Un Framework para mundos virtuales

Matías M. Basílico

Juan R. Arnaude

Agradecemos a la Profesora Alicia Díaz por haber aceptado dirigir nuestra tesis y por su ayuda incondicional durante el desarrollo del trabajo.

Índice General

Índice de Figuras, Diagramas y Tablas

Prólogo

Antes de comenzar con el desarrollo de la Tesis, creemos necesario e importante realizar una pequeña reseña de nuestra evolución dentro de la carrera, junto con los conceptos y paradigmas que hemos aprendido a lo largo de la Licenciatura.

Al comienzo de la carrera, se nos transmitió insistentemente acerca de una de las metodologías más poderosas sobre la que se sustenta la informática: **la modularización**. Es decir, *"agrupar sentencias lógicamente relacionadas en módulos que describen una función o procedimiento para resolver un problema bien definido*".

Este concepto de modularización trae aparejado el diseño **Top-Down** para soluciones a problemas complejos, dividiendo el problema en subproblemas más pequeños de manera de reducir complejidad y así poder solucionarlo más fácilmente: el enunciado tan conocido de *"divide y vencerás".*

Luego de entender estos conceptos que involucran atacar el análisis de sistemas en forma modular y aprender los lenguajes procedurales de tercera generación (3GL), se nos mostró el problema que surgía al tener los datos separados de la funcionalidad. Este enfoque generaba la difícil tarea de estudiar en detalle que procedimientos se utilizaban y que información requerían, pues se encontraba todo disperso, lo que implicaba un mayor esfuerzo en unificar datos y operaciones sobre los mismos. Así comenzamos a estudiar los **TAD's (Tipos Abstractos de Datos)**, *"entidades que describen un tipo abstracto junto con sus operaciones"*, que proveen propiedades de encapsulamiento y ocultamiento de la implementación. Con esta herramienta, el desarrollador implementaba el TAD y los demás desarrolladores los utilizaban sin preocuparse por la implementación. Estos TAD's nos abstraen de los detalles de implementación y sólo nos preocupamos por utilizarlos.

Luego de los TAD's se nos enseñó que en la realidad hay clases de "entidades" u **objetos** que se agrupan por atributos y comportamientos comunes y otras clases de objetos que podrían ser definidos en base a otras clases ya analizadas. Es en este punto donde tomamos contacto con el **Paradigma Orientado a Objetos**, que agrega a los TAD's los conceptos de clase, objeto, polimorfismo, herencia y, como consecuencia de esta última, jerarquía de clases.

Presentado el paradigma Orientado a Objetos, estudiamos las relaciones posibles entre objetos, el análisis de la realidad y posterior diseño e implementación de sistemas informáticos dentro de esta filosofía de desarrollo de software.

Una vez aprendido el paradigma de la programación orientada a objetos se estudiaron los **patrones de diseño** como técnica para la solución de problemas recurrentes y la reutilización no sólo de código sino también de diseño y experiencia de muchos desarrolladores.

Aún no estaba todo dicho ya que faltaba el entorno donde puedan convivir todos estos conceptos y poder así desarrollar sobre una base en la cual los problemas recurrentes del dominio a tratar estén en su mayoría resueltos y listos para ser utilizados. A cubrir esta necesidad llega el concepto de **Framework**.

Los Frameworks nos proveen la herramienta para diseñar las relaciones de un dominio determinado sentando así una **Arquitectura de Software** que modela gran parte de la lógica de todas las aplicaciones encargadas de solucionar problemas del dominio. El Framework es la base subyacente sobre la que "descansan" las aplicaciones desarrolladas.

Ya en tercer año de la carrera, realizamos una aplicación sobre mundos virtuales de criaturas y colonias de seres vivos (**VRColonies**). Llegando al final de nuestra carrera y movilizados por dicho trabajo, nos planteamos algunos interrogantes:

- ✓ Este tipo de sistemas de objetos, o mas bien de agentes, que poseen objetivos propios y cuya principal meta es satisfacer sus necesidades e, "inconscientemente", la de sus semejantes, ¿nos estarán vislumbrando otro paradigma o área de investigación?.

- ✓ ¿Alcanza con los conceptos de la Programación Orientada a Objetos (POO) para analizar y desarrollar sistemas que ataquen la problemática de sistemas masivos de agentes que interactúan entre ellos de manera de cumplir sus objetivos?.

