de mayor experiencia y ha obtenido méritos tanto en competencia como en las publicaciones presentadas en dicho evento. Muchos equipos de la liga, entre estos el *TecRams*, han usado el código fuente y documentación del *GermanTeam*, disponible en el sitio web del equipo, como parte del proceso de colaboración e intercambio de ideas. En la Figura 3.1 se presenta las tareas modeladas por el *GT2004* para el fútbol robótico.

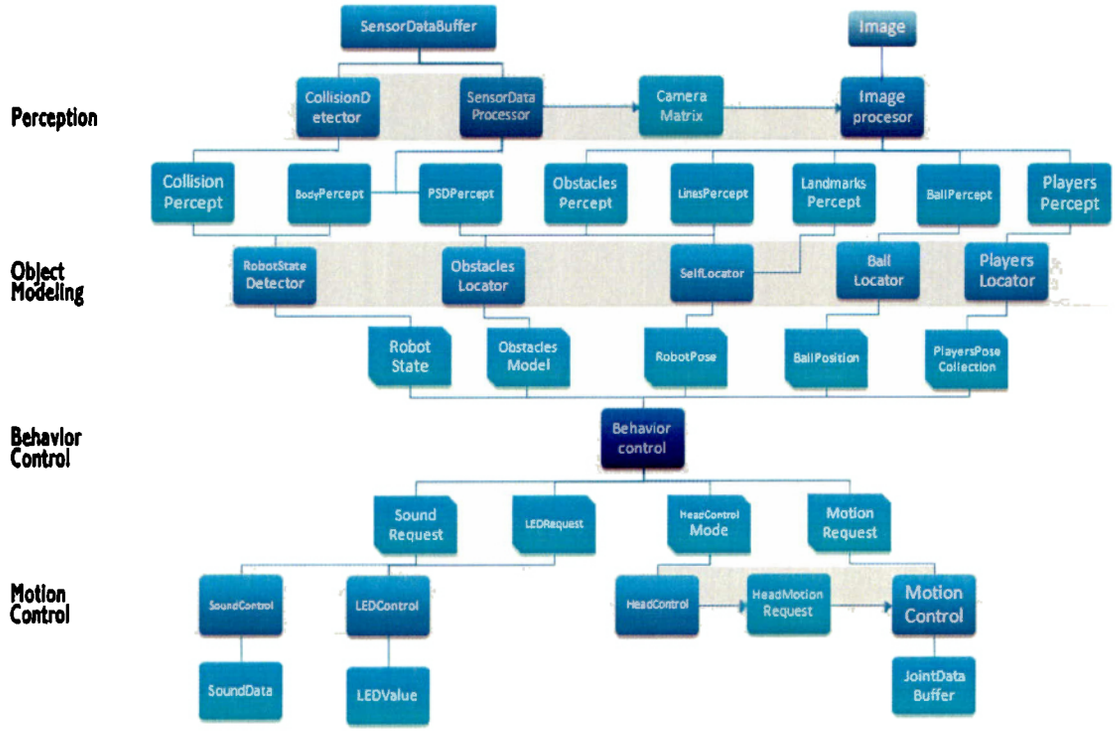


Figura 3.1: Tareas identificadas por *GT2004* para el fútbol robótico

El *GT2004* es una arquitectura de control de comportamiento jerárquica, que consiste en máquinas de estado que pasan de un comportamiento a otro en dependencia del último estado y la situación actual. Está desarrollado en *C++* como lenguaje nativo y *XABSL* (*Extensible Agent Behavior Specification Language*) un lenguaje de comportamiento basado en *XML* con un lenguaje específico que es interpretado y transformado por esta herramienta, que arroja como resultado, archivos en formato C++, el cual permite complementar especificaciones del comportamiento de los agentes autónomos[Lötzsch et al., 2004].

El XABSL fue desarrollado para diseñar comportamientos para agentes que juegan fútbol, sin embargo se pueden generar comportamientos para cualquier tipo de ambiente. Así mismo la herramienta incluye el *Robot Control* útil para el diseño de una jerarquía de opciones y la implementación de sus máquinas internas de estado, ambas soportadas por *XABSL* usando las ventajas de la tecnología de *XML*.

La arquitectura está diseñada para proveer al desarrollador la posibilidad de decidir ya sea expresar condiciones complejas en *XABSL* al combinar diferentes *símbolos* de entrada como operaciones booleanas y decimales o implementar condiciones por medio de un analizador de funciones en *C++* y hacer referencia a estas por medio de una sencillo símbolo de entrada. Los comportamientos básicos están escritos en *C++*, ver Figura 3.2 así como la definición de símbolos; los prototipos y parámetros han sido definidos en un documento *XABSL*, así que cada estado puede aplicarse a estos comportamientos, ver Figura 3.3.

```
#include "CommonBasicBehaviors.h"
void CommonBasicBehaviors::registerBasicBehaviors(Xabsl2Engine& engine)
{
  engine.registerBasicBehavior(basicBehaviorSpecialAction);
  engine.registerBasicBehavior(basicBehaviorStand);
  engine.registerBasicBehavior(basicBehaviorWalk);
}
void BasicBehaviorWalk::execute()
{
  motionRequest.motionType = MotionRequest::walk;
  motionRequest.walkRequest.walkType   = static_cast<WalkRequest::
  WalkType>(static_cast<int>(type));
  if(motionRequest.walkRequest.walkType >= WalkRequest::
    numOfWalkType || motionRequest.walkRequest.walkType < 0)
    motionRequest.walkRequest.walkType = WalkRequest::normal;
  motionRequest.walkRequest.walkParams.translation.x = speedX;
  motionRequest.walkRequest.walkParams.translation.y = speedY;
  motionRequest.walkRequest.walkParams.rotation = fromDegrees(rotationSpeed);
}
```

Figura 3.2: Definición de símbolos en *C++* `walk`

La especificación de un comportamiento en *XABSL* puede ser distribuido en varios archivos. El *GT2004* usa diferentes archivos *XML* para la definición de símbolos, definiciones de comportamientos básicos, condiciones predefinidas, agentes y opciones; ver Figura 3.4.

```
<basic-behavior name="go-to-ball" description="A behavior that walks to the ball">
        <parameter name="go-to-ball.distance"
                   measure="mm" range="0 ..."
                   description="The distance to the ball at the end."/>
        <parameter name="go-to-ball.max-speed.x"
                   measure="mm/s" range="0 ..."
                   description="The max speed in x direction."/>
        <parameter name="go-to-ball.max-speed.y"
                   measure="mm/s" range="0 ..."
                   description="The max speed in y direction."/>
</basic-behavior>
```

Figura 3.3: Implementación comportamiento básico en *XABSL* `go-to-ball`

```
<state name="ball-just-seen-not-moving">
        <subsequent-basic-behavior ref="goalie-position">
        <decision-tree>
                <if>
                        <condition description="ball was seen in the last 3500 msecs">
                                <less-than>
                                        <decimal-input-symbol-ref
                                        ref="ball.time-since-last-seen"/>
                                        <decimal-value value="3500"/>
                                </less-than>
                        </condition>
                        <transition-to-state
                                        ref="ball-just-seen"/>
                </if>
                <else>
                        <transition-to-state
                                        ref="ball-not-seen"/>
                </else>
        </decision-tree>
</state>
```

Figura 3.4: Definición de comportamiento en *XABSL* `golie-position`

Se desarrollaron herramientas para la generación de tres tipos de documentos *XABSL* [Lötzsch et al., 2004]:

1) Código intermediario, el cual es ejecutado por *XABSLEngine* para brindar portabilidad a las herramientas de análisis sintácticos de *XML*.

2) Símbolos de depuración, que contienen los nombres de todas las opciones, estado, comportamientos básicos y símbolos que hace posible implementar la plataforma por medio del análisis de opciones y estados tanto de entrada como de salida.

3) Una extensa auto-generada documentación en *HTML*, conteniendo información de los agentes, opciones y estado, con el fin de apoyar a los desarrolladores a comprender lo que los comportamientos están realizando.

La razón principal que fundamenta el uso de *GT2004* por el equipo *TecRams* es por la facilidad de integración de los módulos de visión, locomoción y decisión, esencial para la operación directa con los robot *Sony AIBO ERS-7*. Se debe destacar que el código original ha sido modificado en gran

manera y se ha desarrollado herramientas alternas para complementar y mejorar el desempeño de la herramienta, para mayor información visitar la página principal del *TecRams* [TecRams, 2008].

3.1.2. XML BEHAVIOUR CONTROL

El *XML Behaviour Control* es una plataforma de definición de comportamientos de robots, por medio de autómatas finitos (AF) descritos mediante *XML (Extensive Markup Languaje)*, en el cual se puede implementar tareas complejas sin conocer a detalle el entorno de programación del robot, tal como se refiere en [Vega, 2005].

El *XML Behaviour Control* ha sido desarrollado en el proyecto de investigación de robots cuadrúpedos del ITESM-CEM, con el fin de proporcionar una herramienta gráfica para la comprensión del sistema de decisión de los agentes de fútbol robótico.

Inspirado en el entorno de competencia de la liga de robots cuadrúpedos de *RoboCup* con robots tipo *Sony AIBO ERS-7*, es una utilidad de manejo intuitivo, lo cual permite un aprendizaje acelerado y simple, ver Figura 3.5, entorno principal de la herramienta *XML Behaviour Control.*

Figura 3.5: Entorno principal del simulador *XML Behaviour Control*

La plataforma de definición de comportamientos está separada de la plataforma de programación nativa de los robots, lo que permite generar y probar comportamientos de juego de forma rápida y eficiente, lo que acelera el desarrollo de nuevas acciones o eventos al realizar actualizaciones en

los archivos correspondientes. De la misma forma el modelado de comportamientos por medio de máquinas de estado jerárquicas, facilita la mejora continua y la reutilización de comportamientos, con la posibilidad de ir encapsulando comportamientos para que las máquinas de estado se mantengan en un nivel de baja complejidad.

Otra de las funcionalidades que posee la herramienta es la creación de un programa gráfico para mostrar eventos, estados activos, inactivos y transiciones entre ellos, lo cual permite brindar un panorama más claro sobre el funcionamiento de las máquinas de estado y de esta manera facilitar la evolución del comportamiento de los jugadores tal como se describe en [Vega et al., 2006]. En la Figura 3.6 se muestra la interfaz de manejador de comportamientos.

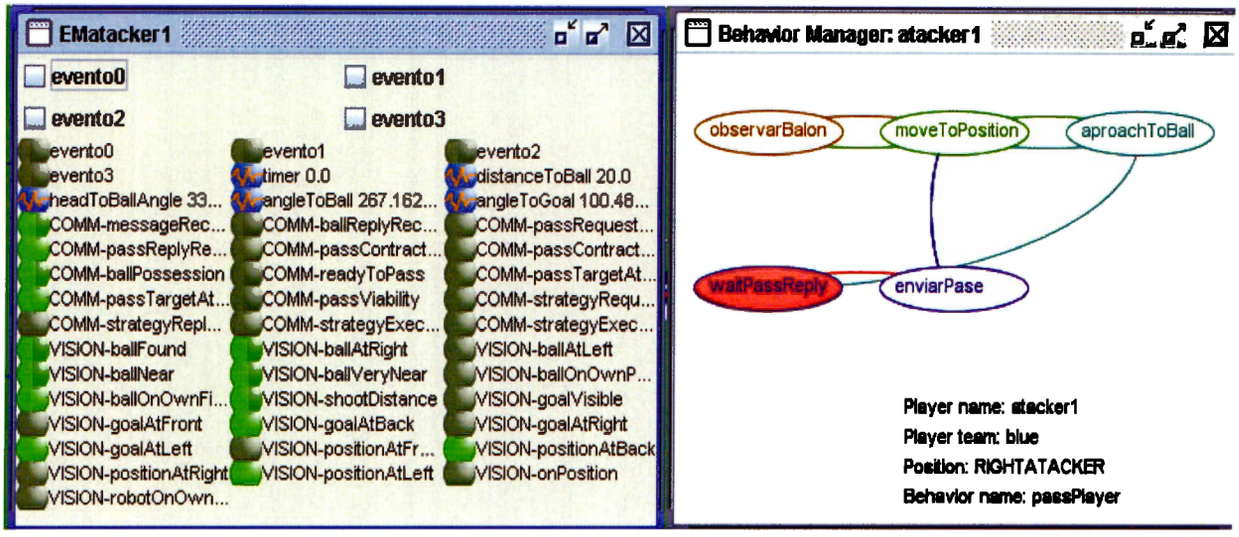


Figura 3.6: Manejador de comportamiento jugador:*atacker1* estado activo: *waitPassReply*

La herramienta de simulación se programó en lenguaje *Java* y esta organizado en varios paquetes como `errorHanding`, `world`, `xbdEngine` y `xbdParse`, en los cuales se define los objetos del mundo virtual, ejecución de la máquina de estados y traducción al documento *XML*, en la Fig: 3.7 se muestra parte de la clase que tiene la función de implementar la comunicación entre los robots.

La definición del ambiente del simulador y los comportamientos de cada uno de los jugadores se realizan en archivos *XML*, un lenguaje basado en etiquetas similar al *HTML*, con la salvedad que este, *XML*, es mas genérico y tiene como objetivo principal describir, crear, estructurar y guardar datos. Los elementos se estructuran con etiquetas que marcan el inicio `<etiqueta>` y el fin `< /etiqueta>` de un elemento.

El fragmento de *XML*, `behavior` es un elemento que poseé otro de tipo `state`, el cual a su vez incluye a otro de tipo `action`. Los elementos además de incluir datos pueden tener atributos, por ejemplo `< state name= "waitPassReply">`, tal como se muestra en Figura 3.8.

```
public AiboRobotCommunication(
              String robotId, String team,
              AiboRobot robot,
              AiboRobotLocomotion locomotion, AiboRobotVision vision)
        {
                this.communicationId= robotId;
                this.team = team;
                this.events = new RobotBooleanEvent[eventStrings.length];
                this.receivedMessages = new Vector();
                this.locomotion = locomotion;
                this.vision = vision;
                this.robot = robot;
        }
```

Figura 3.7: Clase para manejar la comunicación con el *Sony AIBO*

```
<behavior name="init">
        <state name="enviarPase">
                <complexAction name="turnToPass"></complexAction>
                <transition state="moveToPosition">
                        <action id="COMM-turnOffPassRequestReceived"/>
                        <action id="COMM-yieldBallPossession"/>
                        <action id="MOTION-shootFront"/>
                        <event name="COMM-readyToPass"/>
                </transition>
                <transition state="moveToPosition">
                        <action id="COMM-turnOffPassRequestReceived"/>
                        <action id="COMM-yieldBallPossession"/>
                        <event name="VISION-ballFound" operator="not"/>
                </transition>
        </state>
</behavior>
```

Figura 3.8: Definición de un comportamiento en *XML* `enviarPase`

Con respecto a la definición del entorno de juego, tipo de cancha, número de jugadores, posición inicial, definición del equipo y rol a ejecutar, se detalla en el archivo *environment.xml* localizado en la directorio principal del programa, tal como se muestra en la Figura 3.9, en la cual se define el entorno de juego mediante `environment` y cada uno de los agentes, en total ocho, con sus respectivas características con la instrucción `robot type= aibo`.

3.1.3. ABG

Descripción

La herramienta *ABG (Automatic Behavior Generation)*, Generación de Comportamiento Automático; es una herramienta desarrollada en el proyecto de robots cuadrúpedos del ITESM-CEM que hace uso de la Programación Evolutiva para generar comportamientos por medio de Máquinas de Estado Finito en voz inglesa *Finite State Machines*, que pueden ser aplicados a robots que parti-

```
<environment virtual_space="PlayGround2005"
             robot_delay="40" object_delay="150" friction="120">
<robot type="aibo" file="passPlayerViability.xml"
             team="RED" name="rightAtackerRed"
             initial_positionX="300" initial_positionY="100"
             position="RIGHTATACKER"/>
<robot type="aibo" file="passPlayer.xml"
             team="BLUE" name="passDefenderBlue"
             initial_positionX="-200" initial_positionY="-300"
             position="DINAMICDEFENDER"/>
</environment>}
```

Figura 3.9: Definición del entorno de juego y los jugadores

cipan en la categoría de cuadrúpedos de *RoboCup* [Tapia et al., 2006]. Esta herramienta se ejecuta sobre el simulador *XML Behaviour Control*, descrito anteriormente.

La herramienta genera un conjunto de estrategias, tomando como referencia las acciones aplicadas en el simulador *XML Behaviour Control*, para un escenario específico como por ejemplo, el comportamiento de un portero, y aplica funciones de ajuste para producir máquinas de estado optimizadas y especializadas, en dependencia de la situación, para posteriormente aplicarlas sobre el robot real.

El computo evolutivo, aplica algoritmos de selección natural, al codificar cada una de las características de la solución deseada en pequeñas estructuras llamadas *genes* y la suma de todas estos genes es llamado *cromosoma*, cada cromosoma representa un *individuo* y la suma de todos los individuos, de una generación determinada, es llamada *población*. Solo el 'mejor' material genético (programas y fragmentos de programa) de los *individuos* de una población, son transferidos hacia la siguiente generación. Con base en estos principios, la programación que hace uso de algoritmos genéticos, es capaz de desarrollar algoritmos más complejos para dar solución a un problema específico.

Aplicación

El objetivo de *ABG* es hacer uso del *Computo Evolutivo*, para generar comportamientos en un sistema multiagente, aplicado al equipo de robots cuadrúpedos, *TecRams*. En la Figura 3.10, se observa el proceso de generación de comportamientos a partir del simulador *XML Behaviour Control*, hasta la aplicación en los robots reales definido en [Tapia et al., 2006].

Las etapas de *Computo Evolutivo*, para esta aplicación, esta definido por los siguientes pasos:

1) *Definición de restricciones y objetivos del algoritmo*:
   Para evitar la sobrecarga de cromosomas que realizan acciones incoherentes sobre aquellos que realizan acciones útiles. En el caso de *búsqueda de la pelota* el comportamiento restringe las salidas hacia aquellas acciones que involucran movimientos. La herramienta inicia

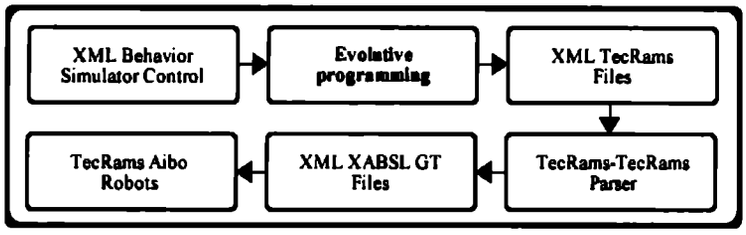


Figura 3.10: Proceso de generación de comportamientos haciendo uso de computo evolutivo

al crear un conjunto de individuos que conformarán la primera generación, estos serán evaluados y cada uno de ellos crearan otro conjunto llamado *descendientes* a partir de genes transformados o *mutados*. Posteriormente ambas generaciones entran en un proceso de *selección* para determinar los mejores atributos que serán transferidos a una tercera generación. El proceso continua hasta alcanzar las condiciones de paro.

2) *Codificación del cromosoma*:
Cada comportamiento esta compuesto por un conjunto de *genes de comportamiento*. Un gen de comportamiento es una lista de estados donde cada estado tiene una lista de acciones, transiciones y acciones complejas. En la Figura 3.11 se muestra el comportamiento de un individuo como un conjunto de máquinas de estado, que posteriormente puede ser usada por otro comportamiento como salida para una acción compleja.

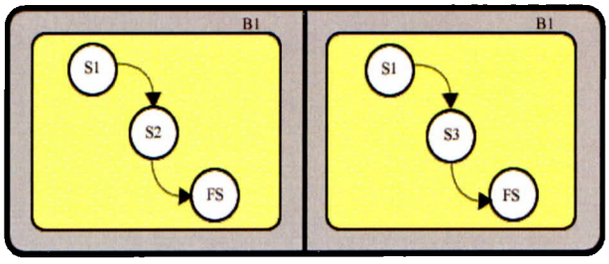


Figura 3.11: Par de cromosoma en una generación específica, cada conjunto de comportamiento esta compuesto por un comportamiento de tres estados

3) *Mutación*:
El proceso de mutación hace uso de la variable *Poisson* con una media de tres, que determina el número de mutaciones que un individuo puede sufrir antes de formar parte de la siguiente generación. Las operaciones de mutación pueden ser: alteración de la salida de un determinado estado, agregar o eliminar de forma aleatoria transiciones, agregar un estado independiente, eliminar un estado y las transiciones a este y cambiar el estado inicial. El Cuadro 3.1 se presenta las operaciones permitidas multiplicadas por un factor que indica el grado de aplicación de la mutación.

4) *Ajuste*:
La función de ajuste evalúa cada *genoma* y retorna un valor que representa que tan bien se ha

Mutación	Probabilidad
Cambiar la salida del estado	1.0
Cambiar la dirección de transición	1.0
Cambiar la condición de activación a una transición	1.0
Agregar un estado al comportamiento	0.5
Cambiar el estado inicial	0.5
Eliminar un estado	0.5

Cuadro 3.1: Probabilidades de Mutación de un Cromosoma

resuelto el escenario. Ya sea al tratar de generar un comportamiento, para un solo agente, en alguna acción específica o un acción compleja. Una forma de evaluar es aplicando un valor binario de 0 ó 1 en aquellos individuos que logran el objetivo; por ejemplo, los individuos que encontraron el balón se les asigna 1 aquellos que no 0.

5) *Población*:
Una forma de definir la población que se introducirá en cada generación es calculando el número posible de individuos que formará parte del *universo*. Si se considera un conjunto de `n` estados, y cada estado tiene `t` transiciones, se puede definir que el número de combinaciones esta dado por la fórmula:

$$CombinationNumer = States * Transitions = (n-1) * (\frac{n*(n+1)}{2}) \quad (3.1)$$

6) *Selección y Elitismo* Es una etapa que implica competencia entre dos individuos a la vez; se aplica el principio de aquellos individuos que estén en mejor 'condición física' serán seleccionados. Existe la posibilidad que se pierdan los mejores individuos de generaciones pasadas debido a la aleatoriedad de la competencia, es por esto que se aplica otro proceso de *elitismo* que permite crear una nueva generación, en las cuales las mejores características de sus padres serán copiadas de forma idéntica en la nueva generación procurando evitar la perdida de información.

Al final, la herramienta genera un reporte que presenta un resumen de los procesos de ajuste que sufrieron las generaciones, cada vez que una generación nueva es creada. Así mismo, el sistema obtiene un archivo de comportamiento escrito en formato *XML* para ser probado en *XML Behaviour Control* y posteriormente ajustarse a la codificación de un robot real.

3.1.4. BADRL

La herramienta *BADRL (Behavior Agent Definition by Reinforcement Learning)*, Definición de Comportamientos de Agente por medio del Aprendizaje por Refuerzo, es un nuevo paquete de código o solución, generado a partir de la herramienta *XML Behaviour Control*, en el cual se aplica el algoritmo de aprendizaje por refuerzo *Q-Learning*, para generar comportamientos en los cuales los agentes emplean un conjunto de acciones dentro de un proceso de aprendizaje para llevar a

cabo objetivos específicos, tal como se detalla en [Reyes, 2008].

Este paquete o solución tiene como entradas, en una clase especial, las acciones, la configuración y las políticas del aprendizaje y tendrá como resultado, un conjunto de acciones aprendidas. En la Figura 3.12 se presenta el esquema general de la solución que se empleará sobre *XML Behaviour Control*.

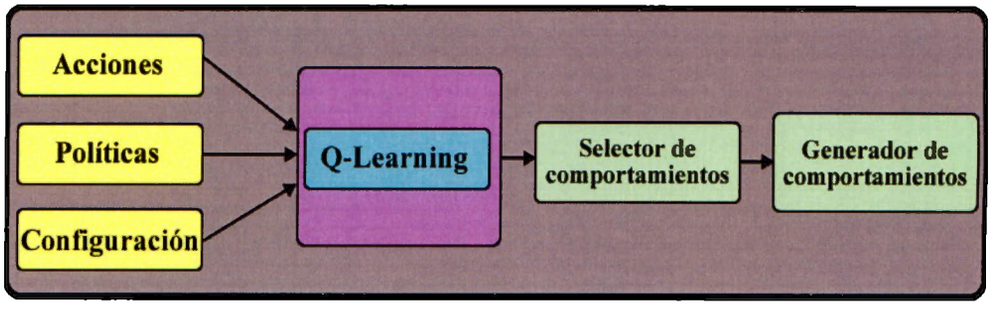


Figura 3.12: Nueva Solución en el *XML Behaviour Control*

La caja que corresponde al aprendizaje *Q-Learning* evaluará el conjunto de entradas, aplicará el proceso de aprendizaje y generará un objeto especial que contenga el conjunto de acciones aprendidas, para ser posteriormente utilizado en algún comportamiento específico, tal como se muestra en la Figura 3.13.

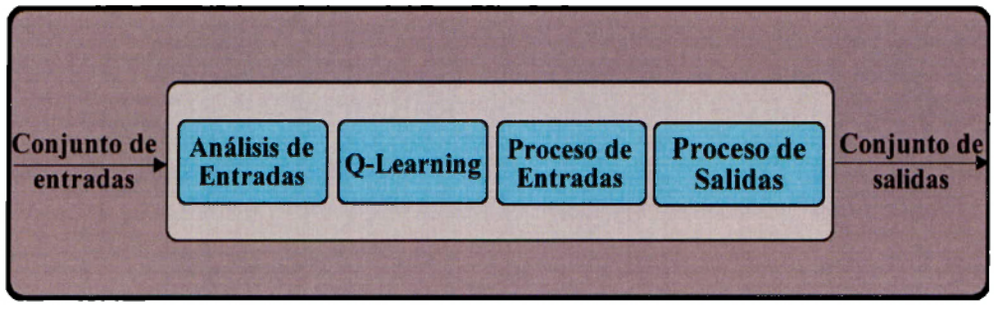


Figura 3.13: Procesos General de *BADRL*

La implementación de la herramienta *BADRL* fue planificada y llevada a cabo con base en el código de desarrollo conocido como *Free Connectionist Q-Learning Java Framework* mediante las siguientes fases:

1) Generación de archivo de configuración con la información sobre los parámetros, que determinan el control del cerebro de la aplicación (*Brain*).

2) Generación de archivo de configuración con las acciones que puede realizar el agente, y éstas, se traducen automáticamente en una clase anónima heredando de la clase *Action*.

3) Generación de comportamientos utilizando la información arrojada por el sistema *Free Connectionist Q-Learning Java Framework*, para cada acción seleccionada.

4) Generación de archivo de configuración con las reglas y la evaluación de cada comportamiento definido anteriormente.

5) Generación del reporte estadístico con la información evaluada por *BADRL*.

Archivo de Configuración

Q-Learning es un algoritmo que utiliza como modelo de aprendizaje por refuerzo una diferencia temporal definido mediante la formula:

$$Q(S_t, a_t) \leftarrow Q(S_t, a_t) + \alpha[r_{t+1}] + \gamma[\max_a [Q(S_{t+1}, a_t)] - Q(S_t, a_t)] \tag{3.2}$$

Donde S_t es el estado en el que se encuentra el agente y realiza la acción a_t. $Q(S_t, a_t)$ es el valor y no recompensa, obtenido de estar en estados S_t y realizar la acción a_t. Los parametros α (alfa) y γ (gamma) determinan el grado de aprendizaje de la fórmula y se encuentran definidos en el archivo de `conf.xml`. En la Figura 3.14 se muestra la información de los parámetros definidos en `conf.xml` para modificar las variables utilizadas por *Q-Learning*.

```
<!-- qLearning parameters -->
     <parameters>
           <alfa-param value="0.5"/>
           <gamma-param value="0.8"/>
           <lambda-param value="1"/>
           <boltzmann-param value="true"/>
           <temperature-param value="0.9"/>
     </parameters>
```

Figura 3.14: Archivo de configuración `conf.xml`

Repositorio de Acciones

Las acciones son generadas automáticamente al leer el archivo `actions.xml` creado por el programador con el fin de crear un repositorio de acciones. Éstas son generadas por medio de clases anónimas para crear un arreglo que es utilizado por el *Free Connectionist Q-Learning Java Framework*. En la Figura 3.15 se representa como BADRL genera un estado con una acción atómica y una transición a un estado principal por cada acción definida en `actions.xml`. Posteriormente se genera el estado principal cuyo objetivo es esperar cambios en el ambiente que activen otros estados y las acciones correspondientes.