Las respuestas a estas preguntas fue la aparición en nuestro camino de las jóvenes, y no tanto, disciplinas de la **Vida Artificial** y **Sistemas Multi-Agentes**, que enfocan estos sistemas con nuevas ideas, utilizando el paradigma **Bottom-Up** como forma de resolver estos **sistemas complejos**.

Por esto decidimos realizar un Framework para contener los conceptos relacionados a estos mundos virtuales que conjuntamente con el paradigma **Bottom-Up** nos darán las herramientas necesarias para desarrollar la presente Tesis.

Introducción

Como enunciáramos en el Prólogo, el proceso enseñanza-aprendizaje que nos fue formando a lo largo de la carrera hizo que de alguna manera el modelado de la interacción de poblaciones de agentes u objetos autónomos sea hoy nuestro tema de interés para investigar en esta tesis.

Podemos definir los **mundos virtuales** como el estudio de programas de computadora que implementan mundos digitales con sus propias leyes físicas y biológicas[26].

La importancia de este tipo de sistemas radica en el estudio del comportamiento emergente que surge de dicha interacción y las respuestas buscadas para estos interrogantes deben encontrarse como resultado de una investigación minuciosa y perseverante dentro de las disciplinas de la Vida Artificial y los Sistemas Multi-agente.

Uno de los problemas existentes hoy en día en las disciplinas relacionadas con la Vida Artificial es la gran dispersión de conceptos. Es necesario plasmar la experiencia y conocimiento sobre el dominio aunando criterios. La creación de un Framework sobre mundos virtuales es una alternativa de solución.

Otro punto importante que nos motiva es la necesidad de realizar aplicaciones que se centren en simular procesos reales que serían irrealizables e impracticables en forma "natural" como por ejemplo: estudio de evolución de poblaciones, estudio de epidemias, dinámica de poblaciones, etc.

Hoy en día el área de simulación por computadora está en constante crecimiento y aún hoy se hace difícil predecir la utilidad que podrá alcanzar en las distintas áreas científicas.

Con estos argumentos planteados e intentando hacer realidad nuestro deseo de dilucidar este tipo de cuestiones, afrontamos este trabajo con los objetivos que exponemos a continuación.

Objetivos y resultados esperados

En primer lugar pretendemos crear un Framework que soporte aquellas aplicaciones que se desarrollen dentro del dominio de los mundos virtuales de **agentes reactivos biológicos**.

Es nuestro objetivo dotar al Framework con la infraestructura necesaria para soportar la ejecución de simulaciones que corran sobre el mismo. Esto incluye: manejo de eventos que ocurren en la simulación, manejo del factor tiempo, broadcast de los estímulos que ocurran dentro del ambiente hacia las criaturas, administración de procesos de ejecución continua, etc.

Incluiremos en el Framework la capacidad de deducir información surgida de las simulaciones (patrones dinámicos de comportamiento).

Dominio de interés

En particular nos interesan los mundos virtuales de agentes reactivos biológicos. Estas simulaciones se componen de un conjunto de agentes autónomos, activos, que persiguen ciertos objetivos y que presentan características de los seres vivos.
Intervienen también recursos de los que se valen los agentes para realizar sus tareas, sucesos que ocurren en el ambiente, modificando el estado del mismo e influyendo en el comportamiento de los agentes. En estos dominios, los agentes tienen una percepción local del entorno que impera la necesidad de colaborar con otros agentes para resolver los problemas

Organización de la Tesis

Esta sección tiene como objeto presentar el esquema general de la Tesis, resumiendo los objetivos centrales y los principales temas tratados de cada capítulo.
Los primeros dos capítulos conforman la base teórica que sustentan el desarrollo del Framework. A partir del Capítulo 3, se desarrollan los conceptos y elementos que forman parte del Framework.

Capítulo 1: Marco Teórico

El bagaje teórico sobre el tema al que se puede acceder es abundante en cantidad y variedad de fuentes. Por eso es necesario realizar una síntesis concreta y precisa sobre el material, pero que a su vez incluya completamente y clarifique los conceptos necesarios para convertirlos en base teórica para nuestro trabajo.

Para realizar esta selección y recopilación de fundamentos teóricos, inclinamos nuestra lectura e investigación de manera de expresar las intenciones o el campo de acción en el cual queremos basar este trabajo.

En este capítulo se pondrá de manifiesto lo investigado acerca del entorno teórico en el cual nos sustentamos para realizar el trabajo. El marco teórico constituye una recopilación de distintos enfoques, cuyo eje central en cuestión es la simulación de vida en un entorno informático.