Definición de Metas y Objetivos

El conjunto de objetivos se describe en el archivo `rules.xml`, el usuario es capaz de modificar las reglas o las políticas de evaluación que el agente seguirá durante la ejecución del comportamiento específico. Por ejemplo, para el comportamiento de *conducción de la pelota*, se definió como meta que el agente avanzara con la pelota una distancia mayor que la mitad del campo

```
<!-- DEBUG XMLWriting: ACTION STATES -->
<state name='camina_lento' iterate='true'>
      <action id='MOTION-goForward' param1='2'/>
      <transition state='firstState'/>
</state>
```

Figura 3.15: Estado que ejecuta una acción atómica y transición a un estado principal

de juego, esto implica que el agente es capaz de localizar la pelota en el terreno de juego y aproximarse a ésta a una distancia que le permita patearla hacia adelante y de esta forma avanzar hasta alcanzar la meta. En la Figura 3.16 se presenta la definición de una regla para la conducción de la pelota.

```
<ql-rule id="1" name="Principal_Targer" useTimer="18">
<and>
<ql-event name="eventSD" value="VISION-shootDistance" function="isActive()"/>
<ql-event name="eventBF" value="VISION-ballFound" function="isActive()"/>
<ql-event name="agent" value="positionX" function="greaterThan(80)"/>
</and>
<reward goWell="true" goBad="true"/>
<debug text="Target Reached!!" info="positionX" />
</ql-rule>
```

Figura 3.16: Definición de reglas en `rules.xml` para *conducción de la pelota*

Criterio de Evaluación y Retroalimentación

Un criterio de evaluación es diseñado para que de forma obligatoria regrese una retroalimentación para *Connectionist*. Esta retroalimentación es creada por el programador en el archivo `conf.xml` dentro de las etiquetas *recompensas*; en la Figura 3.17 se muestra los criterios de evaluación establecidos en BADRL en el archivo `conf.xml`.

```
<rewards>
      <movingWell-reward value="-0.05" />
      <collides-reward value="-0.2"/>
      <default-reward value="-0.2" />
</rewards>
```

Figura 3.17: Criterios de evaluación en `rules.xml`

Estas recompensas se activan de acuerdo a lo establecido en los criterios de evaluación. En el caso de la conducción de la pelota, si el agente pierde la bola después de un periodo se activa la bandera de *collides-reward* (colisión)y se desactiva la de *movingWell-reward*, con el objetivo que se obtenga una retroalimentación adecuada.

Reporte Estadístico

Se generan archivos en formato *html* que contienen información sobre la evaluación de las acciones del agente respecto al tiempo de ejecución del proceso de aprendizaje de cada uno de los comportamientos, con el objetivo de determinar si el agente ha aprendido a emplear las acciones que recibió como entrada en el sistema y en qué iteración se logró si fue el caso. El reporte principal contiene la siguiente información: número de iteración, arreglo generado de acciones, para crear un comportamiento, recompensa otorgada, promedio de recompensa y un resumen de lo ocurrido.

3.2. RESUMEN DEL CAPÍTULO

A pesar que *XABSL* es una herramienta poderosa para modificar los comportamientos de los agentes, es limitada para generar cursos de acción dinámicos en tiempo real. El proceso de desarrollo de comportamiento es bastante lento y a pesar de las herramientas de ayuda, es difícil localizar errores ocurridos al momento de ejecución incluso en el mismo proceso de compilación. Sin embargo, la facilidad de integrar información relacionada al control de bajo nivel del robot (locomoción y visión) con la administración de comportamientos (decisión), lo convierte en un punto de referencia para futuras investigaciones.

El *XML Behaviour Control* ha sido una herramienta de mucha ayuda para crear, probar y analizar los comportamientos de los agentes de forma gráfica y de fácil comprensión; y evaluar su ejecución antes de ser implementado en el entorno real. Así mismo, el modelado de comportamientos por medio de máquinas de estado jerárquicas, facilita la mejora continua y la reutilización de comportamientos. El módulo de comunicación provee la flexibilidad de incorporar nuevas características y comportamientos más complejos entre varios jugadores. Esta herramienta puede ser utilizada como plataforma para la aplicación de procesos de coordinación entre sistemas multiagentes, como lo es la negociación y cooperación, así mismo aplicar aprendizaje.

La aplicación de *Computo Evolutivo* en la herramienta *ABG* ha permitido transferir, con la aplicación de algunos ajustes, los resultados obtenidos de experimentos realizados en el simulador *XML Behaviour Control* hacia el entorno real *GT2004*. Pero es necesario destacar que este es un proceso gradual y la función de ajuste debe ser capaz de diferenciar entre soluciones: *muy buenas*, *buenas*, *débiles*, *malas* o *muy malas*. Hasta el momento la herramienta ha sido útil en el desarrollo de comportamientos sencillos que generalmente involucran a un sólo agente realizando una actividad, como por ejemplo buscar la pelota (*searchBall*). A pesar de lo anterior, el *Computo Evolutivo* es una alternativa poderosa para la solución de una infinidad de problemas en los cuales deba definirse adecuadamente la restricciones y metas de cada uno de los escenarios, y de cada uno de los agentes involucrados en el sistema.

El *Aprendizaje por Refuerzo* utilizado en BADRL como una herramienta útil para la generación de comportamientos, ha sido probado de forma exitosa, similar a la herramienta *ABG* en la generación de comportamientos sencillos generalmente acciones que aplica un sólo agente; a pesar

que se realizó un aproximación en la que varios jugadores aprendieran un comportamiento colaborador al mismo tiempo, caso *triangulación a disparo a portería*, en el cual se utilizaban dos agentes, *BADRL* fue utilizado dos veces (para este caso), uno por cada agente, lo que provoca que el comportamiento sea colaborador más no en coordinación con otros agentes. Así mismo mismo es necesario implementar mecanismos que permitan asegurar que las acciones aprendidas son correctas, es decir, el agente, aprende a conducir el balón con un número finito de acciones, de forma eficaz, mas no se identifica si las realiza de forma eficiente.

Cuando se controla un equipo de robots es mucho mas complejo diseñar una aplicación que seleccione de un conjunto de acciones reactivas que a la vez coordinen la ejecución de todos los robots para lograr un objetivo en común; el comportamiento reactivo se impondrá, si no hay control sobre ello, sobre cualquier acción coordinada, dado que esta sólo responde a un estímulo inmediato y no realiza un análisis a largo plazo. Esto se complica aún más cuando este equipo debe estar preparado para responder a un entorno dinámico, incierto y competitivo, como lo es el fútbol robótico. Este panorama es el actual, los agentes juegan y realizan las acciones que deben aplicar según el comportamiento definido, si es un portero, un defensa, un delantero, pero no cooperan entre sí para lograr un objetivo.

El *Razonamiento Basado en Casos* se propone en esta investigación como mecanismo de apoyo en la toma de decisiones, al aplicar componentes básicos del sistema, entre estos el motor de búsqueda con base en *similitud*, de experiencias pasadas. Esto permite a los agentes, como equipo, emplear un plan de juego - *estrategia*, al *alinear y corresponder* una situación nueva, en una fase de *juego - en línea*, con una *base de estrategias* previamente aplicadas, en una fase de *práctica - fuera de línea* para identificar el conjunto de casos que se corresponden, luego de la comparación, y seleccionar el más acorde a la circunstancia.

El trabajo realizado hasta la fecha en el proyecto de investigación de robot cuadrúpedos ha sido de mucha importancia en el desarrollo de un equipo cada vez mas competitivo, *TecRams*, de la categoría de cuadrúpedos (ahora llamada categoría estándar), se ha logrado posicionar como uno de los mejores a nivel latinoamericano. El presente trabajo es un aporte de valor que da continuidad al trabajo realizado y a la vez, explora otras metodologías computacionales, para dar solución al problema de toma de decisión al coordinar un equipo de robots.

4. TEAM AGENT BEHAVIOR ARCHITECTURE - TABA

"Si tenemos que aprender a hacer alguna cosa antes de hacerla, podemos aprender mientras la hacemos"

Aristóteles

Los Sistemas de Multiagentes proporcionan una base natural como apoyo en el proceso de toma de decisión, especialmente en dominios complejos y dinámicos. Así mismo, este proceso de decisión, puede ganar una experiencia valiosa por medio de modelos de simulación basados en agentes que representan, con la mayor exactitud posible, el entorno real y sus desafíos. Un sistema de multiagentes que involucra a varios agentes que colaboran para alcanzar un objetivo en común, es visto como un *equipo de agentes*. Una *arquitectura* de control apropiada es crucial para que este *equipo de agentes*, lleven a cabo con éxito sus objetivos de diseño, especialmente en agentes autónomos como los *robots móviles*.

En este capítulo se propone la arquitectura de control para el comportamiento coordinado entre un equipo de agentes, llamada *TABA* (*Team Agent Behavior Architecture*). Entre los aspectos claves de la estructura se presentan: un control híbrido de comportamientos reactivos, planeación y capacidades de razonamiento basado en casos; aplicación de la tecnología *XML* para organización y representación de los casos y entorno de simulación aplicado al fútbol robótico mediante *JAVA*.

4.1. INTRODUCCIÓN A LA ARQUITECTURA

Un agente es esencialmente caracterizado por la manera en la cual es diseñado y por sus acciones, en otras palabras por su *arquitectura* y por su *comportamiento*. En [Ferber, 1999], la arquitectura corresponde al punto de vista del diseñador, quien lleva a cabo la tarea de definir las funciones, acciones de los agentes y la forma en la cual los agentes lograrán sus objetivos de diseño; además, identificar qué tanta percepción y representación del entorno tienen los agentes o simplemente actúan de forma reactiva; cómo se relacionan con otros agentes, esto es, la relación que existe entre el agente, su entorno y otros agentes.

Este trabajo tiene la finalidad de diseñar una arquitectura de estrategias que sea aplicada en ambientes de juego para fútbol robótico y que además puede extenderse a otros entornos de aplicación. Una *estrategia* es un plan de acción en un tiempo definido con el fin de lograr un objetivo general, en el caso del fútbol robótico, las estrategias pueden ser útiles para lograr la anotación de goles como parte de una estrategia ofensiva de un *equipo* de agentes o evitar que los adversarios (*oponentes*) marquen un gol, como ejemplo de una estrategia defensiva.

Así mismo, una estrategia esta compuesta de una serie de eventos que deben realizarse de forma consecutiva, para lograr el fin propuesto, estos eventos deben contemplar en gran medida todos los acontecimientos que afecten de forma interna al *equipo*, como lo es la coordinación de los mismos, y aspectos externos, como lo es el *ambiente* y el proceder de los *oponentes*. En las secciones posteriores se proporcionará una definición formal para los elementos de la arquitectura.

El fútbol robótico es un enorme reto para los sistemas inteligentes debido a que conlleva muchos aspectos relacionados a los sistemas multiagentes, entre estos, la arquitectura, la comunicación, la interacción, la planeación, la detección de conflictos y específicamente, la toma de decisión, que se relaciona directamente con la selección y aplicación de una estrategia, aunado al hecho que se desarrolla en una ambiente dinámico y lleno de incertidumbre, que implica no solo contar con la información del campo del ambiente de juego (porterías, pelota, oponentes, equipo), si no, la definición de roles y formación del equipo al momento de aplicar una estrategia.

El entorno de simulación para la aplicación de la arquitectura de estrategias hace uso de la herramienta *XML Behaviour Control*, analizada en la sección 3.1.2 y desarrollada en *Java 5*, el cual representa un ambiente de juego aplicado al fútbol robótico, sobre la base de las reglas de juego de la categoría estándar-robots cuadrúpedos, promovida por la competencia internacional *RoboCup* visto en detalle en la sección 2.3.2.

La definición del uso de estrategias se desarrolló como parte de un nuevo módulo que incluye definición de estrategia, de patada inicial, defensiva u ofensiva, definiciones de roles, tiempo de ejecución de la estrategia, acontecimientos de cada una de las estrategias, mejoras en la comunicación de los agentes, acciones especiales, proceso de selección de estrategias entre otros.

Como parte de la definición de arquitectura y un elemento innovador de la misma, se presenta la definición, incorporación y aplicación de un sistema simbólico aplicado a la definición de estrategias con orientación al fútbol robótico al cual se le denomino *SSDS Soccer Strategy Description Symbols*, Descripción Simbólica para la Estrategias aplicadas a Fútbol.

4.2. ENTORNO Y REGLAS DE FÚTBOL ROBÓTICO

Es importante destacar algunos aspectos correspondientes a las reglas del juego de fútbol robótico que son importantes para la comprensión de la terminología usada en la arquitectura. Las reglas que se analizarán a continuación corresponden a la versión 2007 para la categoría de cuadrúpedos de *RoboCup*, nuevamente se recalca que el entorno es solo un medio para probar la arquitectura y este puede ser modificado, actualizado o incluso reemplazado por otro entorno.

Debido a que las reglas para la categoría de simulación son diferentes a la de cuadrúpedos, se decidió extender un entorno de simulación basado en las reglas de este último, desarrollado en el proyecto de investigación de robótica móvil del ITESM-CEM. En [AiboSoccer, 2007] se puede descargar el documento completo de las reglas.

Las reglas del juego son, en general, sencillas: dos equipos: Rojo y Azul, cada equipo consta de un portero (*goalKeeper*) y de tres jugadores de campo (*fieldPlayer*); se usan robots de tipo Sony AIBO ERS-7. El objetivo del juego es anotar la mayor cantidad de goles durante el partido.

Un partido consiste de tres partes: la primera mitad, receso de medio tiempo y la segunda mitad, cada una es de 10 minutos. Para efectos de esta simulación el receso de medio tiempo esta omitido, y las pruebas realizadas no contemplan partidos completos, sólo fracciones de 10 minutos. El entorno de juego incluye: cancha de fútbol, pelota, dos porterías, dos postes y dos equipos. En la Figura 4.1 se

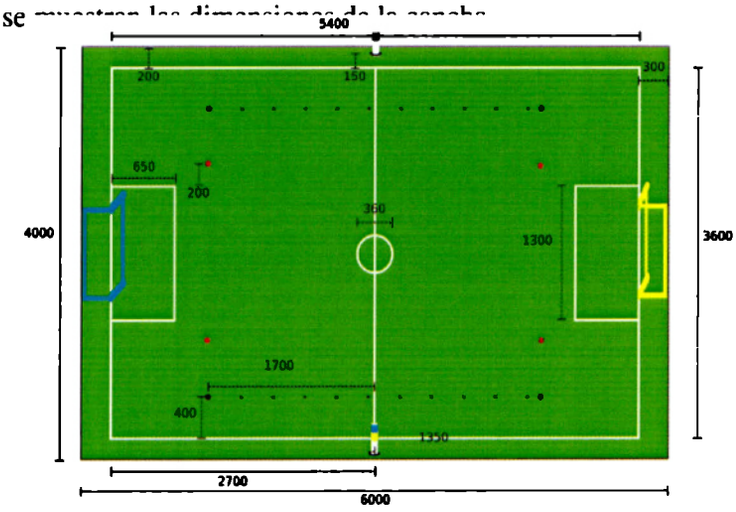


Figura 4.1: Dimensiones de la cancha

En la Figura 4.2 se muestra el entorno general de juego, con base en las reglas aplicadas a cuadrúpedos en RoboCup.

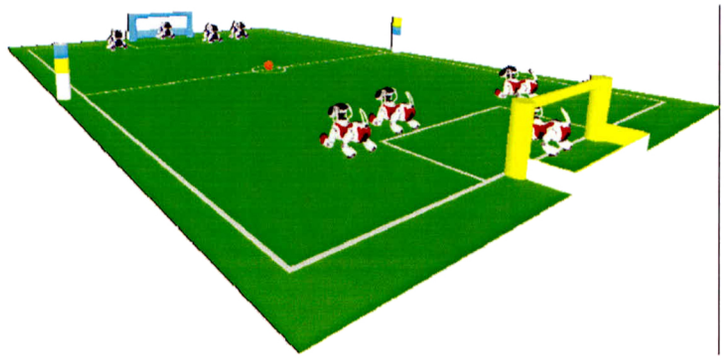


Figura 4.2: Entorno general del juego

Las reglas básicas aplicadas a este entorno de simulación son:

- pelota en el centro: cuando inicia el juego y cuando se ha anotado un gol.
- pelota en saque de banda: cuando la pelota salió por las bandas laterales (paraleas a portería), la posición es en dependencia del equipo que la haya sacado. Se coloca sobre la cancha en una línea imaginaria de puntos de color negro tal como se muestra en la Figura 4.1.
- pelota en tiro de esquina: cuando la pelota salió por las bandas verticales (perpendiculares a portería), la posición es en dependencia del equipo que la haya sacado. Se coloca sobre la cancha en un punto de color rojo tal como se muestra en la Figura 4.1.

Estas reglas son eventos importantes que generar interrupciones o alertas que dan inicio al proceso de selección de estrategias, este aspecto se analizará en secciones posteriores. En el Apéndice B.1 se presentan parte de los métodos implementados.

4.3. ENTORNO DE SIMULACIÓN

El entorno de simulación de *XML Behaviour Control*, fue actualizado y modificado para la aplicación de la arquitectura de estrategias en un ambiente en 2D, en el que se modelan objetos fijos (cancha, porterías, postes, líneas) y objetos móviles (pelota y jugadores).

4.3.1. XML BEHAVIOUR CONTROL Y TABA

Como parte de la aplicación original, existe un menú de opciones en los que destaca el manejador de comportamientos y eventos por cada robot; este muestra una máquina de estados y las

transiciones entre estos, así como los estados activos o en ejecución. Así mismo en otros aspectos de la animación incluyen el movimiento y trayectoria de los objetos (robots y pelota), contador de goles, tiempo de juego transcurrido y capacidad de mover manualmente la pelota por medio del menú de opciones.

Dentro de las modificaciones realizadas para la aplicación de la arquitectura de estrategia se debe mencionar la actualización del entorno a la versión 2007 de las reglas, la incorporación de una consola informativa en la parte inferior útil para identificar los estados principales del juego y la división de la cancha de juego en regiones, como se analizará a continuación.

4.3.2. DIVISIÓN EN CUADRANTES

La división en regiones responde al hecho en el que los jugadores necesitan conocer dónde están localizados en el terreno de juego. A pesar de su posición en el eje de coordenadas X y Y para una ubicación específica, es de mucha utilidad la división de la cancha en regiones que los jugadores puedan emplear para la definición y ejecución de estrategias de juego.

Otro aspecto que motivó la división de la cancha en regiones es el hecho que en el fútbol humano también se hace uso de la división del terreno de juego como una forma intuitiva de identificar áreas de oportunidad o en la misma especificación de estrategias de juego. En [Sng et al., 2002, Buckland, 2004] se muestra algunas aplicaciones relacionadas a la división de la cancha en regiones.

El tamaño de los cuadrantes es variable, esto responde al hecho que por experiencia en competencia de fútbol robótico se ha analizado, en el proyecto de investigación de robots cuadrúpedos, que existen áreas críticas como las cercanas a portería, que por lo general deben ser pequeñas vitales para mejorar el control de la pelota y áreas de transición situadas a lo largo de la cancha, un poco mas grandes, para brindar oportunidad a pases, similar a la forma en la que se compite en un juego de fútbol humano.

De esta forma la cancha es dividida en 35 regiones tal como se muestra en la Figura 4.3. Se aclara que tal como se muestra en la figura, las división por medio de líneas muy delgadas es solo un aspecto visual para ayudar al operador del simulador a identificar la ejecución del programa sin que esto altere la atención principal en los agentes; si no se observa adecuadamente las líneas, se puede verificar mediante la Figura 4.4b de la sección 4.5

4.4. EQUIPO Y JUGADORES

La coordinación es un aspecto crítico en el diseño de equipos de robots que ejecutan diversas tareas para llevar a cabo un objetivo. Un robot por sí solo puede llevar a cabo metas aplicando comportamientos reactivos; pero cuando se trata de coordinar *equipos* de robots, es un poco más

Figura 4.3: Entorno general del simulador con la división en cuadrantes

complejo aplicar comportamientos meramente reactivos, se puede establecer una combinación de principios deliberativos. A este punto es importante definir lo que es *equipo* y un *adversario*.

Equipo es el conjunto de jugadores que colaboran entre sí para lograr una meta en común.

Adversarios es el conjunto de jugadores que compiten contra el *Equipo*.

4.4.1. ESTADOS DEL EQUIPO

Los estados del equipo son las situaciones en las que un equipo se encuentra en un momento dado del juego. Está definido por tres estados básicos: **Patada Inicial**, **Ofensivo** y **Defensivo**. La definición de estos estados es vital para la definición y selección de una estrategia para una situación específica. Como se verá en secciones posteriores, en la definición de estrategia, el estado del equipo es conocido como el atributo `type`, que posteriormente es utilizado en el procesamiento de selección de estrategia.

Patada Inicial

La patada inicial o *KickOff*, es el estado en el cual un equipo da inicio a un partido o cuando un gol es anotado durante el juego. Normalmente los jugadores regresan a la posición inicial y la pelota es posicionada en la parte central de la cancha. Un método analiza de forma constante si ha habido anotación y de esta forma se realiza una solicitud para aplicar una estrategia de *KickOff* ya sea que la patada inicial la ejecute el *Equipo* o los *Adversarios*.

Ofensivo

Un equipo ingresa a un estado ofensivo *OF-Ofensive* por la combinación de dos factores: la posición de la pelota 'cerca' de la portería contraria y el análisis de riesgo de los jugadores contrarios. Con 'cerca' se entiende que la pelota está situada en la otra mitad de la cancha, de esta forma, si nuestro equipo es el Rojo y la pelota se encuentra cerca de la portería Azul, el equipo Rojo entra en un estado potencialmente ofensivo. El factor de análisis de riesgo, consiste en identificar la posición los jugadores contrarios y calcular un valor porcentual de riesgo para la ejecución de una estrategia, es decir, a mayor contrarios cercanos a nuestros jugadores, mayor el riesgo en la ejecución. De esta forma a pesar que la pelota esté cerca del portería contraria, es un estado potencialmente ofensivo, que está en dependencia del porcentaje de riesgo. El análisis de riesgo se analizará con mayor detalle en la sección 4.7.

Defensivo

Un equipo ingresa a un estado defensivo *Defensive* por la combinación de dos factores: la posición de la pelota 'cerca' de nuestra portería y el análisis de riesgo de los jugadores contrarios. Con 'cerca' se entiende que la pelota está situada en la mitad de la cancha que nos corresponde, de esta forma, si nuestro equipo es el Rojo y la pelota se encuentra cerca de nuestra portería, nuestro equipo entra en un estado potencialmente defensivo. Sin embargo, similar al caso anterior, el análisis de riesgo, identifica los jugadores contrarios que están cercanos y calcula un margen porcentual para la ejecución de una estrategia. El análisis de riesgo brinda certeza para la ejecución de una acción pero no define en totalidad el curso del equipo. El análisis de riesgo se analizará con mayor detalle en la sección 4.7.

4.4.2. ESTADOS DE LOS JUGADORES

Como se mencionó anteriormente cada equipo está conformado por cuatro jugadores un portero *goalKeeper* y tres jugadores de campo *fieldPlayer*. La función esencial del *goalKeeper* es prevenir la anotación de goles por parte del equipo contrario. Los *fieldPlayer* recorren la cancha con el objetivo de obtener la pelota, pasarla, bloquearla, anotar goles, entre otros.

rol es la asignación de una tarea que un robot puede ejecutar en algún momento dado. Según la posición un agente puede adquirir de forma indirecta (no solicitado) un rol *ofensivo* o *defensivo*.

líder es un tipo especial de rol en el cual el agente que lo adquiere de forma directa (lo solicita) se convierte en el *capitán* del equipo por un tiempo determinado, dentro de su función principal es dar inicio al proceso de selección de estrategias cuando el estado del juego lo permita. Todos los agentes, incluyendo el *goalKeeper*, pueden ser *leaderAgent*.

Estados del Portero

A continuación se definen los principales estados que un *goalKeeper* puede experimentar, cabe destacar que no todos son eventos condicionales (verdadero o falso) mas bien pueden ser acciones o eventos que se ejecutan, estos a su vez pueden contener sub estados. Para efectos de ésta investigación se identificará al *goalKeeper* como `PLAYER 1`.

- goPosition: localizar al portero en su posición principal cerca de la portería.
- getLeadership: cuando solicita y obtiene el liderazgo.
- defendGoal: estado permanente.
- searchBall: acción que se ejecuta de forma constante para identificar la posición de la pelota.
- aproachBall: acción de obtener la pelota.
- blockBall: acción de despejar la pelota.

Estados de Jugadores de Campo

A continuación se definen los principales estados que un *fieldPlayer* puede experimentar, al igual que el *goalKeeper*, no todos son eventos condicionales (verdadero o falso) mas bien pueden ser acciones o eventos que se ejecutan, estos a su vez pueden contener sub estados. Para efectos de ésta investigación se identificará a los jugadores de campo como `PLAYER 2`, `PLAYER 3`, `PLAYER 4`.

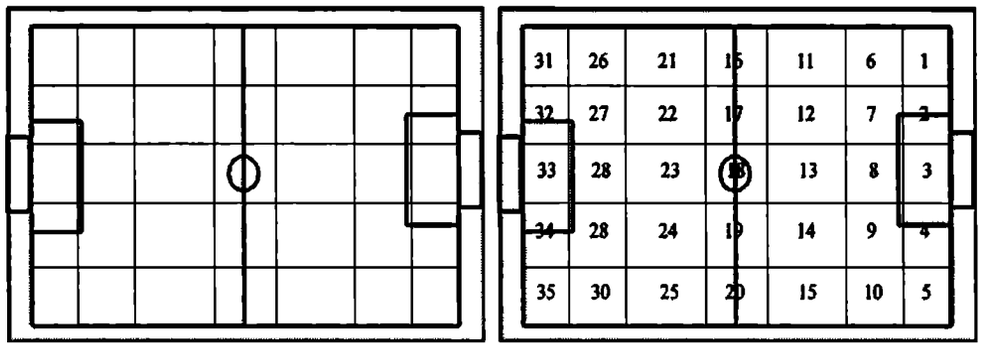
- goMasterPosition: localizar al jugador en una posición correspondiente a un rol de referencia.
- goToQuadranPosition: el jugador se localiza en una posición aproximada de uno de los cuadrantes.
- getLeadership: cuando solicita y obtiene el liderazgo.
- searchBall: acción que se ejecuta de forma constante para identificar la posición de la pelota.
- aproachBall: acción de obtener la pelota.
- turnWithBall: acción de girar con la pelota.
- shootToPosition: acción de disparar la pelota de forma aproximada en una posición.
- goalShot: disparar a portería.

4.5. ESTRATEGIA Y BASE DE ESTRATEGIAS

Estrategia es una secuencia coordinada de acciones de un equipo que representa un plan deliberativo predefinido en un sistema multiagentes.

Base de Estrategias es el conjunto de *estrategias* que están disponibles para ser ejecutas por un *equipo*.

Una *estrategia* se puede representar aplicando lenguaje de acciones sencillas, en las que se especifiquen los comportamientos de cada uno de los jugadores que intervienen en la estrategia. Así mismo se puede aplicar una notación simbólica que permita identificar de forma visual las características básicas de la estrategia. En la Figura 4.4 se muestra una representación del ambiente de la cancha de juego y la división en 35 cuadrantes sobre la cual se definirán las jugadas. Esta figura se proporciona sólo para fines de apoyo visual, la aplicación real se realiza en el simulador *XML Behaviour Control*.