Las cinco secciones que en las que se dividen el capítulo (Vida Natural, Vida Artificial, Sistemas Multi-Agentes, Sistemas Complejos y Framework), refieren a temas auto-contenidos que permiten una lectura selectiva. Por esta razón, el lector podrá decidir iniciar la lectura de la Tesis a partir del Capítulo 2 y utilizar el Capítulo 1 como referencia y apoyo teórico.

Capítulo 2: Caracterización del Framework

Se analiza el Marco Teórico, se presentan opiniones sobre el contenido del mismo y se enumeran aquellos elementos extraídos que sustentan la caracterización y el desarrollo del Framework.

Capítulo 3: Agente

Se define y diseña el componente Agente del Framework. Se explica el modelo elegido y se detallan el resto de componentes que conforman una Agente (tareas, objetivos, etc).

Capítulo 4: Ambiente

Se define y diseña el componente Ambiente. Se detallan los aspectos que intervienen al modelar un ambiente y sus componentes (señales, recursos, espacio físico, etc.)

Capítulo 5: Relaciones

En este capítulo se realiza el nexo entre el agente y el ambiente, detallando los componentes intervinientes en esa interacción (recursos, sensores, señales, etc.).

Capítulo 6: Infraestructura para la simulación

En este capítulo se presenta la infraestructura provista por el Framework en cuanto a manejo de simulación se refiere. Se describen los diferentes componentes que soportan la ejecución de simulaciones construidas y configuradas por el usuario final.

Se proveen además mecanismos para extraer y analizar la información surgida de la ejecución de las simulaciones construidas sobre el Framework.

Finalmente, expresamos las conclusiones obtenidas y planteamos sugerencias para continuar la investigación o iniciar nuevos trabajos afines con el tema. Entre otras cosas nos planteamos que brinda de nuevo el trabajo, que logros realizamos, por que seguir investigando en estos temas y cual es el futuro de este tipo de enfoque.

Capítulo 1: Marco Teórico

El marco teórico involucra dos categorías de conceptos: aquellos que estudian las entidades del mundo real y aquellos conceptos involucrados en la ciencia de la computación.

*Comenzamos este capítulo con la sección de **Vida Natural**, del que extraemos los conceptos básicos de la biología y nos aporta varias ideas útiles para el desarrollo del trabajo. Richar Dawkins es el autor del libro "El gen egoísta", que será la base de los conceptos planteados de la sección de Vida Natural.*

*La sección de **Vida Artificial**, nos plantea el paradigma bottom-up, de componentes simples a comportamiento complejos, que implementa el método sintético de la biología. Se establece además la diferenciación entre Vida Artificial e Inteligencia Artificial. Se observan diferentes sistemas de vida artificial para comprender su funcionamiento.*

*En la sección siguiente, comprendemos y definimos características que conforman un **Sistema Multi-Agente**. Se estudian dos grandes grupos de agentes, reactivos y cognitivos y conocemos las arquitecturas de agentes mas difundidas.*

*Los conceptos de las secciones anteriores se generalizan en la sección de **Sistemas Complejos**, introduciendo las definiciones de sistemas complejos, caos, autoorganización, etc.*

*Finalmente se analiza los **Frameworks** en forma teórica: las maneras de desarrollarlo y elementos a tener en cuenta. Se analizan Frameworks existentes extrayendo características relevantes.*

1.1. Vida Natural

En los orígenes reinó la simplicidad. Es difícil explicar el súbito nacimiento, con todos los atributos, de una organización tan compleja como es la vida.[1]

La **supervivencia de los más aptos** de Darwin es realmente un caso especial de una ley más general relativa a la **supervivencia de lo estable**. El universo está poblado por cosas estables. Una cosa estable es una colección de átomos bastante permanente o común para merecer un nombre.

La forma primaria de selección natural fue, simplemente, una selección de formas estables y un rechazo de las inestables.

De ello, por supuesto, no se deriva que se pueda explicar la existencia de seres tan complejos como los seres humanos exactamente por los mismos principios, sin más.

Es en este punto donde la teoría de Darwin, en su aspecto más general, viene al rescate. La teoría de Darwin interviene desde el momento en que la lenta construcción de las moléculas ha cesado.