(a) Cancha (b) Cancha dividida

Figura 4.4: Representación simbólica de la cancha dividida en cuadrantes

Para aplicar el razonamiento basado en casos es necesario emplear un proceso de *indexación*, que se analizará posteriormente, el cual permite registrar de forma ordenada y específica la información relacionada a una estrategia, para posteriormente aplicar procesos de *recuperación* por medio del índice generado. Es por esto que se optó por escribir la representación de una misma estrategia de dos formas, una versión extendida y una versión simplificada. Esta división responde a dos fines diferentes, tal como se verá a continuación:

Estrategia Extendida es la definición de una estrategia que incluye la especificación de cada una de las acciones que los jugadores aplican de forma secuencial, vital para el proceso de ejecución. Contiene información relacionada al identificador de la estrategia, máquinas de estado y transiciones entre estados de cada una de las partes de la estrategia y un identificador que la relaciona con la versión simplificada.

Estrategia Simplificada es la definición de una estrategia que incluye los atributos básicos útiles para el proceso de selección, contiene información relacionada a las condiciones ideales

para la selección de una estrategia en particular, entre estas: tipo de estrategia, posiciones de jugadores, posición de la pelota y un identificador que lo relaciona con la versión extendida.

4.5.1. REPRESENTACIÓN DE UNA ESTRATEGIA

Las estrategias en su forma extendida son especificadas usando un lenguaje de comportamientos de fácil comprensión en un archivo de tipo ".xml". Este lenguaje de comportamiento especifica el conjunto de estados y transiciones entre estado de cada estrategia, que incluye los eventos y las acciones para cada una. Posteriormente, se realiza la lectura de archivo y se ejecuta en el simulador *XML Behaviour Control.*

El conjunto de estrategias especificadas dentro del archivo forma la base de estrategias que pueden ser aplicadas por los jugadores durante el partido. Cada robot es creado con un archivo de tipo ".xml" conteniendo la base de estrategias, de esta forma todos comparten la misma base de conocimiento. En el apartado 4.5.2 se abordará en mayor detalle la definición de una estrategia extendida.

Las estrategias en su forma simplificada son también escritas en un archivo de tipo ".xml", con la salvedad que es un solo archivo (archivo de índices), que facilita su lectura, análisis y posterior aplicación durante la ejecución de la simulación, en el apartado 4.5.3 se proveerá un mayor detalle. Así mismo, para la definición de una estrategia por medio de un archivo de tipo ".xml" - extendido y simplificado, consultar el Apéndice C.

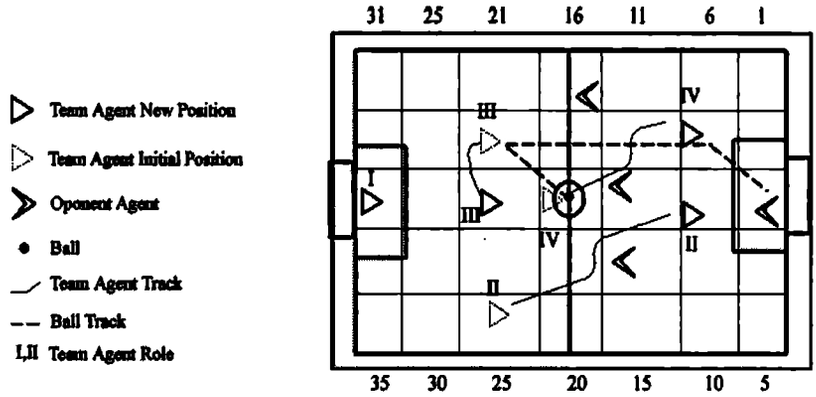
4.5.2. SIMBOLOGÍA - ESTRATEGIA EXTENDIDA

Como se mencionó al inicio de este capítulo, uno de los aspectos innovadores de la arquitectura para definición y selección de estrategias, es la descripción y aplicación de una simbología o notación que fuese de utilidad al momento de diseñar una estrategia y así mismo al momento de programarse y ejecutarse.

A pesar que el uso de símbolos descriptivos, es a menudo utilizado por equipos deportivos para la definición de estrategias, no así, la definición de una notación como tal. En [Marling et al., 2003] se encuentra una aproximación de uso de una notación en un entorno de juego para el fútbol robótico. Esto impulso la incorporación de *SSDS Soccer Strategy Description Symbols*, Descripción Simbólica para la Estrategias aplicadas a Fútbol, como mecanismo para la comprensión rápida y en un solo vistazo, de la descripción de una estrategia.

Cabe destacar que a pesar que la notación tiene una fuerte orientación hacia el entorno de juego de fútbol, en general, con algunas modificaciones puede ser aplicado a otros entornos, por ejemplo, podrían cambiarse los *adversarios* por otros agentes o *"target"*, la pelota por otros *objetos* o material de estudio, entre otros.

En la Figura 4.5 se muestra un ejemplo de la notación simbólica SSDS de una estrategia de tipo *KickOff* y la simbología empleada para su análisis. En el Cuadro 4.1 se muestra la representación textual de la Figura 4.5 ésta corresponde a la terminología empleada por los jugadores en la representación de sus estados, tal como se vió en la sección 4.4.2. En la imagen, los números en los bordes de la cancha representan un apoyo para identificar el número de cuadrante.



(a) Simbología (b) Estrategia de KickOff

Figura 4.5: Representación simbólica de una estrategia. Los números laterales representan los cuadrantes

	ESTRATEGIA		
	ID	K02	
	PLAYER 1	defendGoal	[33]
	PLAYER 2	goToPosition	[8]
		approachBall	[8]
		passViability	[7]
		shootToGoal	[3]
	PLAYER 3	waitPass	[22]
		approachBall	[22]
Roles		sendPassToPosition	[7]
		defendPosition	[23]
	PLAYER 4	approachBall	[18]
		sendPassToPosition	[22]
		goToPosition	[7]
		waitPass	[7]
		approachBall	[7]
		passViability	[8]
		shootToGoal	[3]

Cuadro 4.1: Representación textual de una estrategia K02

En el cuadro 4.1, se muestra la ejecución de la estrategia **KO2**, en términos generales participan

cuatro jugadores representados por las columnas, las filas refieren los comportamientos que ejecutan de forma secuencial cada jugador. Cada comportamiento está identificado por el nombre del estado que encapsula acciones y eventos y la posición en la que ejecutará tal comportamiento, acción o acciones.

Así el `PLAYER 1` ejecuta un comportamiento de portero, para tal caso su objetivo primordial en la ejecución de la estrategia, es proteger la portería por medio de sub comportamiento o estado *defendGoal*, el `PLAYER 3` ejecuta un comportamiento defensivo, de esta forma estará localizado, luego que realice un pase, en la posición 23 y realizando el estado *defendPosition*, los jugadores `PLAYER 2` y `PLAYER 4` serán los principales protagonistas de la estrategia, el `PLAYER 4` iniciará el proceso acercándose a la pelota en el cuadrante 18 (que está localizada en el centro de la cancha) y se la envía a `PLAYER 3`, que se encuentra en estado de espera en el cuadrante 22 para posteriormente tomarla y reenviarla a `PLAYER 4` que esta esperándola en la posición 7, este último la alcanzará y, finalmente, hará disparo a portería en el cuadrante 3, mientras el `PLAYER 2` se aproximará al área de acción para apoyar al `PLAYER 4`, en caso que el disparo de `PLAYER 4` sea bloqueado por el portero, ambos, `PLAYER 2` y `PLAYER 4`, iniciaran un comportamiento reactivo como medida auxiliar en la ejecución de la estrategia.

En la Figura 4.6 se muestra un extracto de la representación de la estrategia por medio de un archivo de tipo ".xml" que corresponde al `PLAYER 4`. Se observa que en la etiqueta `<behavior name>` se define el nombre descriptivo del comportamiento y posteriormente el atributo `<ID>` define el identificador de la estrategia, útil para su búsqueda posterior, a continuación se detallan los estados con la etiqueta `<state>` y las transiciones a otros estados cuando se activan ciertos eventos. En el Apéndice C se proporciona un mayor detalle de los elementos que componen la definición de estrategia.

4.5.3. INDEXACIÓN - ESTRATEGIA SIMPLIFICADA

indexación es el proceso por el cual se selecciona los atributos más importantes de una estrategia y se organizan en clases acordes a estos atributos.

índice de estrategias es el conjunto de todas las estrategias de la *base de estrategias* que experimentaron el proceso de **indexación**

Las estrategias en su forma simplificada, representan el **índice de estrategias**, este a su vez, similar a la *definición* de una estrategia, se consolida en un archivo de tipo ".xml", con la salvedad que el nombre es fijo y este es `indexing.xml` que se colocan en el directorio principal de la aplicación. En general este archivo contiene la base de *casos* que hará uso la aplicación para ejecutar la selección de la estrategia más acorde, de acuerdo a la *situación* del mundo.

El archivo de indexación se actualiza cada vez que se crea una nueva estrategia en la base de

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<behavior name="KickOff" ID= K03 isInitialBehavior="true">
        <state name="aproachToBallKickOff">
                <complexAction name="aproachToBall"></complexAction>
                <transition state="enviarPase">
                        <event name="VISION-shootDistance"/>
                </transition>
        </state>
        <state name="enviarPase">
                <action id="COMPLEX-shootToPosition" param1="17" param2="1"/>
                <complexAction name="turnToPass"></complexAction>
                <transition state="goToGoal">
                        <action id="MOTION-shootLeft"/>
                        <event name="COMM-readyToPass"/>
                </transition>
        </state>
        <state name="observarBalon" iterate="true">
                <complexAction name="observarBalon"/>
                <transition state="aproachToBall">
                        <event name="VISION-ballFound"></event>
                        <boleanEvent name="VISION-ballVeryNear"/>
                </transition>
        </state>
</behavior>
```

Figura 4.6: Parte del archivo de definición de estrategias

estrategias y es cargado en memoria al iniciar la aplicación. La representación de un *caso* y la definición de la *situación*, así como el detalle del archivo `indexing.xml` se analizará en las secciones posteriores.

4.6. DEFINICIÓN DE UN CASO PARA EQUIPO DE AGENTES

caso es la descripción del entorno desde el punto de vista de un equipo (definición de problema) y la solución que el equipo debe aplicar para el problema.

base de casos es el conjunto de casos.

Un caso contiene las *posiciones*, con base en la definición de cuadrantes hecho previamente, de los jugadores y la pelota, esto se puede puntualizar como el problema; y la *identificación* de la estrategia que responde de la forma más adecuada a tal configuración, esto conformaría la solución, además contiene la información del *tipo* de estrategia para efectos de clasificación en el *índice de estrategias*. De esta forma un *caso* puede definirse en la siguiente ecuación:

$$C = \{(a_0 : V_0), (a_1 : V_1), ..., (a_n : V_n)\}, \tag{4.1}$$

donde $A_i = (a_i : V_i)$ es un par compuesto de un atributo y su valor. En el Cuadro 4.2 se presenta la definición de atributos, descripción y un ejemplo de valores para cada atributo.

De esta forma la base de casos esta compuesto por el par (p, s) donde p es el problema y s la

Atributo	Descripción	Valor	Ejemplo
id	Identificador de la estrategia	Cadena compuesta por tipo y un consecutivo	K03
Tipo	Tipo de estrategia	0 de KickOff, 1 Ofensivo, 2 Defensivo	0
pl2-pos	Posición de jugador 2	Número de 1 al 35	22
pl3-pos	Posición de jugador 3	Número de 1 al 35	25
pl4-pos	Posición de jugador 4	Número de 1 al 35	18
bl-pos	Posición de la pelota	Número de 1 al 35	18

Cuadro 4.2: Atributos y valores de un caso

solución, así P_t^* es la unión de la definición del problema de todos los casos, en la base de casos, en un momento t y S_t^* es la unión de todas las soluciones. De esta forma la base de casos para un momento t esta dada por:

$$C_t^* = \{C_j \| C_j = \{p_j, s_j\}, p_j \in P_t^*, s_j \in S_t^*\} \tag{4.2}$$

La Figura 4.7 muestra parte del archivo `indexing.xml`, en el cual se observa que la etiqueta `<strategySet>` da inicio al archivo de índices al identificar el conjunto de estrategias, en su forma simplificada, que formarán la base de *casos*, a continuación la etiqueta `<strategy id>`, define el atributo identificador de la misma y la etiqueta `<name>`, es un apoyo para proporcionar una breve descripción de la estrategia, en el Apéndice C se proporciona un mayor detalle de los elementos que componen la índice de estrategias.

De esta forma, en el momento que se ejecute el proceso de selección de estrategias, se definirá una

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<strategySet>
        <strategy id="K03" name="Right_kickOff" useTimer="7000" type="0">
                <and>
                        <pl-pos name="Pl2Pos" value="25"/>
                        <pl-pos name="Pl3Pos" value="22"/>
                        <pl-pos name="Pl4Pos" value="18"/>
                </and>
                <bl-pos value="18"/>
        </strategy>
        <strategy id="OF7" name="Right_ofensive" useTimer="7000" type="1">
                <and>
                        <pl-pos name="Pl2Pos" value="6"/>
                        <pl-pos name="Pl3Pos" value="3"/>
                        <pl-pos name="Pl4Pos" value="2"/>
                </and>
                <bl-pos value="4"/>
        </strategy>
</strategySet>
```

Figura 4.7: Parte del archivo indexación de estrategias

situación que entrará al proceso de selección y determinará cuál es la estrategia, dentro del índice

de estrategias, que cumple con mayor similitud, con las condiciones especificadas; por ejemplo si tomamos una "instantánea" del juego y determinamos la siguiente situacion (0, 26, 23, 18, 18) que significa que es una situación de tipo KickOff y los jugadores 2, 3 y 4 estan en la posición 26, 23 y 18 respectivamente y, la pelota está en la posición 18, el proceso de selección encontrará que bajo estas condiciones se debe ejecutar la estrategia K03, por ejemplo. A continuación un mayor detalle de este procesamiento.

4.7. ARQUITECTURA DE ESTRATEGIAS

En la categoría de robots cuadrúpedos de la iniciativa de investigación de *RoboCup*, cada robot actúa de forma autónoma usando únicamente la información local disponible. La información es la secuencia de cada percepción, es decir, cada entrada que ha sido captada por los sensores del agente en un momento determinado. Pero es usualmente imposible implementar una tabla de almacenamiento que contenga una acción por cada posible secuencia percibida, especialmente en un ambiente dinámico e incierto como el fútbol robótico. Ésta es la principal razón por la que se ha decidido implementar una lista de estrategias predefinidas o planes de equipo que los agentes seguirán de forma consecutiva.

Así mismo, el carecer de un controlador global, una cámara dispuesta de tal forma que tiene acceso a todo el entorno de juego, como ocurre en otras categorías -la de robots pequeños- es considerado como una desventaja debido a que tal configuración no existe en la categoría de cuadrúpedos. A esto se le considera un sistema distribuido de múltiples robots sin un líder que pueda dirigirlos o guiarlos. Una solución a esta limitante es implementar una arquitectura de estrategias de juego con base en la asignación de roles para cada agente, con el fin de colaborar como un equipo tal como sucede en un entorno de fútbol real. La arquitectura propone además, la selección de un agente líder, parte del proceso coordinado entre los agentes, como mecanismo regulatorio en la función de búsqueda, selección y ejecución de estrategias de juego que permitan la realización exitosa de la meta del equipo.

Otras definiciones importantes en la arquitectura son la formación y la asignación de roles, éstas describen la forma en la que los jugadores se localizarán en el terreno de juego y su comportamiento en determinado momento. Diferentes formaciones y asignación de roles pueden ser aplicadas en dependencia de la actitud del equipo (patada inicial, ofensivo o defensivo) que corresponden a los cambios que el equipo experimente a lo largo del juego.

En la Figura 4.8 se muestra la arquitectura de estrategias de juego propuesta para el trabajo en equipo en fútbol robótico. En el resto del documento se usará el termino *TABA* para hacer referencia a la arquitectura, este proviene de las siglas en inglés de *Team Agent Behavior Architecture*. Esta arquitectura se caracteriza por la descomposición de tareas en forma jerárquica, con base en la planeación y ejecución de acciones. Posee cuatro etapas principales: selección del agente líder, selección de la estrategia, asignación de roles y ejecución de la táctica.

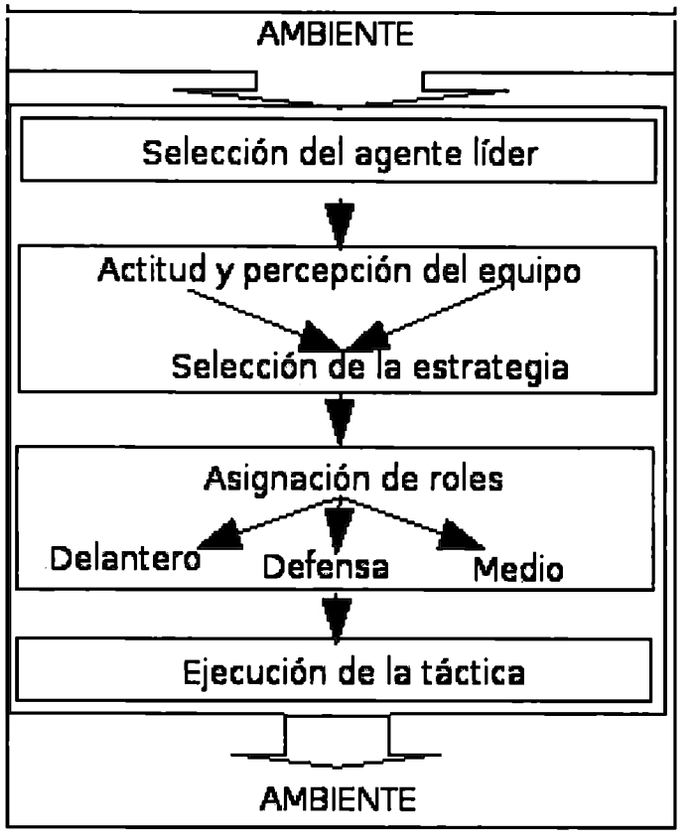


Figura 4.8: Arquitectura de estrategias de juego - TABA

En términos generales, el procesamiento en *TABA* comienza al dar inicio un partido, como se observa en la Figura 4.8 el ambiente afecta y es afectado por la arquitectura. El ambiente es toda aquella información que modifica el progreso del juego, tales como: localización de nuestro equipo, adversarios y pelota; anotación de goles a nuestro favor o en contra; tiempo transcurrido y restante, *etc.*, a continuación se detallará cada una de las etapas de *TABA*.

4.7.1. SELECCIÓN DEL AGENTE LÍDER

Un agente líder tiene la habilidad de tomar una decisión acerca de la mejor estrategia para una situación específica. El liderazgo es un rol dinámico en nuestra propuesta y es asignado al agente que posee un comportamiento pasivo en determinado momento del juego, de esta forma, se evita entorpecer las acciones de algún jugador activo -que esté ejecutando alguna acción- al incorporarle más carga de procesamiento. El Algoritmo 1 muestra el proceso general de la selección del agente líder.

Los aspectos principales para la selección del agente líder son: el estado del juego (patada inicial) y la distancia de la pelota a los jugadores. El algoritmo analiza en principio el estado del juego, por

```
input  : Game state
output: Leader agent
if gameState = kickOff OR ballPosition = opponentSide then
    leaderAgent =GoalKeeper;
end
else
    if ballPosition = ownSide then  foreach teammate i of the line N do
        FindDistanceBall(d)
    end
    leaderAgent =TeammateMaxDistanceToBall;
end
```

Algorithm 1: selección del agente líder

ejemplo, sí se está iniciando el juego, por defecto el rol de agente líder es otorgado al portero, esto responde al hecho que éste agente -el portero- tiene un comportamiento pasivo en comparación con el resto de los jugadores. Por el contrario, sí el estado del juego no es de inicio y la pelota está cerca de nuestra portería, el rol de agente líder se le asignará al agente que esté más lejos de la pelota, en otras palabras, el agente con menos actividad en ese momento.

4.7.2. SELECCIÓN DE LA ESTRATEGIA

Vega *et.al.* hace mención sobre la definición de estrategia estableciéndola como un plan de equipo firme y consistente que involucra una serie de parámetros y condiciones de selección, ejecución y terminación, que se lleva a cabo en un campo de juego, [Vega et al., 2006]. La selección de la estrategia es la parte medular de TABA debido a que es el proceso en el cual los agentes toman una decisión en equipo. En la Figura 4.9 se muestra de forma general la selección de una estrategia.

Una vez que se realiza la solicitud de ser líder de inmediato se solicita a todos los agentes sobre la información que estos perciben en su alcance de visión. La información que se obtiene se relaciona a la posición en coordenadas X y Y del agente, posición estimada en coordenadas X y Y de los agentes contrarios que logra identificar, y la distancia aproximada hacia la pelota. Una vez obtenida la información se genera la definición de un caso que entrará al proceso de Cálculo de Similitud.

Cálculo de Similitud

El cálculo de similitud entre casos es un aspecto importante dentro del razonamiento basado en casos en la fase de recuperación. La efectividad de la medida de similitud determina la utilidad de un caso o estrategia recuperado para resolver un nuevo problema. Existen, en general, dos formas de recuperación principales. La primera se basa en el cálculo de la distancia entre casos, donde el caso más similar se determina por la evaluación de una métrica de similitud, y la segunda se rela-

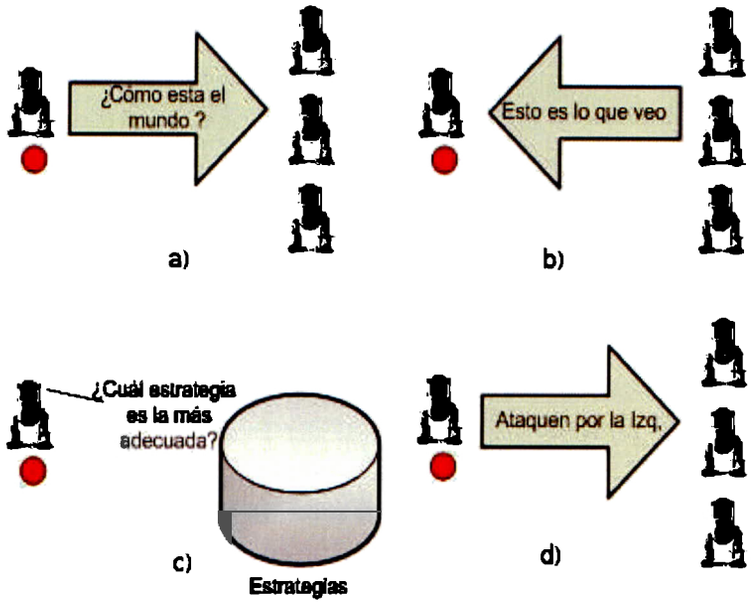


Figura 4.9: Selección general de una estrategia

ciona a la representación por medio de estructuras de índices de cada caso. Para esta aplicación se generó una combinación de ambas aproximaciones al hacer uso de una métrica de similitud como lo es el *K-vecino más cercano* sobre datos que han sido representados por medio de índices.

El algoritmo del *K-vecino más cercano* mide la distancia entre una situación específica durante el juego y un conjunto de estrategias previamente definidas. De esta forma la selección de la estrategia más acorde para un momento dado será aquella estrategia del conjunto de estrategias que tenga una medida de similitud mayor con una situación propuesta o mejor dicho aquella cuya distancia entre una y otra sera menor.

Para calcular la distancia entre dos situaciones se hace uso de funciones de distancia $d(x, y)$, donde x y y son casos de estrategias compuestas de F características, tal que $x = x_1, ..., x_F$, $y = y_1, ..., y_F$. Para esta aplicación se hace uso de la **distancia absoluta** dada por:

$$d_A(x, y) = \sum_{i=1}^{F} |x_i - y_i| \tag{4.3}$$

Así mismo se hace uso de la **distancia euclidiana** definida por la ecuación:

$$d_E(x, y) = \sum_{i=1}^{F} \sqrt{x_i^2 - y_i^2} \tag{4.4}$$

Debido a que la distancia entre dos casos es dependiente de intervalos, es recomendable que el resultado de las distancias sean escalados haciendo uso de una **media aritmética** de 0 sobre el conjunto de estrategias, y una **desviación estándar** de 1. Esto se lleva a cabo reemplazando las escalas de x y y con x' y y' acorde con la ecuación 4.5:

$$x' = \frac{x - \bar{x}}{\sigma(x)} \tag{4.5}$$

Donde x es el valor sin escalar, $\bar{x}$ es la media aritmética de la característica de x sobre el conjunto de casos o estrategias, ver Ecuación 4.6, σ es su desviación estándar, ver Ecuación 4.7 y, x' es el valor resultante ya escalado. De esta forma la media aritmética está dado por:

$$\bar{x} = \frac{1}{F}\sum_{i=1}^{F} x_i \tag{4.6}$$

La desviación estándar se calcula de acuerdo a la ecuación:

$$\sigma(x) = \sqrt{\frac{1}{F}\sum_{i=1}^{F}(x_i - \bar{x})^2} \tag{4.7}$$

La aplicación permite modificaciones en la configuración del cálculo del *K-vecino más cercano*, de esta forma puede emplearse cualquiera de las funciones de distancia, Ecuación 4.3 ó Ecuación 4.4, y ya sea utilizar los valores sin escalar o transformarlos haciendo uso de la Ecuación 4.5. A este punto se destaca que para la fase de prueba de la aplicación se opto en principio por aplicar la función de distancia absoluta para determinar, a primera vista, la eficiencia del cálculo en general, posteriormente se incorporó la distancia euclidiana basada en la localización de objetos en un espacio Euclidiano. En la Figura 4.10 se muestra parte del archivo de configuración que tiene por nombre `KnnConfig.java` en el cual puede ser definido la función de distancia a Euclidiana con la variable $KNN_DISTANCE_EUCLIDEAN$ especificada a **true**, si se cambia a **false** se implementará la distancia absoluta. Para mayor detalle ver Apéndice D.

La distancia Euclidiana ponderada para el conjunto de estrategias esta expresada por la matriz $SB = e_1, e_2, ..., e_N$ que representa el *Strategy Base* conteniendo N estrategias o casos. Cada caso esta representado por un índice con su correspondiente característica, tal como se comento en la Sección 4.6. De esta forma cada caso conforma una estrategia o curso de acción para todo el equipo. Así, se usa una colección de características $F_j(j = 1, 2, ..., n)$ para indexar los casos (definir un problema p) y una variable s para identificar la estrategia o curso de acción.El caso número i e_i en la base de estrategias puede ser representado como un vector $(n + 1)$ dimensional, que es, $e_1 = (x_{i1}, x_{i2}, ..., x_{in}, \Theta_i)$ donde x_{ij} corresponde al valor de la característica $F_j(1 \leq j \leq n)$ y Θ_i corresponde al valor de la acción $s(1, 2, ..., N)$.

Así mismo para cada característica $F_j(1 \leq j \leq n)$ se le ha asignado un peso w_j tal que $w_j \epsilon [0, 1]$ resultante de un proceso de valuación escalar, vista en las ecuaciones descritas anteriormente, que indica un valor de importancia de esa característica. A este punto podemos mencionar, por ejemplo, que la característica *type* que contiene los valores de *Kickoff, Ofensive, Defensive* tiene una peso de mayor relevancia sobre la posición de algún jugador.

Cabe aclarar que en la fase de prueba de la aplicación, todas las características tienen el mismo peso o importancia, ésta funcionalidad también puede ser configurada en la aplicación, tal como la función de distancia, en la Figura 4.10 se muestra parte del archivo de configuración que tiene por nombre `KnnConfig.java` en el cual puede ser definido si se hará ponderación de los pesos para cada característica con la variable $KNN_DISTANCE_SCALED$ especificada a **true**, si

se cambia a **false** no se realizará ponderación. Para mayor detalle ver Apéndice D.

Finalmente el cálculo de la similitud para cada par de casos e_p y e_q donde e_p representa un caso en la base de estrategias y e_q una situación nueva, una métrica de distancia ponderada puede ser definida como:

$$d_{pq}^{(w)} = d^{(w)}(e_p, e_q) = \left[\sum_{j=1}^{n} w_j^2 (x_{pj} - x_{qj})^2\right]^{1/2} = \left(\sum_{j=1}^{n} w_j^2 x_j^2\right)^{1/2} \tag{4.8}$$

Donde $x_j^2 = (x_{pj} - xqj)^2$. Cuando todos las características o pesos son iguales a 1, la métrica de distancia ponderada determina una distancia Euclidiana de $d_{pq}^{(1)}$.

Un aspecto importante en la selección de la estrategia del SB más acorde a una situación dada, es determinar un número de *vecinos* más cercanos al caso en cuestión, es decir, la aplicación provee la posibilidad de seleccionar el más idóneo dentro de un subconjunto de casos posibles, esto permite asegurar que la estrategia seleccionada es la más acorde e incluso brinda la posibilidad de tener disponible una estrategia secundaria que minimice el tiempo de selección de una nueva estrategia en el caso que la situación actual ya se haya presentado en el juego anteriormente.

En la Figura 4.10 se presenta la variable de tipo entero $KNN_NEIGHBORS$ que específica el número de *K-vecinos más cercanos* que la aplicación calculará, éste está contenido en el archivo de configuración que tiene por nombre `KnnConfig.java`. Para mayor detalle ver Apéndice D.

```
public interface KnnConfig {
        public  IndexDefinition idt = new IndexDefinition();

        public final static boolean KNN_DISTANCE_EUCLIDEAN = true;
        public final static boolean KNN_DISTANCE_SCALED = false;
        public final static int KNN_NEIGHBORS = 3;
        public final static int KNN_THRESHOLD=2;
(...)
}
```

Figura 4.10: Configuración del Cálculo de similitud

Al momento de realizar las pruebas de la aplicación se identificó, que a pesar de determinar un número de vecinos más cercanos, aún no era suficiente para asegurar la mejor decisión, debido a que las distancias entre la situación nueva y las registradas como vecinas podrían aún tener una valor alto, es decir, a pesar que eran las más cercanas, el grado de similitud era muy bajo, lo que implicaba que reducía notablemente la eficiencia de los jugadores al tratar de ejecutar una estrategia que era obviamente muy lejana a las posiciones actuales.

Para solventar esta problemática se incorporó una última medida de control que se conoce como *threshold* (umbral) de selección, que fija un valor, de tipo entero, que debe cumplirse como requisito para la selección final. Básicamente es determinar la distancia mínima posible entre la situación

actual y la estrategia selecciona de la SB. Como los aspectos anteriores, en la Figura 4.10 se presenta la variable $KNN_THRESHOLD$ que específica el umbral de selección.

El *Strategy Base* puede ser representado como una matriz $C = P * F$, que contiene P escenarios o estrategias de $s_1, ..., s_P$, donde cada escenario s_i contiene N características $s_i = s_{i1}, s_{i2}, ..., s_{iN}$. Un vector v de longitud P acompaña a esta matriz, listando los valores de salida v_i para cada escenario s_i. Este vector v puede ser visto como las columnas de la matriz y se extenderá a medida que las características para la selección de la estrategia aumenten.

De forma general el proceso de selección de estrategia acontece de forma inmediata luego que el agente líder es seleccionado, de esta forma, éste inicia la búsqueda tomando como valor de entrada la información proveída por sus compañeros - posición de jugadores, posición de contrarios, posición de pelota, actitud del equipo -. Se genera una *situación* nueva que entrará en un proceso de búsqueda sobre SB, para determinar los *K-vecinos mas cercanos* y la selección final de la estrategia, si alguno cumple con el *umbral* establecido. El Algoritmo 2 presenta el entorno general del la selección de estrategias.

```
input : New Situation
output: Strategy to Play
setNewSituation; while reach the end of the strategy set do
    calculateDistanceNewSituationStrategySet;
    mNearestNeighbors=minDistance;
end
foreach mNearestNeighbors i of the line N do
    calculateArithmeticMean
end
strategySelected =MAX(arithmeticMean);
checkThresold(strategySelected);
```

Algorithm 2: selección de la estrategia

4.7.3. ASIGNACIÓN DE ROLES

Una vez que la estrategia ha sido seleccionada, se les comunica al resto de agentes la identificación de la estrategia a ejecutar. Cuando los agentes reciben la comunicación del código de la estrategia, reciben de forma implícita su rol a ejecutar; el rol es dinámico, porque a pesar que inicialmente el agente posea una actitud ofensiva, si en la estrategia seleccionada se determina que será mas útil apoyando a la defensa este tendrá que actuar de acuerdo a la definición de la estrategia. En general todos los agentes apoyan en la realización de una estrategia, en un pase, en un despeje, para confundir al oponente o para anotar un gol. Cabe aclarar que la ventaja de la arquitectura es combinar planeación estratégica con acciones reactivas. Es decir si no hubo estrategia seleccionada

los agentes actuarán de forma reactiva tomando un rol por defecto que se les definió al momento de crearse.

4.7.4. EJECUCIÓN Y TERMINACIÓN DE LA ESTRATEGIA

La ejecución de la estrategia es la última fase de la arquitectura de estrategias, es el momento cuando los agentes, teniendo conocimiento de sus roles, aplican la estrategia seleccionada, y cada quien cumple cada una de sus acciones en dependencia de la definición de la estrategia seleccionada.

La ejecución describe cada acción que los agentes ejecutan en un orden que ha sido aplicado previamente en un entorno de prueba, como se mencionó anteriormente, las tácticas son las acciones rápidas en un periodo de tiempo corto, el conjunto de todas las acciones para todos los agentes conforma la estrategia general.

De la misma forma en esta fase de la estrategia se especifica qué condiciones se deben presentar para que los agentes terminen con la ejecución de la estrategia; ejemplo de estas condiciones de paro son: el tiempo de ejecución se haya excedido, la pelota haya salido del terreno de juego, se haya anotado un gol o que el juego haya terminado.

4.7.5. EJEMPLO

A continuación se proporcionará un ejemplo que engloba todas las partes de la arquitectura de estrategias. Se iniciará con la ejecución de una estrategia inicial hasta su terminación, posteriormente la selección de líder y selección de una nueva estrategia para reiniciar el proceso. La Figura 4.11 representa la ejecución de la estrategia **K03** que corresponde a la estrategia inicial de este ejemplo.

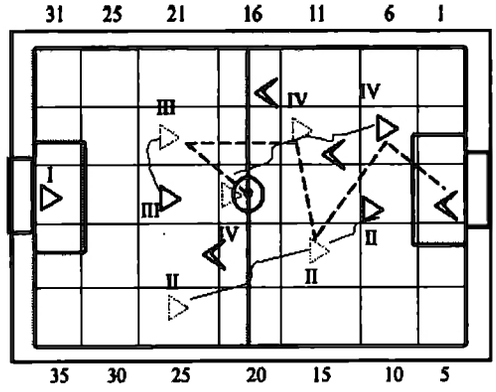


Figura 4.11: Representación simbólica de la estrategia KO3

Ejecución estrategia inicial K03

El Cuadro 4.3 muestra la definición de la estrategia de tipo *KickOff* **K03** así como las condiciones ideales para su ejecución. Esta información corresponde a la estrategia en su versión simplificada que está contenida en el archivo de índices `indexing.xml`.

	ESTRATEGIA	
DEFINICIÓN	ID	K03
	TYPE	0
	NAME	Left KickOff
	TIME	120
CONDICIONES	PLAYER 1	[33]
	PLAYER 2	[25]
	PLAYER 3	[22]
	PLAYER 4	[18]
	BALL	[18]

Cuadro 4.3: Definición y condiciones de la estrategia de KICKOFF K03

El apartado DEFINICIÓN contiene información relacionada a los datos generales de la estrategia: *ID, TYPE, NAME, TIME*. El campo *ID* permite relacionarla con la definición en su versión extendida; *TYPE* indica que es una estrategia de Patada Inicial o *KICKOFF*; *NAME* es un nombre descriptivo para su manejo; *TIME* es un valor de referencia del tiempo máximo para la ejecución de la estrategia. En el área de CONDICIONES, si indica las posiciones (representadas entre corchetes) ideales para la ejecución de la estrategia, en cuanto a la posición de los jugadores y de la pelota.

Posteriormente en el Cuadro 4.4 se muestra la definición de la estrategia en su versión extendida que contiene información relacionada a la ejecución de las acciones por cada agente. La información está contenida en los archivos de comportamientos de los cuatro agentes que participan en esta estrategia. Es importante aclarar que los comportamientos que los agentes ejecutan se refiere a los generales, estos pueden contener de forma interna acciones que en conjunto forman el comportamiento. Como se observa en el Cuadro 4.4 el `PLAYER 1` estará ejecutando en todo momento un comportamiento defensivo alrededor de la portería, identificado por la posición en el cuadrante con número 33. El `PLAYER 2` ira de su posición inicial 25 (definida en el Cuadro 4.3) a la posición 14 esperará un pase del `PLAYER 4` y se lo devolverá en la posición 7 luego irá a la posición 8 donde apoyará a `PLAYER 4` y de ser el caso, hará intentos por tirar a gol en portería contraria con posición 3.

El `PLAYER 3` esperará en su posición 22 un pase que le será enviado de `PLAYER 4`, posteriormente se lo regresará (una maniobra de despeje) a este mismo que se encuentra en la posición 12 y se incorpora, finalmente, en un estado defensivo en la posición 23. El `PLAYER 4` iniciará la estrategia aproximándose a la pelota y enviando pase al `PLAYER 3` en la posición 22 para posteriormente esperar un pase de regreso en la posición 12, una vez que obtenga la pelota, la enviará al

	ESTRATEGIA		
	ID	K03	
ROLES	PLAYER 1	defendGoal	[33]
	PLAYER 2	goToPosition	[14]
		waitPass	[14]
		approachBall	[14]
		sendPassToPosition	[7]
		goToPosition	[8]
		approachBall	[8]
		passViability	[7]
		shootToGoal	[3]
	PLAYER 3	waitPass	[22]
		approachBall	[22]
		sendPassToPosition	[12]
		defendPosition	[23]
	PLAYER 4	approachBall	[18]
		sendPassToPosition	[22]
		goToPosition	[12]
		waitPass	[12]
		approachBall	[12]
		sendPassToPosition	[14]
		goToPosition	[7]
		waitPass	[7]
		approachBall	[7]
		passViability	[8]
		shootToGoal	[3]
TERMINACIÓN	checkTimeOut		
	checkGoalScored		
	checkBallWentOut		
	checkAbortionMessage		

Cuadro 4.4: Ejecución y terminación de la estrategia KO3

PLAYER 2 en la posición 14 y esperará un pase de regreso en la posición 7 y, finalmente realizar intentos por anotar gol en la portería contraria en la posición 3, de no ser posible analizará la posibilidad de enviarse pases con el PLAYER 2. De la misma forma el cuadro muestra las condiciones terminación que están siendo verificadas en todo momento (anotación, pelota fuera, tiempo finalizado.

En la Figura 4.12 y 4.13 se muestra la ejecución secuencial de cada una de las partes de la definición de la estrategia K03 en el simulador *XML Behaviour Control*. La Figura 4.14 muestra

la última parte de la estrategia, la anotación de gol que culmina con el inicio de la selección de un líder y consecuentemente la selección de una estrategia. Una vez que se da la anotación en 4.14a entra en funcionamiento el inicio de la arquitectura y se puede apreciar en 4.14b como los agentes regresan a una posición de KickOff, ya teniendo una estrategia seleccionada.

(c) (d)

Figura 4.12: Ejecución estrategia K03 parte I en XML Simulator Control

En la Figura 4.15 se muestra la información correspondiente a la consola de ejecución del simulador *XML Behaviour Control* con la información generada por TABA al momento de realizarse la anotación. La comunicación se realiza por medio del protocolo *KQML* haciendo uso de las performativas **ask** y **reply**. Se envía una solicitud por parte del portero que tiene el nombre de **Golie Blue** para ser líder `LEADER_REQUEST` y adjunto incluye información sobre su posición en el eje de coordenadas X y Y `-378,0`, si ve a algún oponente `378,0`, posiblemente ve al otro portero, y la distancia de la pelota `378`.

El agente con nombre **Center** recibe la solicitud y responde con su propia información: posición `305,49` y en ese momento no ve a ningún oponente ni a la pelota. El agente con nombre **Left**

(c) (d)

Figura 4.13: Ejecución estrategia K03 parte II en XML Simulator Control

recibe la solicitud y responde con su propia información: posición `-123,1`, ve a un contrario en la posición `123,274` y la distancia de la pelota es de `49`. El agente con nombre **Right** recibe la solicitud y responde con su propia información: posición `202,126` y en ese momento no ve a ningún oponente ni a la pelota.

Posteriormente inicia el procesamiento para determinar al líder y una vez seleccionado se envía el aviso `LEADER_ROLE_READY`. Con la información de todos los agentes se crea el vector que identifica el tipo, posiciones de jugadores y posición de pelota, que ingresará al **Cálculo de Similitud** que básicamente se comparará con toda la matriz de `Base de Estrategias`. El vector `(0, 33, 7, 23, 7, 18, ??)` define que el tipo de estrategia es de KickOff con `0`, posición de jugadores del equipo están identificadas por `33,7,23,7`, la posición de la pelota en `18` y los símbolos de interrogación `??` representan la consula de cuál es la mejor estrategia en esa especificación de ambiente.

En la Figura 4.16 se muestra la continuación de la información de consola del simulador con

(a) (b)

Figura 4.14: Terminación estrategia K03 en XML Simulator Control

```
*******
GOL AZUL
Llego el mensaje de ser lider
LEADER_REQUEST 0
ask-if: 0 all
Send GolieBlue:(ask-if : content LEADER_REQUEST,-378,0,378,0,378 :
reply-with 0 : sender GolieBlue : receiver blue-all )

Receive- Center content: LEADER_REQUEST,-378,0,378,0,378
Send Center:(reply : content LEADER_ROLE_REPLY,305,49,-1 :
in-reply-to 0 : sender Center : receiver blue-GolieBlue)
Receive- GolieBlue content: LEADER_ROLE_REPLY,305,49,-1

Receive- Left content: LEADER_REQUEST,-378,0,378,0,378
Send Left:(reply : content LEADER_ROLE_REPLY,-123,1,123,274,49 :
in-reply-to 0 : sender Left : receiver blue-GolieBlue)
Receive- GolieBlue content: LEADER_ROLE_REPLY,-123,1,123,274,49

Receive- Right content: LEADER_REQUEST,-378,0,378,0,378
Send Right:(reply : content LEADER_ROLE_REPLY,202,126,-1 :
in-reply-to 0 : sender Right : receiver blue-GolieBlue)

SE activo LEADER_ROLE_READY
 Ya se definio el nuevo caso
Calculando los 3 vecinos mas cercanos a:
  (0, 33, 7, 23, 7, 18, ??)

*******
```

Figura 4.15: Selección de líder y definición de caso

la ejecución de los cálculos de similitud, con el vector generado se realiza el cálculo de lo tres *K-vecinos más cercanos* y se proporciona una selección final. Al terminar se reporta esto último en un archivo de tipo ".html" que contendrá la información relacionada a la selección de estrategias, ver Apéndice D, *CBR_Report_File.html.*

```
*******
Distancia de (0, 33, 7, 23, 7, 18, ??) a...
data: (1, 33, 9, 10, 2, 2, OF1)
...Strategy 1: 21.330729007701542
data: (2, 33, 5, 6, 3, 6, DF2)
...Strategy 2: 21.37755832643195
data: (0, 33, 6, 3, 13, 6, KO1)
...Strategy 3: 24.1039415863879
data: (1, 33, 5, 10, 13, 3, OF2)
...Strategy 4: 20.85665361461421

(...)
data: (0, 33, 7, 24, 9, 18, KO4)
...Strategy 14: 2.23606797749979

Neighbor #1: Strategy 14:
  distance = 2.23606797749979
  (0, 33, 7, 24, 9, 18, KO4)
Neighbor #2: Strategy 4:
  distance = 20.85665361461421
  (1, 33, 5, 10, 13, 3, OF2)
Neighbor #3: Strategy 6:
  distance = 21.118712081942874
  (0, 33, 25, 22, 18, 18, KO2)


Situacion esperada:
  (0, 33, 7, 24, 9, 18, KO4)
generando archivo de reporte: CBR_Report_File.html
*******
```

Figura 4.16: Cálculo de similitud y selección de estrategia

4.8. RESUMEN DEL CAPÍTULO

El trabajo colaborador y coordinado implica un alto control jerárquico sobre los comportamiento de cada uno de los integrantes de un equipo, con el fin de alcanzar un objetivo general o muchos, se define un conjunto de acciones más específicas para lograrlo o lograrlos. La arquitectura para la definición y selección de estrategias TABA se propone como herramienta útil en el mejoramiento del desempeño de un equipo por medio del uso de estrategias, en este caso aplicadas a fútbol robótico. Para lograrlo se identifican los siguientes aspectos de su naturaleza: (i) coordinar comportamiento de equipo, (ii) combinar secuencias de acción de planeación estratégica (deliberativo) con la habilidad de explotar oportunidades del momento (reactivas), (iii) aplicar metodologías para el razonamiento basado en casos y, (iv) fácil diseño, aplicación y pruebas por los operadores humanos.

En este capítulo se proporcionó en detalle la propuesta de arquitectura TABA, realizando una serie de definiciones básica para su compresión, la especificación y denominación de la nomenclatura y simbología usada SSDS y el detalle de cada una de las etapas de la arquitectura: selección del líder, selección de la estrategia (incluye cálculo de similitud), asignación de roles y ejecución (y terminación) de la estrategia. Así realizó un ejemplo con el detalle de cada uno de los pasos.

5. EXPERIENCIAS Y RESULTADOS

"Si B se comporta como A en algún aspecto, vale la pena suponer que lo hace en otros aspectos también. Ya sea que la hipótesis haya tenido éxito o no, algo habremos aprendido"

Mario Bunge, Filósofo y Físico [Bunge, 1968]

La solución a los problemas del mundo real no inician con un análisis de objetos matemáticos. Los datos iniciales son amenudo especificaciones físicas, económicas, sociales o de alguna característica asociada al problema de estudio. Esta investigación parte de una premisa "el trabajo colaborador y coordinado impulsará las acciones de un equipo para lograr un fin en común", esto desde una perspectiva oportunista, pero desde una punto de vista que denote debilidad, es visto como un problema "un equipo que no aplique procesos de cooperación no podrá maximizar el desempeño individual de sus integrantes". Nótese que se hace uso de la palabra *equipo* que bien puede ser aplicado a cualquier entorno en el cual participen dos o mas elementos, definiéndose a estos elementos, en este estudio, como agentes o robots.

Una vez definido el problema, procede construir un conjunto de alternativas posibles para solucionarlo; aislar los atributos más esenciales para la evaluación de las alternativas, diseñar un criterio, organizar las alternativas por aspectos y realizar estimaciones sobre la utilidad de las alternativas, es ahí donde se podría asociar un valor numérico que pueda dar solución a un problema. Usualmente los métodos para dar solución a problemas de estimación requieren del apoyo de un experto. El problema definido al inicio de la investigación, coordinación de un sistema de multiagentes, se apoya de forma coyuntural en la aplicación de procesos que son llevados a cabo en el mundo real por humanos.

En este capítulo se realizará un breve resumen de las principales alternativas que surgieron para dar solución al problema, específicamente la evolución de estas alternativas, a lo que se le ha denominado Experiencias, así mismo, el detalle sobre los principales logros obtenidos en el cumplimiento del método científico, como lo es el análisis de Resultados, vital para comprobar la hipótesis planteada en el primer capítulo.

5.1. EXPERIENCIAS

Uno de los primeros aspectos que requería la solución inmediata consistía en identificar como se lograría la coordinación entre agentes que compiten contra otro equipo, en un entorno de juego. De aquí surgió la idea de estrategias, aspecto que puede ser visto de forma trivial en un mundo real pero que implica todo un reto en el campo de los sistemas de multiagentes. A continuación se describirán las principales alternativas para dar solución al problema y sus transformaciones que los llevaron de la idea original hasta llegar a la implementación final. Se hace nuevamente la aclaración que el entorno de aplicación es un sistema de multiagentes que participan en una competencia de fútbol robótico bajo las reglas definidas en la categoría de estándar - cuadrúpedos de la organización RoboCup.

5.1.1. ESTRATEGIAS DE PIZARRA

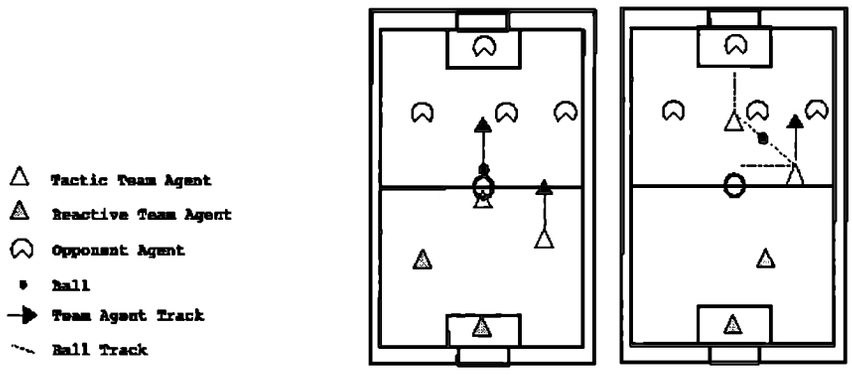
Similar a un entorno real, las estrategias de pizarra son ampliamente usadas por su facilidad en especificar cursos de acción que pueden ser adaptados y ajustados a las necesidades en dependencia de los cambios del medio. Antes de definir la aplicación de estrategias de pizarra, se ideó la aplicación de metodologías como Campos Potenciales, Procesos Ocultos de Markov, Teorema de Bayes por mencionar algunos, para predecir, estimar o aplicar planes de ruta para cada uno de los agentes. A pesar que estas opciones han sido aplicadas en diversos entornos, era parte de los objetivos analizar un tipo de forma que forma "similar" al mundo real.

La planeación de rutas es de mucha utilidad pero resuelve una parte del problema, como optimizar que un agente, o todos los agentes, logren llegar de un punto A a un punto B, considerando las necesidades de computo requeridas para el procesamiento constante de esta planeación. Pero esto no da solución a un problema mayor, era necesario que los agentes cooperaran, identificaran áreas de oportunidad como equipo, aplicaran cursos de acción que previamente ya habían sido aplicados y probados de forma global y, finalmente, combinaran esto último con acciones meramente reactivas que implican la acción de un sólo jugador en local, tal como se observa por ejemplo en el fútbol, existen jugadores que poseen destrezas propias y sobresaltan en ello pero a la vez son parte de un equipo y sus habilidades se combinan con una adecuada planeación estratégica de equipo.

5.1.2. SIMBOLOGÍA DE TÉRMINOS

Una vez que se optó por la aplicación de estrategias de pizarra de inmediato se identificó la necesidad de aplicar algún tipo simbología que describiera los acontecimientos ocurridos en la ejecución de una estrategia. Es común observar que los equipos deportivos e incluso los equipos de trabajo, hacen uso de algún tipo de simbología que les apoye en la definición de estrategias. Pero es muy usual que cada entidad aplica, a su mejor conveniencia, la terminología que todos tienen en común, jugadores, pelota, áreas críticas, cuando se refiere a fútbol. Se optó por aplicar y definir una simbología y proveerle de un nombre descriptivo para futuras referencias. De la misma forma esta ha sufrido cambios desde la idea inicial, en principio por que se dibujaba en varios planos cada etapa de la ejecución de la estrategia, al final se tenían, por ejemplo 5 ó 7 gráficas de la misma estrategia una por cada cambio, tedioso e inmanejable al momento de describir nuevas estrategias.

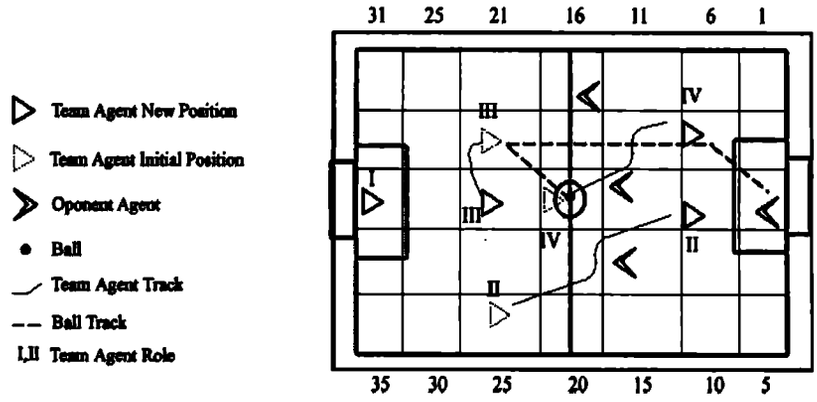
En un segundo momento se simplifico de tal forma que pudieran ser ajustados a una sola gráfica, esto implicaba una limitante palpable, cómo especificar un momento pasado de uno futuro, es ahí donde se incorporó otro elemento a nuestra lista de términos, la trayectoria, tanto de los agentes como de la pelota. Dado que los agentes se mueven constantemente en la cancha de juego, era necesario la identificación de quién es quién en un momento dado, de esta forma se ajusto la simbología con la inclusión de números romanos para asociarlos con los jugadores de nuestro equipo. Finalmente al incluir a los jugadores adversarios, inicialmente teniendo estos las mismas cualidades físicas se usaba el mismo símbolo, aspecto que fue actualizado a la identificación de un nuevo símbolo similar pero con características notablemente diferenciadas. En la Figura 5.1 se observa una representación de la simbología usada en las primeras versiones.



(a) Simbología (b) Estrategia de KickOff

Figura 5.1: Representación simbólica de una estrategia - primeras versiones

En la Figura 5.2 se muestra la versión final de la notación simbólica SSDS descrita en el capítulo anterior. Se notan un cambio incluso en la orientación de la gráfica de vertical a horizontal, este cambio responde a la forma implementada en el simulador *XML Behaviour Control*, así mismo la división de la cancha en regiones, es un aspecto que se abordará a continuación.



(a) Simbología (b) Estrategia de KickOff

Figura 5.2: Representación simbólica de una estrategia - versión final

5.1.3. ENTORNO DE JUEGO

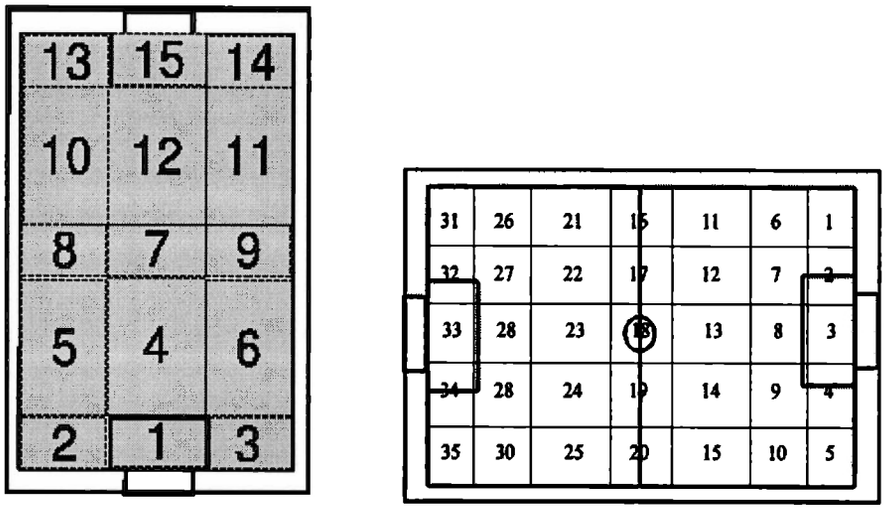
Una cancha de fútbol es el entorno de juego que aplica el sistema de multiagentes para la aplicación de una arquitectura de estrategias. La cancha ha sido dividida en regiones para definir las diferentes áreas de oportunidad durante la ejecución de una estrategia. Similar a un entorno real los jugadores de forma intuitiva se localizan de forma estratégica en determinadas áreas para aprovechar una oportunidad o ejecutar una jugada planeada.