Los químicos han intentado imitar las condiciones químicas de la Tierra en su etapa joven. Han colocado las sustancias simples que existieron en las primeras edades de la tierra (agua, dióxido de carbono, metano y amoníaco) en un matraz y le han aplicado una fuente de energía tal como la luz ultravioleta o chispas eléctricas, en calidad de simulación artificial del rayo primordial. Luego de transcurridas unas cuantas semanas suele descubrirse algo interesante dentro del matraz: un débil caldo café que contiene una gran cantidad de moléculas más complejas que las que originalmente se pusieron allí. Se han encontrado, en particular, aminoácidos, los cuales constituyen la base de las proteínas, una de las dos clases principales de las moléculas biológicas.

Procesos similares a estos deben haber dado origen al **caldo primario** que los biólogos y químicos creen que constituyó los mares hace tres o cuatro miles de millones de años. Las sustancias orgánicas llegaron a concentrarse en determinados lugares. Más tarde, bajo la influencia de una energía tal como la luz ultravioleta proveniente del Sol, se combinaron con el fin de formar moléculas mayores.

En algún punto, una molécula especialmente notable se formó por accidente. La denominaremos **el reproductor**. No tuvo que ser, necesariamente, la más grande o la más compleja de todas las moléculas, pero tenía la extraordinaria propiedad de poder crear copias de sí misma.

Tan pronto como nació el reproductor, sin duda esparció rápidamente sus copias a través de los mares hasta que las moléculas más pequeñas, cuya función era la de ser componentes de convirtieron en un recurso escaso y otras moléculas más grandes no pudieron formarse sino muy rara vez.

Parece que así llegamos a la etapa de una gran población de réplicas idénticas. Pero ahora debemos mencionar una propiedad importante de cualquier proceso de copia: no es perfecto. Ocurrirán errores. Si todas las copias fuesen hechas a partir de un original único, el significado no se falsearía mucho. Pero si las copias se hacen a partir de otras copias, los errores empezarán a ser acumulativos y graves.

A medida que se efectuaron copias con errores y éstas fueron propagadas, el caldo primario se vio poblado, no por réplicas idénticas sino por diversas variedades de moléculas reproductoras, todas **descendientes** del mismo antepasado.

Las tres características que ayudaban a aumentar el numero de copias de los reproductores eran:

- ✓ **Longevidad**:
  Las variedades de reproductores con esta característica harían que los mismos se mantengan estables a través de una mayor cantidad de tiempo permitiendo aumentar el numero de copias.

- ✓ **Fecundidad**:
  Otra propiedad inherente a una variedad de reproductores que tuvo importancia para que fuese difundida en la población, era la velocidad de reproducción.

- ✓ **Fidelidad**:
  Esta característica de las moléculas reproductoras es la referente a la exactitud de la réplica.

Retornando al caldo primario. Sin duda llegaría a estar poblado por variedades estables de moléculas; estables ya sea porque las moléculas individuales duraban un largo periodo de tiempo, porque se reproducían rápidamente o porque se reproducían con precisión. Las tendencias evolutivas hacia estos tres tipos de estabilidad tuvieron lugar en el siguiente sentido: si se hubiesen extraído muestras del caldo en dos ocasiones distintas, la última muestra habría contenido una mayor proporción de variedades con un más alto nivel de longevidad/fecundidad/fidelidad de reproducción. Esto es, esencialmente, lo que quiere decir un biólogo al referirse a la evolución cuando habla de criaturas vivientes, y el mecanismo es el mismo: la **selección natural**.

El siguiente eslabón del argumento, que le sigue en importancia, y que Darwin mismo acentuó (si bien es cierto que él estaba hablando de animales y plantas, no de moléculas), se refiere a la **competencia**.

En la imagen en que representamos al reproductor actuando como un molde o modelo se encontraba bañado en un caldo rico en pequeñas moléculas que hacía el papel de componentes que eran necesarias para hacer copias. Pero cuando los reproductores llegaron a ser numerosos, estos componentes debieron de ser utilizados en proporción tan elevada que se convirtieron en un recurso escaso y precioso. Las diferentes variedades o especies de reproductores **compitieron** por ellos.

Los reproductores empezaron no solamente a existir, sino también a construirse, para ser utilizados por ellos mismos, verdaderos recipientes, vehículos para continuar existiendo.

Los reproductores que sobrevivieron fueron aquellos que construyeron **máquinas de supervivencia** para vivir en ellas. Las máquinas de supervivencia se hicieron más grandes y más elaboradas, y el proceso fue acumulativo y progresivo.