Inicialmente la cancha se dividió en tres áreas grandes (dos cercanas a porterías y un central) que a su vez fue dividida en 15 regiones, con iguales dimensiones; una evaluación posterior se determino que podría explotarse aún más si las regiones tenían dimensiones diferenciadas, debido a que no es lo mismo defender el área chica de portería, que realizar un pase de media cancha. De esta forma se opto por modificar las dimensiones con base a las áreas estratégicas del entorno de juego.

Una vez implementada la arquitectura, en el diseño de pruebas y ejecución de estrategias, se observó que, si bien la división diferenciada de regiones era de mucha utilidad, el proceso de selección de estrategias incurría en la selección de un número finito, las mismas, de estrategias, esto se debía a que las condiciones eran propicias y similares, en una región amplia podían localizarse dos jugadores de mi equipo y la pelota y esto se repetía. Se optó por explotar aún mas las condiciones y dividir la cancha en 35 regiones, siempre considerando las áreas mas vulnerables - cerca de la portería o las esquinas - con dimensiones más pequeñas, y las áreas que se ven involucradas en movimientos o traslados - a campo abierto - más grandes. En la Figura 5.3 un ejemplo del entorno dividido de juego en una versión inicial y la versión final.

5.1.4. ARQUITECTURA DE COMPORTAMIENTO EN EQUIPO

La base de este trabajo de investigación se centra en la definición de una arquitectura que integre mecanismos para la ejecución de un comportamiento coordinado y colaborador entre agentes que



(a) Versión inicial (b) Versión final

Figura 5.3: Cancha de juego

compiten para obtener un beneficio o alcanzar una meta. Sorprendentemente, en rasgos generales la arquitectura y su división en cuatro capas definidas, selección del líder, selección de estrategia, asignación de roles y, ejecución y terminación de la misma, no ha variado en gran medida. Pero sí se considera importante destacar algunos aspectos que fueron propios de cada etapa.

Selección de líder

El líder es quien tiene la responsabilidad de coordinar a un equipo para alcanzar los objetivos planteados. Independientemente de las adversidades del entorno, el líder debe ser capaz de asumir riesgos y aprovechar oportunidades. Tomando en consideración que la categoría de robots cuadrúpedos es un entorno muy competitivo sobre todo por que requiere de destrezas en gran medida relacionadas a la autonomía de los agentes. A diferencia de otras categorías que poseen un procesamiento paralelo, una visión (por medio de cámaras globales) de todo el entorno de juego y en otros casos, la información del desempeño de todos los agentes propios y adversos (bitácoras de juego), los cuadrúpedos se valen única y exclusivamente de su visión limitada y local y deben responder a un estímulo, generalmente de corto plazo.

Aquí surgió la idea de una capitán, un líder, que pueda, a medida de sus posibilidades, aprovechar los protocolos de comunicación y explotar al máximo las oportunidades. Este líder tenía que ser uno del equipo, eso implicaba dos limitantes, la primera, como parte del equipo, este tenía sus responsabilidades propias y segundo también tenia una visión limitada del entorno. Inicialmente se acordó que el líder siempre fuera el portero, quien podría tener una visión mas general de la cancha, pero había un problema, cuando se entraba en una acción defensiva, un número de adversarios ejecutando intento de anotación, este ya no tendría la posibilidad de tomar una decisión objetiva sobre la siguiente ejecución del equipo. Finalmente el líder es un rol dinámico, que de

forma general sera asumido por quien este más lejos de la acción, que corresponde casi siempre a quien esté mas lejos de la pelota.

Selección de estrategia

En este aspecto se comentará sobre el algoritmo de selección de estrategias, que usa principalmente mecanismos de razonamiento basado en casos para su aplicación. En términos generales el proceso inicia inmediatamente que el líder es elegido, éste solicita a sus compañeros información relacionada a la visión local de los mismos (qué posición tienes, a qué distancia esta la pelota de ti, ves algún adversario) a esto le suma la situación en la que se encuentra el equipo (posición defensiva, ofensiva, de patada inicial, la pelota salió), genera un único vector que resume toda esta información, el cual entra a un proceso de cálculo de similitud contra toda una matriz de opciones, experiencias pasadas. Finalmente el proceso genera una recomendación de posibles cursos de acción y se toma una decisión final.

El cálculo general se basa en la identificación del *vecino más cercano* comparando cada uno de los atributos del vector contra cada uno de los atributos de cada elemento de la matriz, con base en una diferencia de distancias, valores absolutos. Pero posteriormente se identificó que no era una situación en general justa, debido a que no todos los atributos, en la realidad, tienen la misma importancia al momento de decidir, por ejemplo, estar en una situación ofensiva o defensiva, es un atributo que tiene mucho mas peso, a la hora de definir un curso de acción, que el atributo que indica si un jugador está en la posición 7 u 8. De esta forma se implementó una modificación que realiza una valuación sobre las prioridades que deben cumplirse.

Otro aspecto en cuanto a la selección de una estrategia y como parte del mismo proceso de cálculo del vecino más cercano, era que no sólo se identificará las estrategia más cercana, de la base de estrategias, si no también, las K estrategias más cercanas, donde K para esta investigación se fijo en tres, valor que fue definido por ser, empíricamente, lo que ocurre en la vida real; en un grado general el ser humano, visualiza al menos tres opciones para responder ante un hecho; esto está relacionado a la Teoría de Chunking en voz inglesa *Chunking Theory* de Chase and Simon citada en [Kimball and Holyoak, 2000].

Pero particularmente en la fase de pruebas no se ha recurrido a la opción B o C, que generalmente son un poco más distantes de la opción A. Se considera que una aplicación, por ejemplo en un juego real de robots cuadrúpedos (u otra plataforma), se pueda explotar esta funcionalidad, por ejemplo, los robots fallan, y lo que se decidió en un primer momento ya no se podrá resolver de esa forma, por que perdí a un recurso, entonces podemos analizar la ejecución de la opción B o C.

Finalmente se debe mencionar una última adecuación que procede enteramente de la fase de pruebas. Anteriormente se habló del cálculo de lo tres vecinos más cercanos, con base a un valor de tipo numérico, pero exactamente qué tan cercano. Esto se detectó cuando el algoritmo identificaba

en las primeras pruebas que las tres estrategias más cercanas tenían una distancia, por ejemplo de 25, 31 y 41, los agentes no podrían alcanzar las posiciones correspondientes sin ser interceptados por los oponentes, era un fracaso la selección.

Claro, esto responde a que no se contaba con una adecuada población de casos (estrategias) en la base de estrategias y a valores concretos y reales. Pero motivó la generación de un último mecanismo de control, la definición de un valor que sirva como *umbral* que descarte aquellas opciones que simplemente no están dentro de los limites. Es más eficiente que los agentes apliquen un comportamiento individual (reactivo) a que se obliguen a ejecutar una estrategia que no tenga las condiciones de aplicabilidad dentro de los límites. Durante las siguientes pruebas se determinó que un umbral menor de cinco era apropiado para una selección acorde a la situación.

Asignación de roles

Por lo general cuando se trata de organizar un equipo de trabajo, de estudio o un equipo de juego, se delimitan las acciones que cada integrante puede ejecutar o al menos debería ejecutar. A esto se le ha denominado la asignación de roles o comportamientos propios de cada participante. En un entorno futbolístico en general se puede clasificar de dos formas, la defensiva y la ofensiva; en un entorno más específico como la categoría de cuadrúpedos de RoboCup en la versión 2007 se cuentan con cuatro agentes que desempeñan acciones delimitadas a portero, defensa, delantero y apoyo, tal como se tiene implementado en la actualidad en la plataforma de robot reales del proyecto de investigación del ITESM-CEM. En la arquitectura propuesta se provee mecanismos para que la asignación de roles sea llevado a cabo de forma dinámica.

Una vez que se ha seleccionado la estrategia se les comunica a los agentes el código que identifica al curso de acción que deben aplicar de forma inmediata. Debido a que todos los agentes tienen en su definición de comportamiento la información de las estrategias disponibles, no es necesario que se le indique qué acciones deben ejecutar, con el código de estrategia, se les realiza la asignación implícita de sus rol dentro de la jugada. El aspecto dinámico implica que, según la definición de estrategia y la disposición de los recursos (agentes) dentro del terreno de juego, así sea el curso de acción; es decir, se relaciona más a un aspecto de posición dentro de la cancha que a un aspecto de función en el juego.

Sin embargo, como se mencionó anteriormente, la arquitectura combina una planeación estratégica con un comportamiento reactivo (cuando así se considere). Es por esto que se mantiene una funcionalidad propia del simulador *XML Behaviour Control* en su versión original, de especificar un rol inicial, por ejemplo *RIGHTDEFENDER, DINAMICATACKER* al momento que el agente es creado. De esta forma el agente puede ejecutar un rol por defecto en el caso que ninguna estrategia aplique.

Ejecución y Terminación

La ejecución es el seguimiento ordenado de las acciones contenidas en la definición de la estrategia seleccionada. Cada agente hace su mejor esfuerzo por cumplir la parte que le acontece en cada momento de la estrategia, y se dice su mejor esfuerzo, por que tal como pasa en la vida real, en una fase de practica y de juego, un jugador II envía un pase a un jugador III en la posición 25, pero existen probabilidades que el pase no sea perfecto, el jugador III no puede esperar pacientemente a que la pelota llegue cabalmente a la posición 25, tiene que poner de su parte para apoyar que esta acción sea exitosa. De la misma forma la arquitectura ha considerado estos aspectos y se le ha proveído de capacidad colaborativa para el lograr en la medida de lo posible la satisfacción de esa necesidad. Pero también se ha podido observar que a pesar que es deseable que se ejecute todo perfecto no siempre se logra.

La terminación es un aspecto que siempre se esta analizando en el proceso de ejecución, verifica las condiciones de paro de la estrategia y de forma consecutiva da inicio nuevamente a un proceso de selección de líder y estrategia. En este aspecto destacan principalmente como condiciones de terminación cuando la pelota ha sido enviada fuera de la cancha de juego y cuando ha habido alguna anotación.

5.1.5. PRACTICA VS JUEGO

Este es un aspecto que tiene dos caras, por un lado fue completamente desolador y por otro completamente reconfortante. Esto es por que a medida que se definían y probaban las estrategias para poblar la base de estrategias que se ha denominado como la fase de práctica, en la cual los agentes realizaban la operación, se identificaban las dificultades y se afinaban las acciones para que culminara, al menos en gran medida, en éxito. Pero esto se realizó sin considerar a un equipo oponente. Cuando se introdujo al equipo oponente (sin estrategias) se identificó casi de inmediato que era necesario realizar adecuaciones a las estrategias ya probadas.

Esto parecía ser un retroceso pero en la realidad la aplicación ya estaba preparada, en gran medida, para realizar estos cambios. Los métodos como `goToPosition` o `shotToPosition` fueron de mucha utilidad para realizar adecuaciones, ejemplo de un caso es cuando el jugador II enviaba un pase al jugador IV de la posición 22 a la 13, en una fase de práctica, no había mayor problema, pero una vez incorporado el oponente, en un porcentaje alto era interceptado, se modificó simplemente dos números en dos estados, pasar de 13 a 10. En otras palabras se estaba cumpliendo parte de los objetivos de diseño, la herramienta debe proveer la capacidad de realizar ajustes a las estrategias definidas y verificar su ejecución de forma simple. Esto dejo como experiencia que siempre hay que tomar en cuanta al oponente, al adversario o en otros entornos a las entidades externas (clientes, proveedores, competencia o efectos adversos de la naturaleza).

5.2. RESULTADOS

El fútbol robótico, alentado por organizaciones como RoboCup, provee el ambiente propicio para el desarrollo de tecnologías o metodologías, ya sea para comprobarlas, discutirlas, relacionarlas, en fin, avanzar de forma tecnológica y en muchos otros casos, como humanidad. De forma más puntual permite la evaluación y prueba del desempeño de los agentes tanto de forma individual como en grupo, compitiendo contra otros agentes desconocidos en un medio incierto.

5.2.1. RESULTADOS GENERALES DE TABA

De forma general la arquitectura para la definición y selección de estrategias de juego para un equipo de agentes TABA (Team Agent Behavior Architecture) ha logrado satisfacer la hipótesis inicial definida en la Sección 1.4 y que se cita textualmente a continuación:

> *"El diseñar e implementar una arquitectura para la selección y ejecución de estrategias de juego, aplicada en fútbol robótico, que proveerá capacidad a los agentes de trabajar en forma colaborativa y coordinada, para llevar a cabo acciones ofensivas y defensivas que maximicen su desempeño individual y en equipo."*

El equipo que usa TABA es capaz de responder a situaciones previamente analizadas así como situaciones inesperadas, realizando acciones coordinadas y colaboradoras que aumenta la probabilidad de alcanzar los objetivos de diseño.

Las capacidades de TABA se pueden clasificar en sus dos vertientes: fase de diseño (definición de estrategias) y fase de ejecución (selección de estrategias), el detalle de las mismas es de mucha utilidad para comprobar los objetivos de este proyecto de investigación.

Fase de diseño

La arquitectura esta diseñada de forma modular con cuatro etapas diferenciadas de fácil comprensión y ajuste; y puede ser extendida a otros módulos tanto de entrada (a la arquitectura) como de salida.

La definición de una simbología para la descripción de estrategias de fútbol - SSDS (Soccer Strategy Description Symbols) es de mucha utilidad para la definición y visualización de una estrategia antes de ser incorporada a la base de estrategias lo que permite reducir el tiempo de diseño. Esta simbología puede ser usada para referencias futuras y ajustada a otros entornos.

Las acciones que se ejecutan en un juego de entrenamiento o práctica sin oponentes provee de capacidad para dar inicio a la definición de estrategias, pero en un juego contra un equipo de oponentes la experiencia enriquece aun más el diseño de una estrategia, al descubrir aspectos no contemplados anterioremente.

La división del ambiente de juego en regiones es la base que posibilita la definición y adecuación de estrategias con ajustes mínimos. También es un apoyo fundamental en la fase de ejecución al momento de seleccionar una nueva estrategia, que se vale en gran manera de la posición de los jugadores y la pelota.

La definición de una estrategia en dos versiones simplificada y extendida, provee de un diseño más modular y eficiente de la estrategia debido a que primero se encarga de las condiciones y el segundo de la ejecución. Como parte de la versión simplificada el atributo `Type` es de mucha utilidad para la adecuada clasificación de las estrategias y mejora el desempeño de la selección de las mismas.

Fase de ejecución

La función principal de TABA, la selección de estrategias, permite escoger de forma precisa una estrategia de un conjunto de estrategias (base de estrategias) acorde a una situación definida, lo cual reduce el tiempo que los agentes invierten en tomar una decisión.

La arquitectura permite a los agentes la realización de múltiples pases de forma bastante acertada, localización en áreas de oportunidad, evasión de oponentes y disparos a portería.

La aplicación de TABA durante la ejecución de un juego, comprobó que existen factores, condiciones que se pueden abstraer y encapsular en una estrategia, en otras palabras, patrones que se repiten y que se pueden explotar en un futuro. Por ejemplo, cuando hay una oportunidad potencial de anotacion, por lo general la posición de los jugadores, contrarios y compañeros, es la misma, cercanas a la región de portería. De la misma forma, cuando la pelota sale por los extremos izquierdo o derecho de la portería, por efecto del guardameta, las posiciones se repiten.

La definición de interrupciones o condiciones de paro, son de utilidad para realizar transiciones de una estrategia a otra. Ejemplo de esto es cuando se sale la pelota o se anota un gol. Ambas interrupciones funcionan de forma adecuada y rápida, lo cual repercute en el inicio de selección de estrategia.

La condición de paro relacionada al tiempo excedido, maximizó el comportamiento individual de los jugadores al aplicar acciones reactivas una vez que no se había logrado cumplir con alguna etapa de la estrategia. Este incumplimiento se debe a las acciones del equipo oponente o por condiciones propias de los agentes (perdió de vista la pelota, no logró tomarla adecuadamente, el pase no alcanzó su objetivo, entre otras).

5.2.2. EJECUCIÓN DE ESTRATEGIAS SIN OPONENTES

Una vez definida la estrategia, es escrita en archivos de prueba y posteriormente es ejecutaba en el simulador para analizar su ejecución de forma general sin oponentes. Esta etapa es de mucha utilidad para ajustar el tiempo en cada etapa de la estrategia esto es el tiempo en el que un agente esperaba la pelota o demoraba para enviar un pase. Era de esperarse que una ejecución sin oponentes, tiende a ser exitosa, si definimos este éxito, en cuanto a si se pudo realizar un pase o realizar una anotación; aún así, se consideraba las limitantes propias de los agentes, simuladas tomando en consideración la limitaciones de los robots reales, como son la pérdida de visión de la pelota, el ángulo de disparo, potencia del disparo y capacidad para mantener la pelota.

5.2.3. EJECUCIÓN DE ESTRATEGIAS CON OPONENTES

Una vez que la estrategia era probada y lograba alcanzar un porcentaje alto de aceptación es incorporada a la base de estrategias para posteriormente probarse en un juego con oponentes. Generalemente la estrategia debe ser ajustada para contemplar al oponente, pero los ajustes son soportados de acuerdo a cambios en los archivos de tipo *XML*

Se procedió a realizar ejercicios, para evaluar la ejecución de un equipo que implementa TABA contra otros oponentes. El grupo de control seleccionado es el equipo Azul los cuales compitieron contra otros oponentes que aplican red de contratantes, esto para que la competencia no sea del todo injusta. El equipo Rojo, que aplica red de contratantes, aplican comportamientos de tipo reactivo y a su vez, validan la posibilidad de pase y disparo a portería, así como acciones para obtener el balón y defender portería.

Para realizar comparaciones se optó por realizar otro set de competencias, sin nuestro equipo de control (Azul) con TABA, aplicando comportamiento de red de contratantes en ambos equipos. De esta forma se pudo evaluar la eficiencia de TABA. A continuación se indicará las competencias realizadas, en principio entre equipos usando red de contratantes y posteriormente nuestro equipo de control Azul usando TABA contra uno que usa red de contratantes. Se realizaron 100 competencias para cada caso, con lapsos de 10 minutos cada una.

Red de Contratantes VS Red de Contratantes

En el Cuadro 5.1 se presenta las competencias realizadas entre equipos que usan red de contratantes, la información incluye el número de partido, los goles anotados por el equipo Rojo y por el equipo Azul.

TABA VS Red de Contratantes

En el Cuadro 5.2 se presenta las competencias realizadas entre el equipo de control Azul que usa TABA y el equipo Rojo que usan red de contratantes.

Partido	Rojo	Azul	Resultado	Partido	Rojo	Azul	Resultado
1	4	1	E	51	1	4	R
2	0	2	R	52	2	0	A
3	3	0	A	53	0	0	R
4	0	0	R	54	3	2	E
5	2	1	E	55	4	1	R
6	1	3	R	56	5	0	R
7	2	3	A	57	2	3	R
8	2	0	A	58	5	1	A
9	3	2	R	59	0	2	R
10	1	1	R	60	2	2	A
11	2	0	E	61	3	0	E
12	2	1	R	62	2	1	R
13	1	2	R	63	2	2	R
14	0	1	A	64	5	0	E
15	2	3	A	65	1	1	R
16	4	6	A	66	2	1	E
17	2	3	A	67	2	2	R
18	3	0	A	68	2	1	E
19	6	1	R	69	2	2	R
20	1	0	R	70	2	2	E
21	2	1	R	71	4	2	E
22	2	3	R	72	1	3	R
23	2	1	A	73	2	3	A
24	2	2	R	74	0	1	A
25	1	1	E	75	1	9	A
26	3	0	E	76	2	2	A
27	1	1	R	77	2	1	E
28	3	0	E	78	1	2	R
29	1	1	R	79	0	1	A
30	2	1	E	80	1	1	A
31	0	3	R	81	3	2	E
32	2	2	A	82	1	5	R
33	1	2	E	83	3	2	A
34	1	1	A	84	1	0	R
35	1	1	E	85	6	0	R
36	2	2	E	86	4	0	R
37	0	3	E	87	1	2	R
38	3	4	A	88	2	0	A
39	4	1	A	89	1	2	R
40	1	1	R	90	1	2	A
41	3	0	E	91	1	1	A
42	3	3	R	92	4	1	E
43	4	3	E	93	3	1	R
44	1	0	R	94	1	1	R
45	1	1	R	95	0	1	E
46	3	2	E	96	2	2	A
47	1	3	R	97	0	2	E
48	2	3	A	98	2	2	A
49	3	0	A	99	2	1	E
50	1	0	R	100	1	1	R

Cuadro 5.1: Competencia entre Red de Contratantes

Partido	Rojo	Azul	Resultado	Partido	Rojo	Azul	Resultado
1	2	2	E	51	1	2	A
2	1	3	A	52	4	3	R
3	0	2	A	53	0	2	A
4	3	0	R	54	0	3	A
5	2	2	E	55	1	3	A
6	1	1	E	56	2	2	E
7	0	1	A	57	2	2	E
8	0	1	A	58	1	3	A
9	0	2	A	59	2	4	A
10	2	1	R	60	1	2	A
11	0	1	A	61	0	3	A
12	0	1	A	62	7	0	R
13	1	1	E	63	1	4	A
14	0	2	A	64	2	1	R
15	0	2	A	65	2	2	E
16	0	1	A	66	0	1	A
17	1	2	A	67	5	0	R
18	4	1	R	68	4	1	R
19	1	2	A	69	0	1	A
20	0	1	A	70	2	1	R
21	0	0	E	71	0	0	E
22	1	2	A	72	2	2	E
23	2	1	R	73	2	4	A
24	0	1	A	74	3	3	E
25	2	6	A	75	0	1	A
26	1	1	E	76	1	1	E
27	1	4	A	77	2	1	R
28	0	0	E	78	1	2	A
29	1	2	A	79	0	2	A
30	0	1	A	80	0	2	A
31	3	2	R	81	0	2	A
32	1	1	E	82	2	2	E
33	0	4	A	83	2	3	A
34	0	1	A	84	2	2	E
35	3	2	R	85	2	3	A
36	0	4	A	86	2	1	R
37	1	3	A	87	1	5	A
38	3	3	E	88	0	0	E
39	0	1	A	89	0	0	E
40	2	2	E	90	0	0	E
41	0	1	A	91	1	1	E
42	1	2	A	92	0	1	A
43	7	0	R	93	1	2	A
44	0	2	A	94	4	0	R
45	0	1	A	95	0	2	A
46	2	2	E	96	0	3	A
47	0	1	A	97	2	2	E
48	2	2	E	98	1	2	A
49	1	2	A	99	1	3	A
50	2	0	R	100	2	3	A

Cuadro 5.2: Competencia entre TABA (Azul) y Red de Contratantes(Rojo)

El resultado final para el conjunto de competencias entre los equipos que aplican red de contratantese se puede observar en la Figura 5.4 y para el conjunto de competencias entre el equipo de control que usa TABA y el equipo que usa red de contratantes, se puede revisar en la Figura 5.5. Se

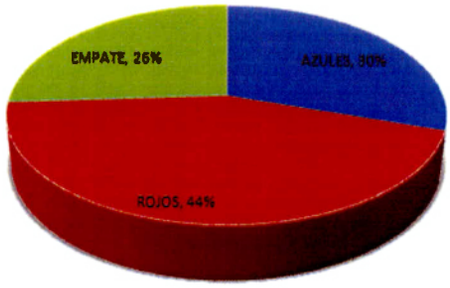


Figura 5.4: Resultado competencia Red de Contratantes

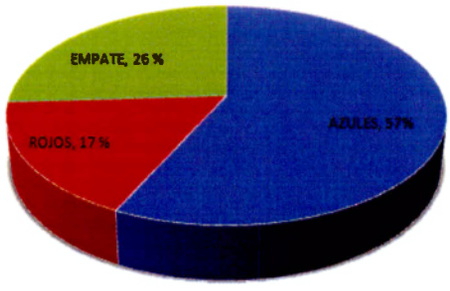


Figura 5.5: Resultado competencia TABA y Red de Contratantes

puede observar que existe una diferencia notable entre el conjunto de competencias; en principio, las pruebas entre equipos que aplican red de contratantes, tienden a ser similares en sus marcadores finales, con un alto porcentaje de empate. Sin embargo, las pruebas realizadas con el equipo que aplica TABA y el equipo de red de contratantes, tiende a tener un marcador positivo para el grupo de control (Azul) y marcadores más bajos para los Rojos ó empate.

En general la aplicación de estrategias permite tener más oportunidades de ganar durante la ejecución de un juego, porque propicia las acciones como despeje, confundir al oponente, aprovechar mejor a los agentes y ser mas dinámico en contraste contra un equipo que no aplica estrategias y que ejecuta las acciones de forma reactiva. Parte de estas experiencias fueron mencionadas en el artículo "Team Agent Behavior Architecture in Robot Soccer" aceptado por el *Latin American Robotic Symposium*, referido en [Ruiz and Uresti, 2008].

5.3. RESUMEN DEL CAPÍTULO

En este capítulo se describió las experiencias en el desarrollo del trabajo de investigación desde la fase de definición del problema, fijación de los objetivos, diseño de la arquitectura, modificaciones al entorno de aplicación (simulador *XML Behaviour Control*), y realización de pruebas.

Así mismo se comentan los principales resultados obtenidos con TABA tanto en la fase de diseño como es la aplicación de una simbología para la definición de estrategias, la división de el terreno de juego en regiones, y en la fase de ejecución, la selección de estrategias, la definición de las condiciones de terminación de la estrategia.

6. CONCLUSIONES Y TRABAJO FUTURO

"Estrategia sin tácticas es la ruta más lenta hacia la victoria. Tácticas sin estrategia es el ruido antes de la derrota"
Sun Tzu

TABA es una aplicación de metodologías y tecnologías de sistemas multiagentes, robótica e inteligencia artificial en el proceso de toma de decisión, explicitamente, "razonar" por medio de "recordar", y tiene como principio, aplicar procesos sencillos que sean fáciles de modificar y aplicar en diversos entornos. El ambiente de prueba de esta arquitectura es un entorno competitivo basado en un juego de fútbol robótico con reglas definidas por la organización RoboCup en la categoría estándar para cuadrúpedos.

La orientación del presente trabajo de investigación ha sido aplicar métodos y procesos análogos a la forma en la que el ser humano actúa sobre todo en aspectos relacionados a la toma de decisión, para aplicarlas en un entorno de juego relacionado al fútbol, elementos como la representación de una jugada basada en símbolos, que consecuentemente es llamado *caso* o *situación*; el análisis de vídeos, de competencias anteriores, para determinar potenciales estrategias de juego; la división del terreno de juego en áreas de diferente tamaño de acuerdo al objetivo de cada sector; aplicación de procesos de selección basados en distancia simple entre objetos de acuerdo a la situación de estado del mundo; aplicación de un agente líder similar a la selección de un capitán en un equipo entre otros aspectos.

A continuación se detallarán las principales conclusiones que se derivan del trabajo de investigación así como la propuesta para trabajos futuros.

6.1. CONCLUSIONES

En este trabajo de investigación se introduce una arquitectura para la definición y selección de estrategias de juego, TABA, que se basa en el concepto de estrategias como un plan de acción, el cual es ejecutado por un equipo de agentes de forma coordinada y colaborativa y los capacita para alcanzar los objetivos de diseño. Estas estrategias son fácilmente representadas por medio de una simbología *SSDS* que hace uso de lenguaje de juegos para su descripción y posteriormente son incorporadas a un entorno de simulación para su evaluación.

6.1.1. VENTAJAS DE TABA

La arquitectura esta diseñada de forma modular y su comprensión es intuitiva; está compuesta por cuatro etapas principales: (i) selección del líder, (ii) selección de la estrategia, (iii) asignación de roles y (iv) ejecución y terminación de estrategia, que se relacionan entre sí y con el medio. Esto permite obtener un mejor rendimiento ante los cambios del ambiente.

La incorporación de una simbología para la definición de estrategias de fútbol SSDS redujo el tiempo de diseño de cada estrategia y propició la creación de nuevas a partir de la modificación de las ya definidas. Así mismo, permitió la comprensión de las etapas que componen la estrategia de forma intuitiva.

La aplicación permite almacenar un número de estrategias en una base de estrategias que se clasifican, de forma general, por medio del atributo `type` que posteriormente es consultado en el proceso de selección de estrategias. De esta forma la aplicación de estrategias de tipo ofensiva, defensiva y de patada inicial permite proveer a los agentes de un curso de acción ajustado a la situación en un momento dado.

El razonamiento basado en casos parte de la premisa de "razonar" por medio de "recordar". Ha sido implementado en TABA como mecanismo de apoyo en la toma de decisiones. El algoritmo de selección de estrategias aplica la técnica del *K-vecino más cercano* que propone un curso de acción ajustado a la situación en un momento dado. El tiempo de procesamiento es rápido y no ocaciona entorpecimiento a otra actividad que estén ejecutando los agentes. Lo más destacado, es que este proceso analiza un conjunto de factores a la vez, considerando pesos o importancias a determinados elementos. Esta metodología impulsa el proceso de toma de decisiones con base en una *similitud* a la forma que lo ejecuta el ser humano.

A medida que se obtiene un conocimiento acertado del entorno los agentes están mejor preparados para responder ante eventualidades. TABA es independiente del dominio de aplicación, puesto que no depende de un dominio especifico (aunque lo afecta) pero la complejidad de implementación de TABA depende de la complejidad del dominio de aplicación.

6.1.2. LIMITACIONES DE TABA

El comportamiento de los agentes contrarios no es considerado al momento de la selección de estrategias.

El desarrollo de comportamientos coordinados requiere de un conocimiento de las capacidades de los agentes y la consideración de los oponentes.

6.2. TRABAJO FUTURO

Los sistemas multiagentes, la robótica y la inteligencia artificial han revolucionado la forma en la que el ser humano interactúa con el entorno que lo rodea, los agentes de software y robóticos están presentes en muchas operaciones cotidianas y no tan cotidianas, desde una búsqueda en internet, pasando por operaciones de negocio, entretenimiento, simulación hasta operaciones de búsqueda y rescate. Es comprensible identificarlos, a los agentes, como una extensión, en algunos casos, del ser humano e incluso la ciencia de la información trabaja día con día para humanizarlos, aplicando diversas técnicas tanto de *hardware* como de *software* para asemejarlos, lo más posible, a un ser humano, tanto *física* como *mentalmente*.

En la sección anterior se analizaron los aspectos sobresalientes del trabajo de investigación realizado, así como las ventajas de aplicar la arquitectura para la generación y selección de estrategias. En este apartado se expondrán la potencialidad de la continuación de esta investigación, entre los que destaca la incorporación de otras metodologías y tecnologías a la arquitectura, así como la aplicación de la misma en otros entornos en especial en plataformas reales.

6.2.1. ACTUALIZACIÓN Y MEJORA DE LA APLICACIÓN

En los capítulos anteriores se definió la arquitectura de estrategias, la plataforma de desarrollo, la definición del entorno virtual y las experiencias y resultados obtenidos. De esto se desprende el análisis de limitantes que pueden ser visto como oportunidades para futuras implementaciones. Algunas de estas son:

- **Archivos XML**: actualmente los agentes son creados con el conocimiento completo de todas las estrategias aplicables, sería de mucha utilidad dividir la información en espacios de memoria distinto emulando parte del sistema nervioso humano (memoria a corto plazo, memoria a largo plazo), es decir, que el agente puede hacer uso de comportamientos sencillos que sean recurrente de forma mas ágil e indagar en la "memoria" aquellos procesos que sean mas complejos, en otras palabras, realizar la división de comportamientos en diferentes archivos de tipo *XML* que posibilite aún más el diseño, adaptación y aplicación con el menor esfuerzo por parte del programador.

- **Retroalimentación**: los reportes y los archivos de salida son una forma de retroalimentación que permite analizar la efectividad de la aplicación, y permite explotar otras posibilidades

que no se habían observado anteriormente. Un ejemplo de reporte sería el análisis de las situaciones que entraron al proceso de selección de estrategias y no encontraron una coincidencia lo suficientemente aceptable, según el umbral de tolerancia definido en la variable `KNN_THRESHOLD` de la clase `KNN_CONFIG`, con esta información se podría realizar un proceso de minería de datos que permita "descubrir" algo, como por ejemplo, potenciales jugadas.

- **División por cuadrantes**: la división del entorno de juego en regiones o cuadrantes, es de vital importancia en el proceso de selección de una nueva estrategia. Esto se realiza en dependencia de las dimensiones del entorno y es un proceso que se ejecuta una sola vez al inicio de la aplicación. Una interesante aplicación sería que esta división fuese mas "intuitiva" para los jugadores, es decir que ellos realicen un reconocimiento previo del entorno y posteriormente lo dividan en áreas que consideren críticas. Esto podría aplicar a un entorno que fuese más dinámico como en el caso de robots que se usen en una operación de búsqueda y rescate.

- **Tipos de estrategia**: ampliar la clasificación de estrategias definidos actualmente en defensivos, ofensivos y de patada inicial a otros más específicos como por ejemplo de bloqueo, de despeje, de evasión, patada inicial - ofensiva, patada inicial - defensiva. Esto podría sentar las bases para la definición de estrategias menos complejas que involucren a menos agentes, y que la duración sea más corta, de esta forma el control sobre estas estrategias se vería más preciso y las adecuaciones requerirían de menos cambios.

- **Más jugadores**: realizar pruebas con más jugadores. A pesar que el simulador *XML Behaviour Control* esta diseñado para especificar *n* número de agentes por medio del archivo `environment.xml`, el proceso de selección de estrategias está limitado a una cantidad definida de atributos, entre estos, cuatro jugadores. A pesar de esto, la codificación se realizó de tal forma que sea fácil la incorporación de más atributos y el algoritmo que analiza la similitud entre casos responde a la actualización de estos atributos.

- **Entorno Virtual**: actualizar el entorno virtual a las especificaciones de las reglas para la categoría estándar - cuadrúpedos de RoboCup en las versiones posteriores al 2007. Actualmente la categoría enfrenta una transición a humanoides y a la fecha la organización ha especificado que las reglas versión 2008 será la última actualización para la liga cuadrúpedos, ver enlace en[AiboSoccer, 2007]. Entre los cambios fundamentales de la nuevas reglas es aplicar un juego 5 *versus* 5 robots de tipo AIBO en una campo de juego de dimensiones superiores a la versión 2007 y con apoyo de identificadores de esquina con los mismos colores aplicados a portería y a postes. En la Figura 6.1 se muestra la definición del entorno virtual con base en las reglas AIBO Soccer 2008 definidas en el sitio oficial de Liga de Plataforma Estándar de RoboCup.

Figura 6.1: Reglas cuadrúpedos 2008

6.2.2. GENERACIÓN DE ESTRATEGIAS

En la actualidad la generación de estrategias es un proceso semi-autónomo, es decir es un proceso que interviene directamente el programador con la ventaja que la aplicación de acciones y eventos reducen en gran medida el tiempo de desarrollo. Se propone la implementación de dos formas para la generación de la bases de estrategias.

La primera es modificar el entorno virtual quizás de un modelo 2D a uno 3D, para ser lo más amigable y realista posible, pero que a su vez tenga la capacidad de realizar la definición de estrategias fuera de línea, es decir que la herramienta defina un momento de práctica en el cual se puedan definir por cada estrategias los atributos correspondientes entre estos:

- datos generales: tipo de estrategia (ofensiva, defensiva, patada inicial), número de jugadores, número de momentos, tiempo global de la estrategia, posición inicial de jugadores.
- posición por momento: registro de la posición (deseada) de cada jugador y pelota en cada momento definido en los datos generales.
- acción por momento: registro de la acción que ejecutará cada jugador en cada momento definido en los datos generales.

Además que pueda definirse esta información implementando mejoras en la aplicación como por ejemplo usar mecanismo de "arrastrar y soltar", en voz inglesa *drag and drop* que permita mover un objeto a una posición deseada y registrar posiciones en cada *momento* de la estrategia en espacios de tiempo definidos. Por ejemplo, luego de especificar la posición inicial de los agentes (1er momento), registrar la siguiente posición (deseable) de todos los jugadores (2do momento) y así sucesivamente hasta completar la definición de estrategia. Para este tipo definición convendría además contar con el apoyo de un experto, un entrenador de fútbol, quien pueda proveer sus comentarios sobre la mejor formación en dependencia del número de jugadores que participen o las condiciones del ambiente.

La segunda es aplicar metodologías y tecnologías de máquinas de aprendizaje para la generación de estrategias que implementen la adquisición de conocimiento, por ejemplo, por medio de aprendizaje por refuerzo, para el conjunto de agentes que participan en el juego. Un ejemplo de esto es al momento de generar la base de estrategias identificar si el caso de entrada es válido o

se propone para agregarse a la base de casos o actualizar un caso existente. De esta forma esta base de estrategias se aplicaría como entrada a la arquitectura de estrategias y el proceso general continuaría de la misma forma que la ya establecida.

6.2.3. RBC Y OTRAS METODOLOGÍAS

Este proyecto de investigación hizo uso de la metodologías de Razonamiento Basado en Casos para realizar la función principal de definir, generar y seleccionar el comportamiento de un equipo de jugadores en fútbol robótico, quienes lo ejecutan con base en un proceso de búsqueda por similitud a una experiencia previa. Se propone la implementación de algunas modificaciones con el fin de identificar áreas de oportunidad no explotadas aún. Cabe resaltar que no es objetivo de esta investigación explotar todas las capacidades de la metodología de RBC si no mas bien, aplicar parte de sus funciones para mantener una complejidad baja en el procesamiento.

- **Representación de casos**: actualmente un caso se define bajo una clasificación de siete atributos que incluyen tipo de estrategia, posición de cuatro jugadores (propios), posición de la pelota e identificador de la estrategia. Esta representación podría explotar aún más los beneficios de la metodología si dentro de la definición de casos se incluyera la posición de los jugadores contrarios. En este momento la posición aproximada de los jugadores contrarios sí es analizada, pero está información se usa para realizar un análisis de riesgo, que permite identificar la orientación de la jugada (ofensiva o pasiva) al momento de buscar una estrategia acorde en la base de estrategias.

  Esta claro que en un medio dinámico y bajo incertidumbre como el fútbol esto sería una información de poca certeza pero a medida que los procesamientos y tecnologías se mejoran también se pueden proporcionar mecanismos para asegurar y validar los datos y posteriormente, ya validados, entrar a la maquinaria de selección.

- **Indexación de casos**: se puede mejorar el proceso de indexación de uno semi-manual basado en árboles a otros métodos por ejemplo modelos bayesianos que apliquen probabilidad de ocurrencia a cada caso.

- **Selección de casos**: la maquinaria de selección se aplico bajo el principio de *K-vecino más cercano* aplicando como medida de distancia la *euclidiana*, queda como parte de futuros trabajos aplicar otras funciones de distancia como por ejemplo de *Hamming*, concebida originalmente para la detección y corrección de errores en comunicación digital y utilizado en el campo de la IA en espacios grandes de trabajo.

- **Adaptación y mantenimiento**: son dos aspectos que no están ampliamente usados en esta aplicación y podrían definirse como trabajo futuro para el mejoramiento contínuo de la base de estrategias y seguridad en la selección de casos. Además se propone la aplicación de memoria de casos como mecanismo evolutivo mediante el cual se le puede otorgar una recompensa aquella estrategia que fue exitosa.

6.2.4. ENTORNOS DE APLICACIÓN

La arquitectura de estrategias TABA ha sido una aproximación en la aplicación de metodologías y tecnologías de los sistemas multiagentes, robótica e inteligencia artificial para satisfacer, en ciertos casos, objetivos generales *"aproximación al comportamiento humano"* y específicos *"proveer a los robots de capacidad de trabajar en forma colaborativa"*. A pesar que el entorno de aplicación se desarrollo sobre una plataforma de competencia, específicamente, fútbol robótico, esta puede implementarse, con determinados ajustes en otros entornos.

La aplicación de trabajo de colaboración en sistemas multiagentes es extenso, por tal razón, se propone mas no se limita, la implementación de la arquitectura, en los proyectos de investigación que se llevan a cabo en la Cátedra Agentes Virtuales y Robóticos en Ambientes de Realidad Dual del Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México. A continuación se presenta algunas opciones de trabajo de tipo colaborador entre robots.

Robots de Rescate

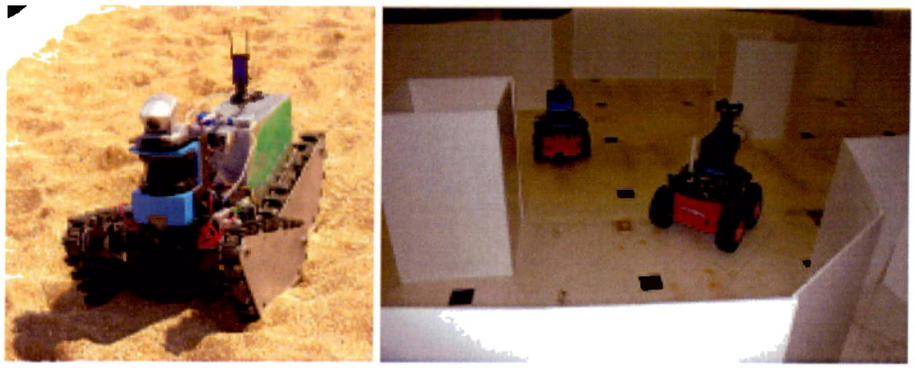
Los robots de rescate han sido diseñados para apoyar a los trabajadores que desempeñan alguna función social, bomberos, socorristas, policías entre otros, en [Schlenoff and Messina, 2005] se mencionan algunas de las fases importantes en las operaciones de búsqueda y rescate, algunas se llevan a cabo en forma secuencial otras en paralelo, entre éstas, se puede destacar: reconocimiento, búsqueda primaria, evaluación estructural, estabilización, asistencia médica, rescate, análisis contínuo, tratamiento de material nocivo, entre otros.

Uno de los primeros usos de robots en operaciones de rescate y búsqueda se llevó a cabo en el World Trade Center tras el atentado a las torres gemelas en la ciudad de Nueva York el 11 de septiembre del 2001, alrededor de más de una docena de robots administrados, en su mayoría a control remoto, hacían uso de luces, cámaras de vídeo y sensores, en búsqueda de víctimas, en lugares donde los socorristas no podían o no se atrevían a ir, tal como se reporta en el periódico New York Times en [Lee, 2001].

Estos enviaban las imágenes a los trabajadores de rescate por medio de cables o de forma inalámbrica y así determinar qué áreas en los escombros merecían exploración adicional y si se podría llevar a cabo sin peligro. Desafortunadamente no se pudieron encontrar víctimas aún con vida pero la aplicación fue de mucha utilidad y los socorristas afirmaron que la colaboración de estos robots fue vital.

En el ITESM-CEM se llevan a cabo trabajos de diseño y programación de robots de rescate, inicialmente como un desafío propuesto por diferentes organizaciones que impulsan el espíritu creativo e innovación así como la colaboración entre investigadores y estudiantes de las diferentes disciplinas relacionadas a la robótica e inteligencia artificial, entre estos RoboCup.

En el reporte técnico del equipo de rescate disponible en [RescueTeam, 2008] se detalla el desarrollo del robot de rescate llamado *Kauil* que tiene como objetivo aplicar y desarrollar algoritmos para la navegación autónoma, reconocimiento de terreno, *SLAM* e identificación de víctimas. Si bien, en la actualidad sólo se cuenta con un robot de este tipo, en el campus ya se han realizado trabajos sobre implementación navegación autónoma de robots de tipo *pioneers*, en los cuales es inherente la aplicación de mecanismos de coordinación, que actualmente no se poseen. En la Figura 6.2 se muestran las imágenes del robot de rescate *Kauil* y los robots *pioneers*. La arquitectura



(a) Kauil (b) Pioneers

Figura 6.2: Robot de Rescate y Robots de navegación autónoma

de estrategias para este entorno de aplicación sería de mucha utilidad para asistir a un equipo de rescate y búsqueda. Por ejemplo en un incendio dentro de un edificio de oficinas, un equipo de robots podrían colaborar para identificar a las víctimas, definir las áreas críticas y generar planes de acción para el rescate de las víctimas. Inicialmente se escogería un líder quien podría tomar decisiones sobre que recursos (cantidad y tipo de robot) enviar a determinada actividad.

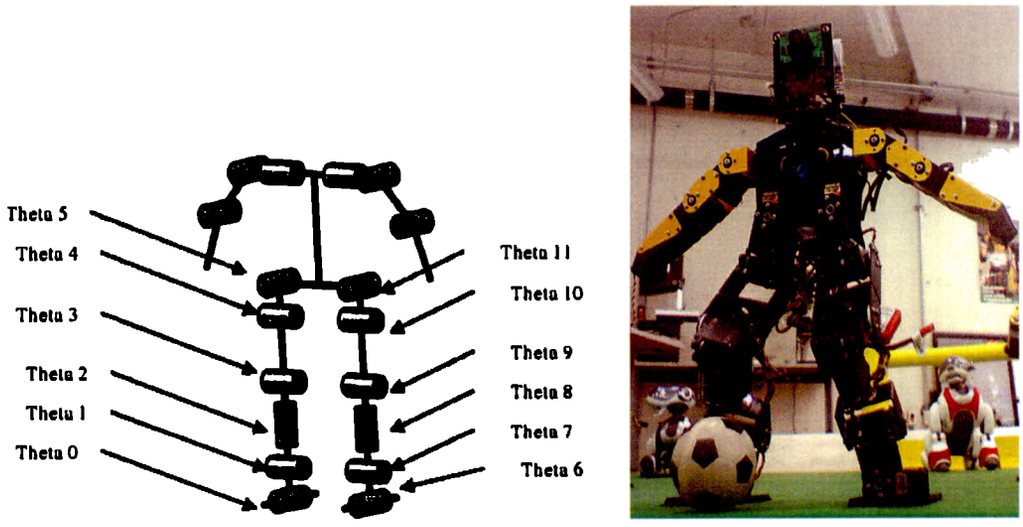
Los robots, supuniendo que tienen a su disposición mapas del edificio, recorrerían los pisos o secciones del edificio y enviarían la información recabada para su procesamiento. Tomando en cuenta que son robots de rescate y búsqueda, estos pueden contener una base de planes de acción que se ajusten a determinadas situaciones, por ejemplo, si una víctima se encuentra cerca de un conducto de gas, o cerca de escaleras, que víctimas requieren atención de inmediato. Una vez que se define la situación entrará a un proceso de selección (calculo de similitud) para definir el mejor curso de acción para el equipo de robots. Posteriormente la ejecución de la actividad y condiciones de paro.

Robots Humanoides

Un robot humanoide es en general un robot diseñado con apariencia humana, es decir, posee un torso, cabeza, dos piernas y dos brazos. Esto en sí, es un desafío de diseño mecánico, electrónico, eléctrico entre otras disciplinas, pero además no solo basta con parecer humano si no comportarse como tal, en otras palabras ser autónomo y que sus acciones respondan a los cambios en el entorno que lo rodea, así mismo, relacionarse con otros robots similares e incluso con el humano mismo.

De la misma forma que la sección anterior, en el ITESM-CEM, ya se han iniciado trabajos que implican el diseño, construcción y programación de robots humanoides que participan en la competencia de fútbol robótico en la categoría de *Humanoide - Kid Size* para la organización internacional RoboCup. Como parte de las definición de reglas de esta categoría dos equipos de tres robots cada uno compiten por obtener la mayor anotación posible.

Actualmente el equipo de humanoide ha desarrollado con éxito los mecanismos relacionados a la locomoción de estos robots como es la cinemática inversa, parámetros de caminado que permiten la ejecución de movimientos de forma omni-direccional y la implementación de 6 grados de libertad, en voz inglesa *Degrees of freedom DOF*, en cada pierna incluyendo un grado de libertad en el eje Z (eje vertical) justo debajo de la articulación de la rodilla, emulando la rotación natural de la tibia en el humano, tal como se refiere en el reporte técnico disponible en [HumanoidTeam, 2008]. Es necesario aplicar algoritmos para la toma decisión en los robots que capaciten a los robot para trabajar de forma coordinada y colaboradora entre ellos. En la Figura 6.3 se muestra el trabajo desarrollado por el equipo Humanoide.



(a) Descripción de grados de libertad (b) Humanoide Kid-Size

Figura 6.3: Robot Humanoide

El equipo de robots humanoides podrían aplicar TABA para la selección de estrategias de juego similar a la implementada en este trabajo de investigación.

BIBLIOGRAFÍA

[AIBO, 2007] AIBO (2007). Aibo web site. http://www.sonystyle.com/intershoproot/eCS/Store/en/imagesProducts/ProductTour/computing/ers7/tour01.html.

[AiboSoccer, 2007] AiboSoccer (2007). Four legged league web site. http://www.tzi.de/4legged/bin/view/Website/WebHome.

[Alpaydin, 2004] Alpaydin, E. (2004). *Introduction to Machine Learning*. The MIT Press, Cambridge, Massachusetts.

[Balaraman, 2004] Balaraman, V. (2004). An evolutionary computation perspective. In Menon, A., editor, *Frontiers of Evolutionary Computation*, pages 211–241, Pittsburgh, Pennsylvania, USA. Kluwer Academic Publishers. ISBN 1-4020-7524-3.

[Bellman, 1978] Bellman, E. (1978). *An Introduction to Artificial intelligence : Can Computers think?* Boyd/Fraser Publishing Company, San Francisco.

[Bowling et al., 2004] Bowling, M., Browning, B., Chang, A., and Veloso, M. (2004). Plays as team plans for coordination and adaptation. In *RoboCup 2003: Robot Soccer World Cup VII*, pages 686–693. Springer Berlin / Heidelberg.

[Britannica, 2008] Britannica (2008). Tactics. http://0-search.eb.com.millenium.itesm.mx:80/eb/article-53028 (25/03/2008).

[Browning et al., 2005] Browning, B., Bruce, J., Bowling, M., and Veloso, M. (2005). Stp: Skills, tactics and plays for multi-robot control in adversarial environments. In *IEEE Journal of Control and Systems Engineering*, pages 33–52. IEEE.

[Buckland, 2004] Buckland, M. (2004). In *Programming Game AI by Example*, pages 133–191, Texas, USA. Wordware Publishing.

[Bunge, 1968] Bunge, M. (1968). Analogy in quantum theory: From insight to nonsense. *The British Journal for the Phylosophy of Science*, 18(4):265–286.

[Ferber, 1999] Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[FIRA, 2008] FIRA (2008). Sitio web de *FIRA*. http://www.fira.net/ (05/04/2008).

[GermanTeam, 2008] GermanTeam (2008). Sitio oficial germanteam. http://www.germanteam.org/tiki-index.php (01/04/2008).

[Groen et al., 2002] Groen, F., Spaan, M., and Vlassis, N. (2002). Robot soccer: Game or science. In *Proceedings CNR2002*.

[HumanoidTeam, 2008] HumanoidTeam (2008). Development and implementation of omni-directional locomotion in a humanoid robot with tibial rotation. (01/05/2008).

[Humboldt, 2007] Humboldt (2007). Team description 2007. http://www.robocup.de/AT-Humboldt (10/03/2008).

[Kecman, 2001] Kecman, V. (2001). *Learning and Soft Computing: support vector machines, neural networks, and fuzzy logic models*. The MIT Press, Cambridge, Massachusetts.

[Kimball and Holyoak, 2000] Kimball, D. L. and Holyoak, K. J. (2000). Transfer and expertise. In Tulving, E. and Craik, F. I., editors, *Handbook of Memory*, pages 109–123, New York, USA. Oxford University Press.

[Labrou and Finin, 2000] Labrou, Y. K. and Finin, T. (2000). History, state of the art and challenges for agent communication languages. *Informatik*, 7(1):17–24.

[Lee, 2001] Lee, J. (2001). Agile in a crisis, robots show their mettle. http://nytimes.com/ (10/01/2008).

[Lötzsch et al., 2004] Lötzsch, M., Bach, J., Burkhard, H.-D., and Jüngel, M. (2004). Designing agent behavior with the extensible agent behavior specification language xabsl. In *RoboCup 2003: Robot Soccer World Cup VII*, pages 114–124. Springer Berlin / Heidelberg.

[Malec, 2000] Malec, J. (2000). Implementing teams of agents playing simulated robot soccer. http://citeseer.ist.psu.edu/malec00implementing.html (15/10/2007).

[Marling et al., 2003] Marling, C., Tomko, M., Gillen, M., Alexander, D., and Chelberg, D. (2003). Case-based reasoning for planning and world modeling in the robocup small size league. In *IJCAI Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments*.

[Meng et al., 2000] Meng, Q., Zhuang, X., Zhou, C., Xiong, J., Y.Wang, T.Wang, and Yin, B. (2000). Game strategy based on fuzzy logic for soccer robots. In *IEEE Int. Conf. Systems, Man, Cybernetics*, pages 3758–3763. RoboCup.

[Oxford, 2002] Oxford (2002). Strategy. http://www.oxfordreference.com (25/03/2008).

[Oxford, 2003] Oxford (2003). Strategy. http://www.oxfordreference.com (25/03/2008).

[Pal et al., 2001] Pal, S. K., Dillon, T., and Yeung (eds), D. S. (2001). Soft computing in case based reasoning. pages 1–28. Springer-Verlag.

[Pal and Shiu, 2004] Pal, S. K. and Shiu, S. C. K. (2004). *Foundations of Soft Case-Based Reasoning*. John Wiley & Sons, Inc.

[Raymond, 2006] Raymond, S. (2006). *Fuzzy-Neuro Approach to Agent Applications from the AI Perspective to Modern Ontology*. Springer Berlin Heidelberg, New York.

[RescueTeam, 2008] RescueTeam (2008). Sitio oficial verlab. http://verlab.cem.itesm.mx/cnieto/Robocup2008/ (01/05/2008).

[Reyes, 2008] Reyes, C. (2008). Generación de comportamientos colaborativos utilizando aprendizaje por refuerzo en un medio de agentes de fútbol. Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey.

[RoboCup, 2008] RoboCup (2008). Sitio oficial robocup. http://www.robocup.org/ (01/04/2008).

[Ross, 2004] Ross, T. J. (2004). *Fuzzy Logic with Engineering Applications*. John Wiley and Sons, The Attrium, Southern Gate, Chichester, West Sussex, England, 2nd edition.

[Ruiz and Uresti, 2008] Ruiz, M. A. and Uresti, J. R. (2008). Team agent behavior architecture in robot soccer. *Robotic Symposium, IEEE Latin American*, 0:20–25.

[Russell and Norvig, 2003] Russell, S. J. and Norvig, P. (2003). *Artificial intelligence : a modern approach*. Prentice Hall/Pearson Education, New York, 2nd edition.

[Schlenoff and Messina, 2005] Schlenoff, C. and Messina, E. (2005). A robot ontology for urban search and rescue. In *KRAS '05: Proceedings of the 2005 ACM workshop on Research in knowledge representation for autonomous systems*, pages 27–34, New York, NY, USA. ACM.

[Sng et al., 2002] Sng, H. L., Gupta, G. S., and Messom, C. H. (2002). Strategy for collaboration in robot soccer. In *DELTA '02: Proceedings of the The First IEEE International Workshop on Electronic Design, Test and Applications (DELTA '02)*, page 347, Washington, DC, USA. IEEE Computer Society.

[Stone, 2000] Stone, P. (2000). *Layered Learning in Multiagent System: A winning approach to robotic*. The MIT Press.

[Tapia et al., 2006] Tapia, J. J., Reyes-Rico, C., and Ramirez, J. (2006). Automatic behavior generation in a multi-agent system through evolutionary programming. In *Third IEEE Latin American Robotics Symposium*.

[TecRams, 2008] TecRams (2008). Sitio oficial tecrams. http://www.cem.itesm.mx/robocup/ (01/04/2008).

[Vega, 2005] Vega, J. L. (2005). Definición de estrategias de robots jugadores de fútbol a través de xml. Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey.

[Vega et al., 2006] Vega, J. L., Junco, A., and Ramírez, J. (2006). Major behavior definition of football agents through XML. In Arai, T., Pfeifer, R., Balch, T., and Yokoi, H., editors, *Intelligent Autonomous Systems 9 (IAS-9)*, pages 668–675, Tokyo, Japan. IOS Press. ISBN 1-58603-595-9.

[Watson, 1997] Watson, I. D. (1997). *Applying case-based reasoning: techniques for enterprise systems*. Morgan Kauffmann, San Francisco, CA, USA.

[Weiss, 1999a] Weiss, G. (1999a). Multiagent systems: A modern approach to distributed modern approach to artificial intelligence. pages 27–66. The MIT Press, London.

[Weiss, 1999b] Weiss, G. (1999b). Multiagent systems: A modern approach to distributed modern approach to artificial intelligence. pages 1–9. The MIT Press, London.

[Weiss, 1999c] Weiss, G. (1999c). Multiagent systems: A modern approach to distributed modern approach to artificial intelligence. pages 83–111. The MIT Press, London.

[Wendler and Lenz, 1998] Wendler, J. and Lenz, M. (1998). Cbr for dynamic situation assessment in an agent-oriented setting.

[Wooldridge, 2001] Wooldridge, M. J. (2001). *Introduction to Multiagent Systems*. John Wiley & Sons, Inc.

[Wooldridge and Jennings, 1995] Wooldridge, M. J. and Jennings, N. R. (1995). *Intelligent agents: Theory and practice*, volume 10. Cambridge University Press.

[Ye and Churchill, 2003] Ye, Y. and Churchill, E. (2003). In Ye, Y. and Churchill, E., editors, *Agent Supported Cooperative Work- An Introduction*, pages 1–25. Kluwer Academic Publishers, Norwell, Masachusetts, USA. ISBN 1-4020-7404-2.

APÉNDICE A. LISTA DE ACRÓNIMOS

ABG	: Automatic Behavior Generation
AF	: Autómata Finito
BADRL	: Behavior Agent Definition by Reinforcement Learning
CBR	: Case Based Reasoning
FIFA	: Fédération Internationale de Football Association
FIRA	: Federation of International Robot Soccer Association
FPS	: Fieldplayer States
GKS	: Goalkeeper States
GT2004	: GermanTeam versión 2004
IAD	: Inteligencia Artificial Distribuida
IDE	: Integrated Development Environment
ITESM-CEM	: Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de Méxic
KIF	: Knowledge Interchange Format
KQML	: Knowledge Query and Manipulation Language
NASA	: National Aeronautics and Space Administration
RBC	: Razonamiento Basado en Casos
RoboCup	: Robot World Cup Soccer Games and Conferences
SB	: Strategy Base
SMA	: Sistema Multiagente
SSDS	: Soccer Strategy Description Symbols
TABA	: Team Agent Behavior Architecture
XABSL	: Extensible Agent Behavior Specification Language

B.1.2. BALLENDLINE

La Figura B.2 muestra parte del método que verifica si un la pelota ha sido enviada fuera de la cancha.

```
private void checkBallEndLine()
{
        Ball b = Environment.getBall();
        double ballx = b.getDoubleX();
        double bally = b.getDoubleY();
        double radio = b.getRadio();

        if(ballx - radio >Environment.getFootballSpace().getFieldwidth()/2)
        {
                if(ballPossession==AiboRobot.RED)//la saco la defensiva
                {
                        if (bally >0)
                                (...)
                        else if(bally <0)
                                (...)
                }
                else if(ballPossession==AiboRobot.BLUE)//la saco la ofensiva
                {
                        if (bally >0)
                                (...)
                        else if(bally <0)
                                (...)
                }
        }
}
```

Figura B.2: Ejemplo pelota fuera

B.1.3. ESTADOS DE *GOALKEEPER*

La Figura B.3 muestra parte del comportamiento del portero.

```
<state name="getLeadership">
        <action id="COMM-getLeaderRole"/>
        <transition state="defendGoal">
        </transition>
</state>
<state name="searchBall">
        <action id="MOTION-doNothing"/>
        <action id="COMPLEX-headToBall"/>
        <transition state="volteaDerecha">
                <event name="VISION-ballAtRight"></event>
        </transition>
</state>
<state name="blockBall">
        <action id="MOTION-doNothing"/>
        <transition state="leftBlock">
                <event name="VISION-goalAtFront"></event>
                <event name="VISION-ballAtLeft"></event>
        </transition>
                <transition state="rightBlock">
                <event name="VISION-goalAtFront"></event>
                <event name="VISION-ballAtRight"></event>
        </transition>
</state>
```

Figura B.3: Ejemplo estados del *goalKeeper*

B.1.4. ESTADOS DE *FIELDPLAYER*

La Figura B.4 muestra parte del comportamiento de los jugadores de campo.

```
<state name="getLeadership">
        <action id="COMM-getLeaderRole"/>
        <transition state="defendGoal">
        </transition>
</state>
<state name="searchBall">
        <action id="MOTION-doNothing"/>
        <action id="COMPLEX-headToBall"/>
        <transition state="volteaDerecha">
                <event name="VISION-ballAtRight"></event>
        </transition>
</state>
<state name="goToPassPosition" iterate="true">
        <action id="COMPLEX-goToQuadrantPosition" param1="19" param2="0"/>
        <action id="COMPLEX-headSearch"/>
        <transition state="observarBalon">
                <boleanEvent name="VISION-ballFound" operator="not"/>
                <analogEvent name="timer" value="3000" operator="greater"/>
        </transition>
</state>
```

Figura B.4: Ejemplo estados del *fieldPlayer*

APÉNDICE C. MANUAL DE USUARIO

El simulador de jugadores de fútbol a través de XML junto a la aplicación de arquitectura para la generación y selección de estrategias es una poderosa herramienta para probar comportamientos de agentes de forma coordinada en un ambiente de simulación, que tiene por objetivo reducir el tiempo de desarrollo en la aplicación en un entorno real, esto es, probar la efectividad de una estrategia y realizar ajustes correspondientes para obtener un comportamiento eficiente.

Así mismo, la arquitectura para la generación y selección de estrategias permite identificar las oportunidades que se presentan durante el juego y de acuerdo a una situación específica, tomar una decisión sobre el proceder, como equipo, en un momento dado.

El presente documento tiene por objetivo identificar los principales aspectos a considerar para generar un ambiente de prueba adecuado en la aplicación de la arquitectura para la generación y selección de estrategias. Es importante destacar que la arquitectura fue desarrollada sobre la base del simulador *XML Behaviour Control*, por lo tanto se hará referencia al documento de Manual del Usuario de esta aplicación. A continuación los principales aspectos para la utilización de la herramienta.

C.1. REQUERIMIENTOS DEL SISTEMA

Para una correcta simulación se recomienda que el equipo de cómputo cuente con:

- Procesador: Pentium IV o superior
- Memoria: 512 de RAM o superior

- GNU/Linux o Windows con maquina virtual de Java 1.6 o superior
- Editor de archivos XML

C.2. ARCHIVOS DE TIPO XML

Las definiciones de estrategias, ambiente y comportamiento de jugadores se realizan con archivos de tipo XML. Para poder crear/modificar estos documentos es necesario conocer algunos aspectos básicos de este lenguaje. XML es un lenguaje basado en etiquetas muy parecido a HTML con la diferencia de que XML, es más genérico cuyo principal objetivo es describir, crear, estructurar y guardar datos. De esta forma a través de los archivos XML se estructura y almacena la información necesaria para la aplicación de la arquitectura de estrategias.

Los elementos se estructuran con etiquetas que marcan el inicio `<etiqueta>` de un elemento y el fin `< /etiqueta>` de un elemento, aunque si la etiqueta no contiene otros elementos ni texto libre se puede abrir y cerrar la vez con la síntaxis `<etiqueta.../ >`, por ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<behavior name="goalShot">
        <state name="shootFront">
                <action id="MOTION-shootFront"/>
        </state>
</behavior>
```

En el ejemplo anterior de código XML, `behavior` es un elemento que posee otro de tipo `state`, el cual contiene otro de tipo `action`. Los elementos además de incluir otros datos pueden tener atributos, por ejemplo: `<behavior name="goalShot">`, en donde el elemento `behavior` tiene un atributo `name` cuyo valor es *goalShot*. Los comentarios de XML inician con la cadena $<! - -$ y finalizan con la cadena $- - >$, estos e ignorarán al momento de analizar las partes del documento.

C.3. EDICIÓN DE ARCHIVOS DE TIPO XML

Para crear/editar un archivo de XML se puede hacer uso de cualquier editor de texto, sin embargo, se sugiere la aplicación de alguna herramienta especial para la edición de estos archivos, por la facilidad de uso y apoyo a los diseñadores y desarrolladores de aplicaciones relacionas a XML. En este trabajo de investigación se utilizó la herramienta **Editix** versión 2008 (Service Pack 2) para Unix/Linux.

C.3.1. HERRAMIENTA EDITIX

Editix es un editor de archivos XML y de validación de archivos XSLT para Windows, Linux, y Mac OS X. Esta herramienta ayuda a los diseñadores de aplicaciones Web y programadores de aplicaciones de tecnologías relacionadas a XML como XSLT / XSL-FO / DocBook y DTD / XML Schema /RelaxNG. Proporciona una extensa variedad de funcionalidades entre estas validación de entradas inteligentes.

El editor construye una ventana gráfica con los elementos del documento al momento de escribirlos o de un documento DTD; todos los elementos se observan como un árbol dirigido, el cual puede ser creado, editado y expandido de manera gráfica. En la Figura C.1 se presenta el entorno principal de la herramienta Editix.

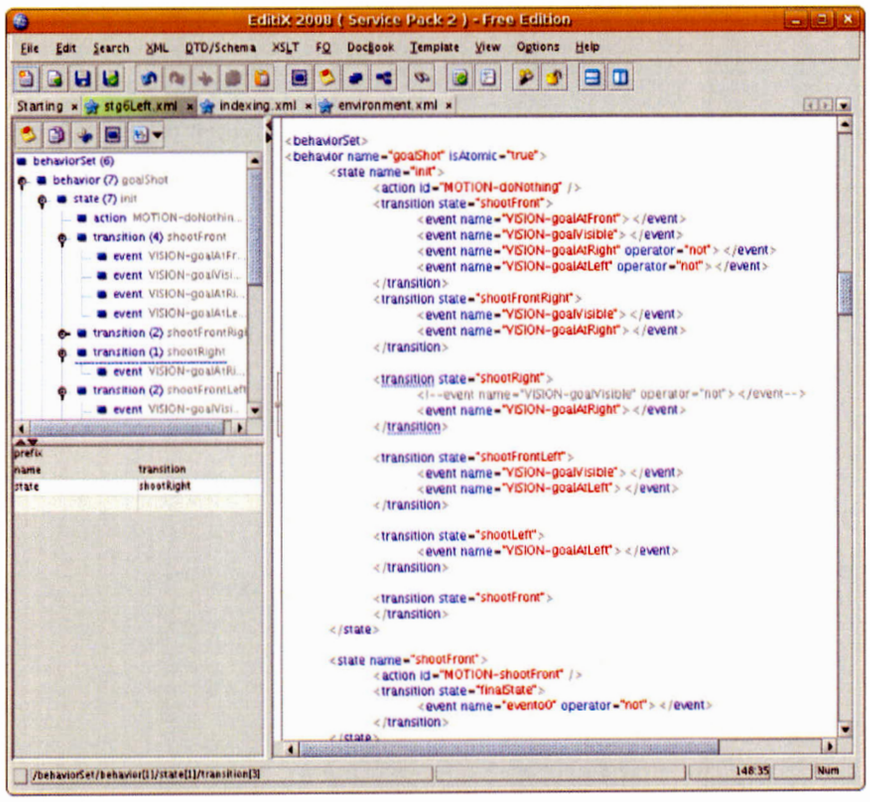


Figura C.1: Herramienta Editix

C.3.2. INSTALACIÓN Y EJECUCIÓN DE EDITIX

La herramienta **Editix** puede ser descargada de forma gratuita para uso no comercial; en la Figura C.2 se presenta la dirección URL para descargar la herramienta **Editix**. Como requerimientos de instalación se debe contar con la *máquina virtual* de Java, versión 1.6 o superior. Para la instalación y ejecución de la herramienta Editix seguir los siguientes pasos:

1) Ingresar al la dirección URL:

```
http://free.editix.com/download.html
```

2) Descargar el archivo de instalación correspondiente al sistema operativo seleccionado

3) Ejecutar archivo de instalación

- Windows: ejecutar editix-free-2008-sp2-win.exe y seguir los pasos de instalación
- Unix/Linux: descomprimir editix-free-2008-sp2.tar.gz
- Mac OS X: descomprimir editix-free-2008-sp2.zip

4) Ejecutar herramienta Editix

- Windows: ejecutar acceso directo desde escritorio
- Unix/Linux: ejecutar el *batch* **run.bat**
- Mac OS X: ejecutar desde el *finder*

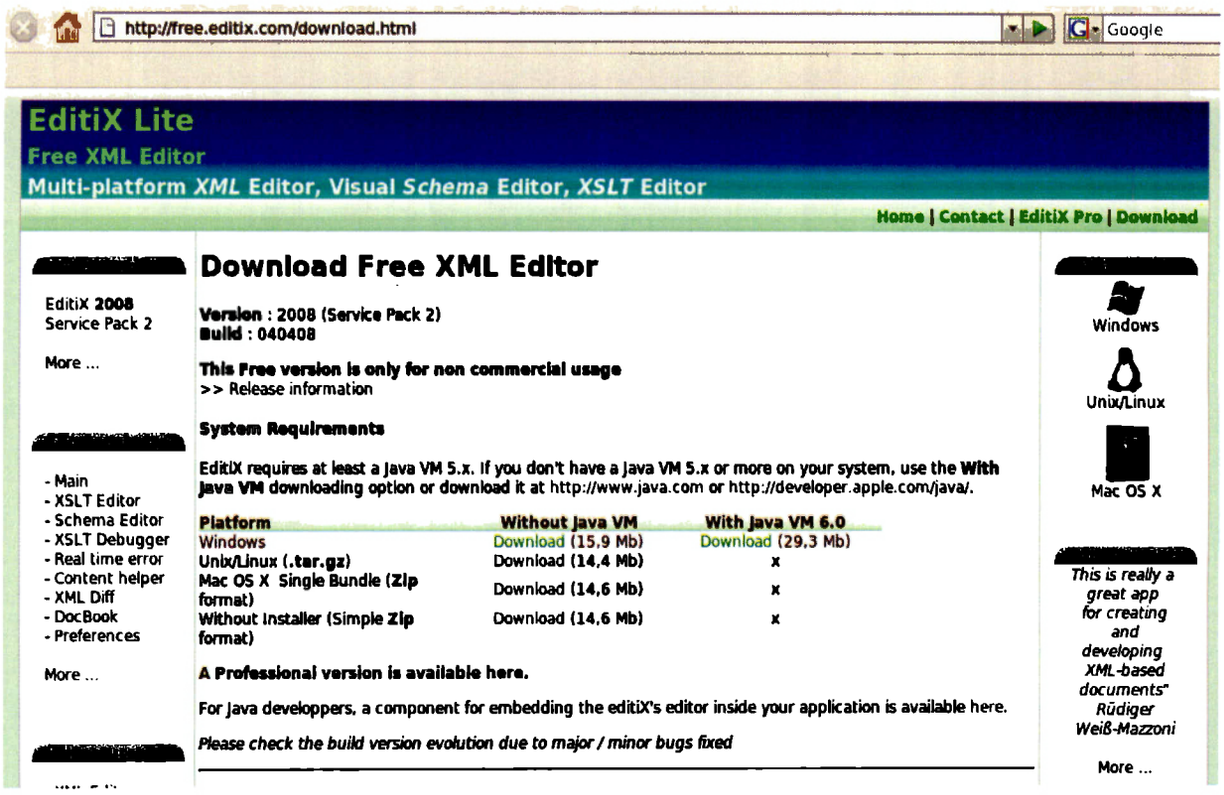


Figura C.2: URL para descargar Editix

C.3.3. EDICIÓN DE ARCHIVOS XML

Una vez ejecutada la aplicación Editix se puede hacer uso de todas las características de la herramienta entre estas crear o modificar archivos de tipo XML. En la siguiente sección se descri-

birá la creación/modificación de los archivos de tipo XML, esenciales en la aplicación de arquitecturas de estrategias para la definición y selección de estrategias.

C.4. ARCHIVO: DEFINICIÓN DE AMBIENTE

Todo agente esta situado dentro de un ambiente, en el cual colabora con otros agentes para llevar a cabo sus objetivos de diseño. La definición de un ambiente es de suma importancia para definir las características de los agentes y el entorno general que lo rodea. Para este caso, el entorno se enmarca en una competencia de fútbol robótico en la cual dos equipos de agentes compiten por ganar.

La Arquitectura de Estrategias para la Generación y Selección de estrategias de juego, hace uso del archivo *environment.xml* aplicado en el simulador *XML Behaviour Control* para la definición del ambiente. En este archivo se definen los parámetros de simulación así como los robots que intervienen en la misma. Está diseñado para que su modificación y adaptación se realice de forma fácil y dinámica, y pueda ser aplicado a otros entornos o modificaciones al ambiente.

El elemento raíz del documento se identifica por la etiqueta `<environment>` y consta de los siguientes atributos:

- *virtual_space*: espacio virtual donde estarán ubicados los agentes.
- *robot_delay*: tiempo de respuesta de los robots.
- *object_delay*: tiempo de respuesta de los objetos.
- *friction*: factor de fricción (desaceleración de la pelota).

Posteriormente que se ha definido el ambiente general, el archivo `environment.xml` también define las características de agentes que intervienen en la simulación mediante la etiqueta `<robot>`, estos contienen los siguientes atributos:

- *type*: indica el tipo de robot que se va a simular.
- *file*: archivo que contiene la definición de comportamiento de este robot.
- *team*: el equipo que pertenece.
- *name*: nombre para la identificación del agente.
- *initial_positionX*: posición inicial en el eje de coordenada X.
- *initial_positionY*: posición inicial en el eje de coordenada Y.
- *position*: rol inicial del agente.

En la Figura C.3 se muestra un ejemplo de la definición del archivo `environment.xml` que contiene inicialmente, bajo la etiqueta `<environment>`, la selección del espacio virtual **PlayGround2007** acorde a la definición del terreno de juego especificado en las reglas 2007, de la categoría de robots cuadrúpedos de RoboCup.

Posteriormente, se define a **40** el tiempo de respuesta de los robots y a **150** el tiempo de respuesta de los objetos y finalmente se especifica un valor de **120** como factor de fricción, cabe destacar que este valor esta definido en mm/seg^2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<environment virtual_space="PlayGround2007" robot_delay="40" object_delay="150" friction="120">
<robot type="aibo" file="TRBehaviors/stg6Center.xml"
                   team="BLUE"
                   name="Center"
                   initial_positionX="-40"
                   initial_positionY="20"
                   position="RIGHTATACKER"/>

<robot type="aibo" file="TRBehaviors/stg6Left.xml"
                   team="BLUE"
                   name="Left"
                   initial_positionX="-140"
                   initial_positionY="115"
                   position="LEFTATACKER"/>

<robot type="aibo" file="TRBehaviors/golie.xml"
                   team="BLUE"
                   name="GolieBlue"
                   initial_positionX="-300"
                   initial_positionY="0"
                   position="DINAMICKEEPER"/>

</environment>
```

Figura C.3: Archivo de Definición de Ambiente

Posteriormente se realiza la definición de todos los agentes que estarán involucrados en la simulación así como los atributos de los mismos. En este ejemplo fueron definidos tres robot de tipo **aibo** y cada uno contempla la definición del comportamiento especificado por medio de la etiqueta `<robot>`; para el caso del primer robot, el archivo **TRBehaviors/stg6Center.xml** define su comportamiento con base en máquinas de estado (para mayor detalle del contenido de este archivo ver Sección C.6), el atributo **BLUE** indica que el agente pertenecerá a este equipo, **Center** es el nombre que se le asignará al robot para su identificación en la simulación, la posición inicial del robot esta dada por el las coordenadas $(-40, 20)$ en X y Y respectivamente, y a su vez, el agente se le define un rol inicial de **RIGHTATACKER**.

C.5. ARCHIVO: ÍNDICE DE ESTRATEGIAS

La arquitectura de estrategias para la definición y selección de estrategias hace uso de la metodología de Razonamiento Basado en Casos, un modelo de razonamiento basado en "recordar" casos experimentados y aplicarlos en una nueva situación, además incorpora a la resolución de problemas, entendimiento y aprendizaje en un solo esquema, al aplicar diferentes tecnologías de Inteligencia Artificial, en alguna de las etapas del su ciclo de vida. Este ciclo comprende cuatro procesos básicos, entre ellos recuperación, reutilización, revisión y retención.

Dentro de los aspectos más importantes de la fase de recuperación es la identificación de casos similares experimentados con anterioridad. Para eso es necesario definir un método de *indexación*. La indexación se refiere a la tarea de asociar la clave de un registro en una unidad de almacenamiento, con otros atributos del registro, o en otras palabras, permite realizar una búsqueda más eficiente de un registro dentro de una base de registros.

Par la arquitectura de estrategia se ha definido un archivo llamado `indexing.xml` que puede ser modificado de forma manual que incorpora el potencial de los archivos de tipo ".xml"para su recuperación y procesamiento. Este archivo tiene como objetivo asociar los atributos generales de una estrategia como son: identificador, nombre, tipo, tiempo, posición del balón y posición de los jugadores con la base de estrategias definidas por otro archivo de estrategias extendido, que incluye las acciones y eventos por cada agente.

El archivo `indexing.xml` contiene las etiqueta `<strategySet>` que indica el inicio de la definición de índices del set de estrategias. El orden de los índices de estrategias es irrelevante, pero se sugiere que se organices por tipo, es decir, índices de estrategias de *KickOff*, *Ofensive*, *Defensive*. La etiqueta `<strategy>` indica el inicio de la definición de un índice de estrategia y contiene los siguientes atributos:

- *id*: un identificador que permite asociar el índice de estrategia con la definición de estrategia.
- *name*: una breve descripción de la estrategia, sólo para propósitos generales.
- *useTimer*: tiempo en milisegundos que la estrategia puede durar como máximo.
- *type*: indica el tipo de estrategia este puede ser *KickOff*, *Ofensive*, *Defensive*.
- *pl-pos*: indica un arreglo de posiciones ideales de los jugadores, el valor de cada instancia es un número de cuadrante.
- *bl-pos*: indica la posición de la pelota ideal para que se ejecute la estrategia.

La Figura C.4 muestra un ejemplo de la definición de índices de estrategias.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<strategySet>
        <strategy id="KO1" name="Right_kickOff" useTimer="7000"  type="KICKOFF">
                <and>
                        <pl-pos name="Pl2Pos" value="5"/>
                        <pl-pos name="Pl3Pos" value="6"/>
                        <pl-pos name="Pl4Pos" value="3"/>
                </and>
                <bl-pos value="3"/>
        </strategy>
        <strategy id="OF7" name="Right_ofensive" useTimer="7000" type="OFENSIVE">
                <and>
                        <pl-pos name="Pl2Pos" value="6"/>
                        <pl-pos name="Pl3Pos" value="3"/>
                        <pl-pos name="Pl4Pos" value="2"/>
                </and>
                <bl-pos value="4"/>
        </strategy>
</strategySet>
```

Figura C.4: Archivo de Definición de Índice de Estrategias

C.6. ARCHIVO: DEFINICIÓN DE ESTRATEGIAS

El archivo de definición de estrategias corresponde al conjunto de comportamientos que los jugadores van desempeñar durante el tiempo que dure un partido. El conjunto de estrategias está definido por la etiqueta `<behaviorSet>` que a su vez engloba la definición de cada una de las estrategias con la etiqueta `<behavior>`. El conjunto de estrategias se define en un archivo de tipo ".xmlüno por cada agente que estará involucrado en el juego, lo cual corresponde a la definición realizada en el archivo *environment.xml*.

Cabe destacar que para ejecutar correctamente la aplicación de arquitectura de estrategias para la definición y selección de estrategias, es importante que todos los agentes compartan los nombres (identificadores) de cada una de las estrategias, a pesar que pueden estar contenidos en documentos diferentes nombre pero de tipo ".xml". Este documento debe contener la información relacionada a la definición de estrategias, estados, transiciones, acciones y eventos. A continuación se detallará cada uno de estos aspectos.

C.6.1. DEFINICIÓN DE ESTRATEGIA

Contiene los datos generales de la estrategia bajo la etiqueta `<behavior>`, en la Figura C.5 se presenta un ejemplo de la definición de estrategia en el archivo ".xml". Sus atributos son:

- name: nombre, identificador de la estrategia
- isAtomicBehavior: estrategia debe ser ejecutada de principio a fin sin ser interrumpida
- isInitialBehavior: comportamiento inicial del robot

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<behaviorSet>
<behavior name="KO1" isInitialBehavior="true">
(...)
</behavior>
</behaviorSet>
```

Figura C.5: Archivo de Definición de Estrategia

C.6.2. DEFINICIÓN DE ESTADOS

Representa el contexto en el cual se encuentra un agente en un momento determinado, la información está contenida en la etiqueta `<state>`, en la Figura C.6 se presenta un ejemplo de la definición de estados en el archivo ".xml". Sus atributos son:

- name: nombre, identificador del estado.
- iterate: determinada si la acción realizada en este estado es iterativa o no.
- isFinalState: determina si el estado es final o no.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<behaviorSet>
<behavior name="KO1" isInitialBehavior="true">
        <state name="observarBalon" iterate="true">
        (...)
        </state>
</behavior>
</behaviorSet>
```

Figura C.6: Archivo de Definición de Estados

C.6.3. DEFINICIÓN DE TRANSICIONES

Representa los cambios de estado, árbol de decisión, de un agente durante la ejecución de una estrategia. Están asociados a eventos que simbolizan los cambios en el medio que afectan el comportamiento de los agentes en un momento determinado. El estado destino debe estar especificado dentro de la definición de estrategia para que pueda ser interpretada correctamente.

La información está contenida en la etiqueta `<transition>`, en la Figura C.7 se presenta un ejemplo de la definición de transiciones en el archivo ".xml". Las transiciones tienen una relación de prioridad implícita, es decir que las transiciones que están al inicio, tienen prioridad sobre las posteriores, esto en dependencia de que se cumplan los criterios o acciones para realizar la transición.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<behaviorSet>
<behavior name="KO1" isInitialBehavior="true">
        <state name="observarBalon" iterate="true">
        <transition state="pararBalon">
        (...)
        </transition>
        <transition state="aproachToBall">
        (...)
        </transition>
        </state>
</behavior>
</behaviorSet>
```

Figura C.7: Archivo de Definición de Transiciones

C.6.4. DEFINICIÓN DE ACCIONES

Las acciones, como su nombre lo indica representa la ejecución de un comportamiento. Estas pueden ser básicas como por ejemplo *MOTION-stopBall* o complejas como por ejemplo *turnTo-Pass*, las acciones complejas representan a su vez un subconjunto de acciones y requieren de un diseño mas elaborado, la definición de las mismas se realiza como un comportamiento definido y puede ser utilizada por todas las estrategias definidas dentro del `<behaviorSet>`.

En la Figura C.8 se presenta un ejemplo de la definición de acciones en el archivo ".xml". Las acciones básicas se representan con la etiqueta `<action>` y cuentan con un atributo *id* el cual la identifica a esta acción; las acciones complejas se representan con la etiqueta `<complexAction>` y cuentan con un atributo *name*, que define el nombre del comportamiento que se le proporcionó al momento de definirla.

C.6.5. DEFINICIÓN DE EVENTOS

Los eventos son los disparadores que permiten ejecutar la transición de un estado a otro, están definidos dentro la especificación de transición. Una transición se ejecuta si todos los eventos se cumplen. Los eventos pueden ser de tipo lógico *boolean* (verdadero o falso) como por ejemplo *portería_visible* ó eventos de tipo analógico como por ejemplo *distancia_portería*, valores que son actualizados de acuerdo a la percepción del agente.

Los eventos básicos se representan con la etiqueta `<event>`, en la Figura C.9 se presenta un ejemplo de la definición de eventos en el archivo ".xml". En dependencia del tipo de evento los atributos son:

- name: identificador del evento.
- operator: operador lógico sobre el evento (negación).
- value: valor contra el cual se realiza la comparación.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<behaviorSet>
<behavior name="turnToPass" isInitialBehavior="false">
        <state name="init">
                <action id="MOTION-stopBall"></action>
                <transition state="turnRight">
                        <action id="MOTION-stopBall"></action>
                        (...)
                </transition>
                <transition state="turnLeft">
                        <action id="MOTION-stopBall"></action>
                        (...)
                </transition>
        </state>
        (...)
</behavior>
<behavior name="KO1" isInitialBehavior="true">
        <state name="girarConBalon" iterate="true">
                <action id="COMPLEX-shootToPosition" param1="7" param2="1"/>
                <complexAction name="turnToPass"></complexAction>
                <transition state="moveToPosition-8">
                        <action id="MOTION-shootFront"/>
                        (...)
                </transition>
        </state>
        (...)
</behavior>
</behaviorSet>
```

Figura C.8: Archivo de Definición de Acciones

Los Cuadros C.1, C.2, C.3 y C.4 listan los eventos y las acciones, implementadas en el simulador *XML Behaviour Control* y actualizados por la arquitectura de estrategias para la definición y selección de estrategias, de acuerdo al módulo al que pertenecen. Con estos eventos y acciones se pueden generar comportamientos completos para definir estrategias de fútbol robótico, pero se tiene la posibilidad de incrementar este listado programando nuevas funcionalidades (ver Manual de Programación).

ACCIONES COMPLEJAS	
Evento	Descripción
COMPLEX-goToMasterPosition	Permite posicionar al robot en su posición principal de acuerdo a su rol
COMPLEX-goToQuadrantPosition	Permite posicionar al robot en un cuadrante definido por un parámetro
COMPLEX-shootToPosition	Permite que el robot patée el balón en un cuadrante definido por un parámetro

Cuadro C.1: Acciones Complejas

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<behaviorSet>
<behavior name="turnToPass" isInitialBehavior="false">
        <state name="init">
                <action id="MOTION-stopBall"></action>
                <transition state="turnRight">
                        <action id="MOTION-stopBall"></action>
                        <event name="COMM-passTargetAtRight"/>
                </transition>
                <transition state="turnLeft">
                        <action id="MOTION-stopBall"></action>
                        <event name="COMM-passTargetAtLeft"/>
                </transition>
        </state>
        (...)
</behavior>
<behavior name="KO1" isInitialBehavior="true">
        <state name="girarConBalon" iterate="true">
                <action id="COMPLEX-shootToPosition" param1="7" param2="1"/>
                <complexAction name="turnToPass"></complexAction>
                <transition state="moveToPosition-8">
                        <action id="MOTION-shootFront"/>
                        <analogEvent name="timer" value="2000"/>
                </transition>
        </state>
        (...)
</behavior>
</behaviorSet>
```

Figura C.9: Archivo de Definición de Eventos

C.7. EJECUCIÓN DE LA APLICACIÓN

Una vez que los archivos de configuración y de definición de comportamientos han sido modificados/generados, la aplicación se puede ejecutar directamente de la herramienta de desarrollo como `Eclipse` (ver sección Manual de Programación), o se puede generar manualmente un archivo **JAR** para la ejecución. Un archivo **JAR** (Java **AR**chive) es usado para agregar muchos archivos en uno solo, útil para distribuir *clases* de Java y *metadatos* asociados. Para la creación y ejecución del archivo **JAR** seguir los siguientes pasos:

1) Abrir línea de comando.

```
$
```

2) Cambiar el directorio a la ruta del proyecto de RoboCup:

```
/workspace/robocup$
```

3) Ejecutar el comando: para generar el archivo `jar`.

```
/workspace/robocup$ jar cvfm RoboCup.jar RoboManifest world/XmlExecution.class
```

4) Ejecutar el comando: para ejecutar la aplicación.

```
/workspace/robocup$ java -jar RoboCup.jar
```

MÓDULO DE VISIÓN	
Evento	Descripción
VISION-ballFound	Boolean, se activa cuando el robot observa el balón
VISION-ballAtRight	Boolean, se activa cuando el balón se encuentra a la derecha
VISION-ballAtLeft	Boolean, se activa cuando el balón se encuentra a la izquierda
VISION-ballNear	Boolean, se activa cuando el balón se encuentra a una distancia corta
VISION-ballVeryNear	Boolean, se activa cuando el balón se encuentra muy cerca
VISION-ballOnOwnPenaltyArea	Boolean, se activa cuando el balón se encuentra dentro del área del portero
VISION-ballOnOwnField	Boolean, se activa cuando el balón se encuentra dentro de la mitad de la cancha donde se encuentra la propia portería
VISION-shootDistance	Boolean, se activa cuando el balón se encuentra lo suficientemente cerca como para disparar
VISION-goalVisible	Boolean, se activa cuando se puede observar la portería
VISION-goalAtFront	Boolean, se activa cuando la portería esta frente al jugador
VISION-goalAtBack	Boolean, se activa cuando la portería se encuentra detrás del jugador
VISION-goalAtRight	Boolean, se activa cuando la portería se encuentra a la derecha del jugador
VISION-goalAtLeft	Boolean, se activa cuando la portería se encuentra a la izquierda

Cuadro C.2: Eventos del Módulo de Visión

MÓDULO DE LOCOMOCIÓN	
Acción	Descripción
MOTION-DoNothing	No realiza ninguna acción
MOTION-TurnRight	Gira el cuerpo del robot a la derecha
MOTION-TurnLeft	Gira el cuerpo del robot a la izquierda
MOTION-GoForward	Camina hacia delante
MOTION-goBackward	Camina hacia atrás
MOTION-shootFront	Patea hacia delante
MOTION-shootRight	Patea hacia la derecha
MOTION-shootLeft	Patea hacia la izquierda
MOTION-shootRightSoft	Patea hacia la derecha con poca intensidad
MOTION-shootLeftSoft	Patea hacia la izquierda con poca intensidad
MOTION-shootFrontRight	Patea a la derecha con ángulo de 45 grados
MOTION-ShootFrontLeft	Patea a la izquierda con ángulo de 45 grados
MOTION-LateralRight	Camina en diagonal hacia la derecha
MOTION-LateralLeft	Camina en diagonal hacia la izquierda
MOTION-turnHeadRight	Gira la cabeza del robot hacia la derecha
MOTION-turnHeadLeftt	Gira la cabeza del robot hacia la izquierda
MOTION-turnRightWithBall	El robot gira con la pelota hacia la derecha
MOTION-turnLeftWithBall	El robot gira con la pelota hacia la izquierda
MOTION-StopBall	Detener el balón cuando se encuentra en frente

Cuadro C.3: Acciones del Módulo de Locomoción

MÓDULO DE COMUNICACIÓN	
Evento	Descripción
COMM-readyToPass	Boolean, se activa cuando el robot está listo para enviar pase
COMM-passTargetAtRight	Boolean, se activa cuando el objetivo de pase hasta a la derecha
COMM-passTargetAtLeft	Boolean, se activa cuando el objetivo de pase hasta a la izquierda

Cuadro C.4: Eventos del Módulo de Comunicación

APÉNDICE D. MANUAL DE PROGRAMACIÓN

La arquitectura para la generación y selección de estrategias esta desarrollada sobre la herramienta de Simulación *XML Behaviour Control* escrito en Java, lenguaje de programación orientado a objetos. La implementación se realizó agregando módulos y realizando mejoras a los ya existentes. La clase principal para ejecutar la aplicación es `XmlExecution` que a su vez hace referencia a la clase principal del la herramienta de selección de estrategias `CBRMainClass`. La aplicación esta compuesta por los siguientes paquetes:

- cbr: contiene las clases principales para la definición y selección de estrategias de juego.
- world: contiene las clases que definen y administran los objetos del mundo virtual.
- errorHandling: contiene las clases para controlar las excepciones de los demás módulos.
- utilities: contiene las clases para realizar funciones matemáticas.

D.1. REQUERIMIENTOS DEL SISTEMA

Para una correcta instalación de la herramienta se recomienda que el equipo de cómputo cuente con las siguientes características:

- Procesador: Pentium IV o superior
- Memoria: 512 de RAM o superior
- GNU/Linux o Windows con maquina virtual de Java 1.6 o superior

- Editor de lenguaje Java
- Editor de archivos XML

D.2. EDICIÓN DE CLASES JAVA

Para crear/editar un archivo de Java se puede hacer uso de algún editor de texto, sin embargo, se sugiere la aplicación de alguna herramienta especial para la edición de estos archivos, por la facilidad de uso y apoyo a los diseñadores y desarrolladores de aplicaciones relacionadas a Java y la administración de proyectos completos. En este trabajo de investigación se utilizó la herramienta **Eclipse** versión Europa (3.3.0) para Unix/Linux.

D.2.1. HERRAMIENTA ECLIPSE

Eclipse es un Entorno de Desarrollo Integrado en voz inglesa *Integrated Development Environment* (IDE) ideal para el desarrollo de aplicaciones en Java; es de código abierto e independiente de una plataforma, desarrollado inicialmente por IBM y actualmente por la organización Fundación Eclipse.

D.2.2. INSTALACIÓN Y EJECUCIÓN DE ECLIPSE

La herramienta **Eclipse** puede ser descargada de forma gratuita desde el URL principal de Eclipse, en la Figura D.1 se presenta la dirección URL para descargar la herramienta. Como requerimientos de instalación se debe contar con la *máquina virtual* de Java versión 1.6 o superior. Para la instalación y ejecución de la herramienta Eclipse seguir los siguientes pasos:

1) Ingresar al la dirección URL:

```
http://www.eclipse.org/downloads/
```

2) Descargar el archivo de instalación correspondiente al sistema operativo seleccionado para Eclipse IDE for Java Developers

3) Descomprimir archivo de instalación

   - Windows: descomprimir eclipse-java-europa-winter-win32.zip
   - Unix/Linux: descomprimir eclipse-java-europa-winter-linux-gtk.tar.gz
   - Mac OS X: descomprimir eclipse-java-europa-winter-macosx-carbon.tar.gz

4) Ejecutar herramienta Eclipse

   - Windows: iniciar archivo ejecutable desde el directorio principal de Eclipse
   - Unix/Linux: iniciar archivo ejecutable desde el directorio principal de Eclipse

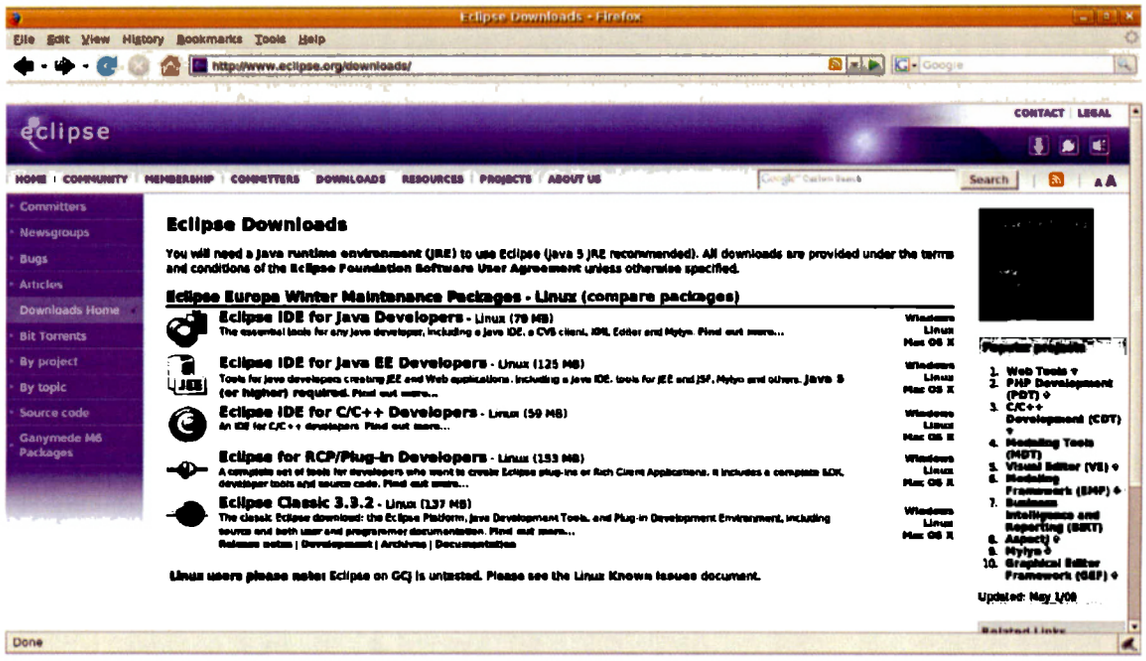


Figura D.1: URL para descargar Eclipse

- Mac OS X: iniciar archivo ejecutable desde el directorio principal de Eclipse

5) Definir espacio de trabajo (workspace) Una vez que ha ejecutado el archivo de Eclipse, la herramienta solicitará la selección del espacio de trabajo, en el cual se instalará la herramienta de simulación. El espacio de trabajo o en voz inglesa *workspace* es un conjunto de metadatos sobre un espacio de archivos que permite la modificación de archivos en tanto correspondan al espacio de trabajo definido. El espacio de trabajo para el sistema operativo Windows puede ser:

```
c:\workspace
```

Y para Unix/Linux:

```
/home/myarias/workspace/
```

D.3. XML BEHAVIOUR CONTROL Y TABA

D.3.1. INSTALACIÓN

Una vez que la herramienta Eclipse esté instalada correctamente se procede a ejecutar la herramienta de simulación XML Behavior Control y TABA (Team Agent Behavior Architecture), para la definición y selección de estrategias. Para la instalación de la herramienta de simulación puede seguir los siguientes pasos:

1) Descomprimir el archivo robocup.zip o robocup.tar.gz dentro del directorio del espacio de trabajo.

2) Ejecutar eclipse.

3) Seleccionar Archivo (File)/ Nuevo (New)/ Proyecto de Java (Java Project) / Contenido (Contens) / Crear proyecto desde un archivo existente. (Create project from existing source) / Botón de búsqueda (Browse) / Seleccionar la carpeta robocup.

4) Definir nombre de proyecto (Project name)/ robocup.

5) Presionar botón finalizar (Finish).

En la Figura D.2 se muestra la definición proyecto **robocup** en Eclipse. La herramienta definirá en

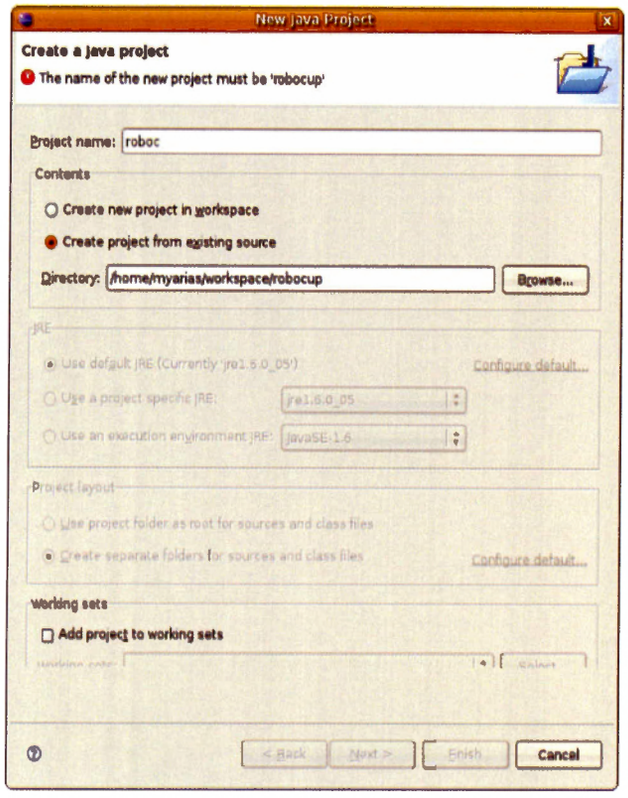


Figura D.2: Creación del proyecto robocup

su explorador de paquete (Package Explorer) una entrada con la raíz robocup y el árbol correspondiente a los paquetes que éste contiene, tal como se muestra en la Figura D.3

D.3.2. MODIFICACIÓN

Al inicio de este manual se mencionó los principales paquetes contenidos en el proyecto robocup, el paquete **world**, **errorHandling** y **utilities** proceden originalmente del simulador *XML Behaviour Control* pero cabe destacar que estos han sufrido modificaciones en especial el paquete **world**. El paquete **cbr** es original de TABA y contiene el centro de las operaciones ejecutadas para la definición y selección de estrategias. A continuación se proporcionará un mayor detalle de los mismos.

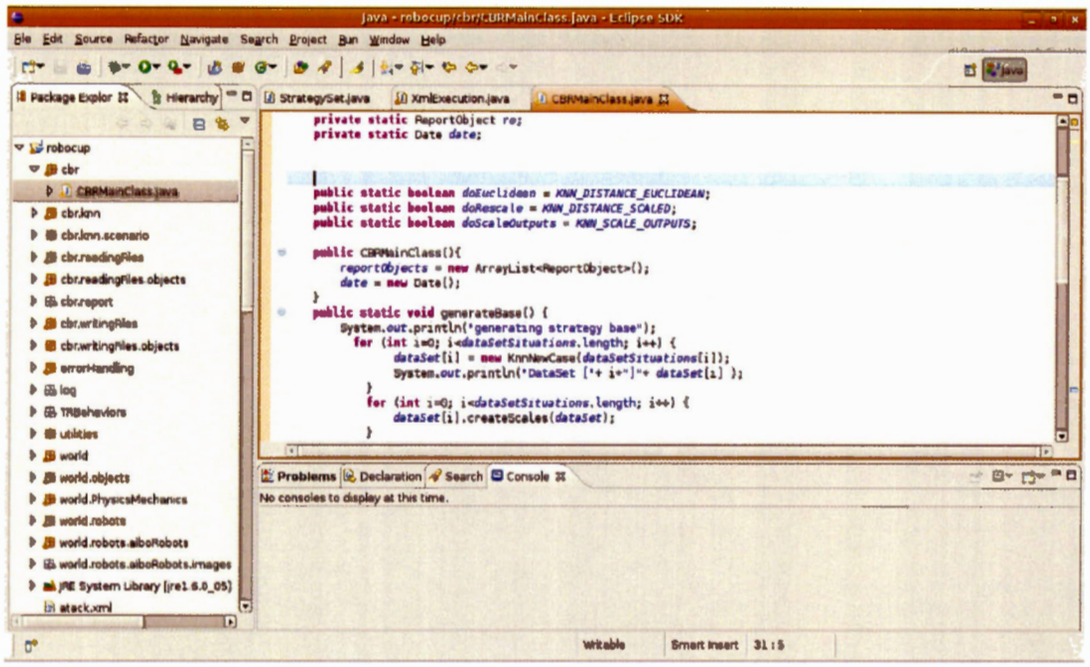


Figura D.3: Proyecto RoboCup en Eclipse

Paquete CBR

Este paquete contiene las clases que especifican la definición y selección de estrategias, por medio de: *i-lectura de archivos*: archivos de definición de ambiente, indexación y estrategia, *ii-escritura de archivos*: archivos para la generación de reportes y análisis generales, *iii- maquinaria de selección*: definición del escenario y configuración de algoritmos del *K-vecino mas cercano*.

Paquete World

Este paquete contiene las clases que definen al mundo virtual del Simulador *XML Behaviour Control*, esto es la definición de: *i-ambiente*: integración de todos los objetos de la simulación, metodos para dividir en secciones el campo, *ii-objetos*: pelota, campo de juego, *iii-robots*: aibo, *iv-módulos del robot*: locomoción, visión, comunicación y *v-ejecución de la aplicación*.

Paquete ErrorHandling

Este paquete contiene las clases útiles para la administración de las excepciones que los demás paquetes pueden arrojar

Paquete Utilities

Este paquete contiene las clases el manejo de algunas funciones matemáticas (trapezoide, recta, transformación de ángulo, distancia euclidiana) implementadas en otros paquetes, por ejemplo para

la evaluación del riesgo o la distancia entre dos casos, ver Figura D.4.

```
public class EuclideanDistance {
        public static double measureDistance(KnnNewCase situation1, KnnNewCase situation2,
                boolean doEuclidean, boolean doRescale) {
                double result = 0.0f;

                for (int i=0; i<situation1.getProperties().length; i++) {
                        if (!situation1.getProperty(i).isUnknown()) {
                                if (doEuclidean) {
                                        result += Math.pow((situation1.getProperty(i).getIndex() -
                                                situation2.getProperty(i).getIndex(doRescale)), 2);
                                } else {
                                        result += Math.abs((situation1.getProperty(i).getIndex() -
                                                situation2.getProperty(i).getIndex(doRescale)));
                                }
                        }
                }
                if (doEuclidean) result = Math.pow(result, 0.5);
                return result;
        }
}
```

Figura D.4: Cálculo de distancia entre dos casos

D.3.3. EJECUCIÓN

La ejecución de la herramienta de simulación desde **Eclipse** se puede hacer por medio de:

1) Menú - Ejecutar (Run) / Ejecutar como (Run As) / Aplicación de Java (Java Application) / XmlExecution ó

2) Explorador de Paquete - World / XmlExecution / clic derecho / Ejecutar como (Run As) / Aplicación de Java (Java Application) / XmlExecution ó

3) Combinación de teclas Ctrl + F11

En la Figura D.5 se muestra una forma de ejecutar la aplicación.

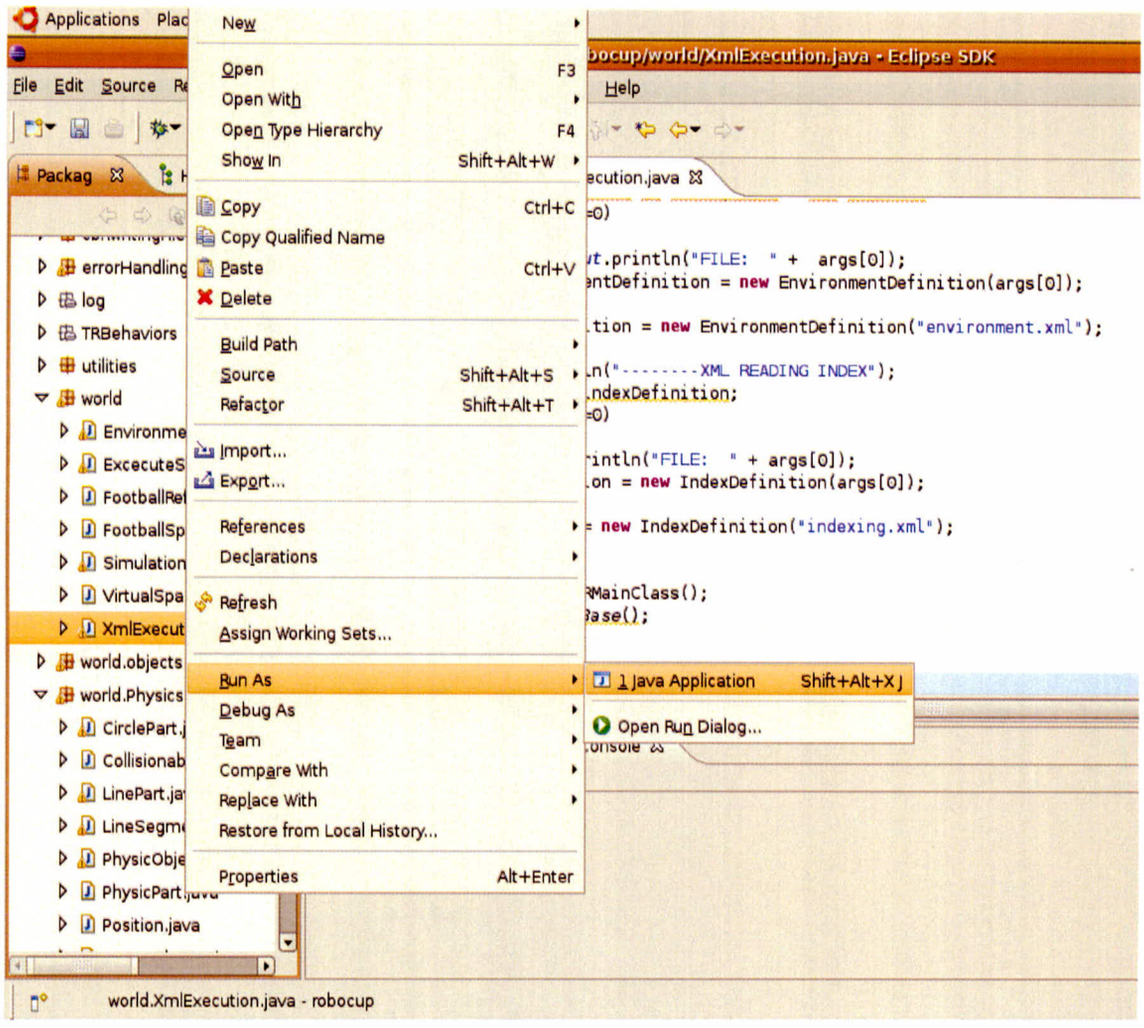
D.3.4. REPORTES

Para ver la información generada por la herramienta de definición y selección de estrategias se puede hacer por medio de:

1) Eclipse / Explorador de Paquete - cbr.report / dos clic CBR Report File ó

2) Sistema Operativo - Explorador de Carpetas - /workspace/robocup/cbr/report - CBR Report File

En la Figura D.6 se puede observar un ejemplo de la selección de estrategias a medida que el juego cambiaba de estado. Así mismo el marcador obtenido por ambos equipos.



TEAM AGENT BEHAVIOR ARCHITECTURE

Initial Time: *Fri Jul 09 08:59:33 CDT 2010* **Final Time:** *Fri Jul 09 09:09:31 CDT 2010*

It	Situation	Neighbors	Similarity	Rojos	Azul
0	(1, 33, 7, 24, 9, 9, ??)	• Neighbor #1: Strategy 8 distance = 4.5826 • Neighbor #2: Strategy 20 distance = 12.1244 • Neighbor #3: Strategy 18 distance = 12.4097	(1, 33, 9, 27, 10, 10, OF3): distance = 3.872983346207417	2	2
1	(0, 33, 7, 28, 14, 18, ??)	• Neighbor #1: Strategy 8 distance = 4.5826 • Neighbor #2: Strategy 20 distance = 12.1244 • Neighbor #3: Strategy 18 distance = 12.4097	(0, 33, 8, 27, 8, 18, KO4): distance = 6.164414002968976	2	2
2	(2, 33, 7, 32, 29, 27, ??)	• Neighbor #1: Strategy 8 distance = 4.5826 • Neighbor #2: Strategy 20 distance = 12.1244 • Neighbor #3: Strategy 18 distance = 12.4097	(1, 33, 3, 25, 26, 17, OF2): distance = 13.228756555322953	2	2
3	(0, 33, 7, 28, 14, 18, ??)	• Neighbor #1: Strategy 8 distance = 4.5826 • Neighbor #2: Strategy 20 distance = 12.1244 • Neighbor #3: Strategy 18 distance = 12.4097	(0, 33, 8, 27, 8, 18, KO4): distance = 6.164414002968976	2	2
4	(2, 33, 7, 24, 14, 29, ??)	• Neighbor #1: Strategy 8 distance = 4.5826 • Neighbor #2: Strategy 20 distance = 12.1244 • Neighbor #3: Strategy 18 distance = 12.4097	(2, 33, 11, 23, 14, 30, DF2): distance = 4.242640687119285	2	2
5	(2, 33, 12, 28, 14, 27, ??)	• Neighbor #1: Strategy 8 distance = 4.5826 • Neighbor #2: Strategy 20 distance = 12.1244 • Neighbor #3: Strategy 18 distance = 12.4097	(2, 33, 11, 23, 14, 30, DF2): distance = 5.916079783099616	2	2

Figura D.6: Ejemplo del archivo 'CBR Report File'