En cuatro mil millones de años ¿cuál sería el destino de los antiguos reproductores? No murieron, porque son maestros en el arte de la supervivencia. Pero no se les debe buscar flotando libremente en el mar; ellos renunciaron a esa desenvuelta libertad y protegidos del mundo exterior, comunicándose con él por medio de rutas indirectas y tortuosas. Ellos nos crearon, cuerpo y mente; y su preservación es la razón última de nuestra existencia. Aquellos reproductores han recorrido un largo

camino. Ahora se les conoce con el término de genes, y nosotros somos sus máquinas de supervivencia.

Genética

Gen

Existen varias definiciones de gen, o lo que es lo mismo, se utiliza la palabra gen en diferentes sentidos.[2]

Gen es la unidad básica de herencia. Esta es la definición más rigurosa y la menos específica.

Los genes tienen básicamente dos funciones: reproducirse a sí mismos e indicar el modo de construcción y comportamiento de un nuevo ser vivo completo.
Podemos ver el gen como unidad funcional, como unidad de mutación o como unidad de recombinación.

En cuanto al gen como unidad funcional, un gen es una secuencia orientada de nucleótidos, generalmente de ADN, que actúan como la unidad mínima de información relativa a cuál es la constitución de otra secuencia orientada de nucleótidos de otro ácido nucleico, o de aminoácidos de una proteína.

En cuanto al gen como unidad de mutación, un gen es la unidad mínima que se puede mutar. Este gen no tiene por qué coincidir con el gen como unidad funcional.

En cuanto al gen como unidad de recombinación, un gen es la unidad mínima que se puede heredar, es decir, es la unidad mínima que puede ser tomada de uno de los progenitores para formar el nuevo individuo. Si las características heredables (como el aspecto de la cara) las descomponemos en otras subcaracterísticas (color de los ojos, color del pelo, tez), cuando ya no podamos dividir más esas características, diremos que esa característica es debida a un gen.

Genotipo y Fenotipo

La herencia biológica es un fenómeno básico de los seres vivos, inherente a su reproducción y desarrollo, y determinado por el material genético. Todas las especificaciones de este material en un cierto organismo constituyen su **genotipo**,

y todas las características efectivamente presentes en el organismo en cierto momento constituyen su **fenotipo**.

Un mismo genotipo, sometido a diferentes estados ambientales, producirá diferentes fenotipos. El fenotipo siempre resulta, entonces, del genotipo y del ambiente. Pero el fenotipo en sí no es heredable. Lo que se hereda de una generación a otra, es el genotipo y su capacidad para producir ciertos fenotipos según el ambiente.

De modo que no se heredan los caracteres (del fenotipo) sino los genes (del genotipo). Si los caracteres se perpetúan a través de las generaciones, es por la herencia de los genes y por la permanencia del ambiente.

ADN

Una molécula de ADN es una larga cadena de pequeñas moléculas llamadas nucleótidos. Consiste en un par de cadenas de nucleótidos enrollados en una espiral "doble hélice". Los nucleótidos que la componen son sólo de cuatro tipos distintos, cuyos nombres podemos abreviar A, T, C y G. Son los mismos en todos los animales y plantas. Lo que difiere es el **orden** o **secuencia** en que están ensartados. [3]

El componente G por ejemplo es igual para todo ser vivo.

Este ADN puede ser considerado como un conjunto de instrucciones de cómo hacer un cuerpo, escritas en el alfabeto A, T, C y G de los nucleótidos.

Las instrucciones del ADN han sido reunidas por selección natural.

El ADN posee dos funciones básicas:

1. **Replicarse**: Hacer copias de sí mismo.

2. **Supervisión**: Supervisa indirectamente la fabricación de un tipo diferente de molécula de proteína.

El mensaje codificado del ADN, escrito en el alfabeto nucleótido de 4 letras es traducido, por una simple mecánica, en otro alfabeto. Este es el alfabeto de los aminoácidos que comprende las moléculas de proteínas. Este es el primer paso del pasaje del **genotipo** al **fenotipo**.

Entrecruzamiento (Crossover)

El cruzamiento es un mecanismo de reproducción a partir del cual se forma un nuevo cromosoma combinando partes de al menos otros dos cromosomas 'padres'. La forma más simple de cruzamiento consiste en que en algún punto aleatorio un cromosoma original es "pinchado", y toda la información procedente del padre "A" es copiada desde el principio hasta el punto de cruzamiento, y a partir de allí hasta el final del cromosoma, toda la información correspondiente al padre "B". La *Figura 1* ejemplifica gráficamente Crossover con un punto de división.

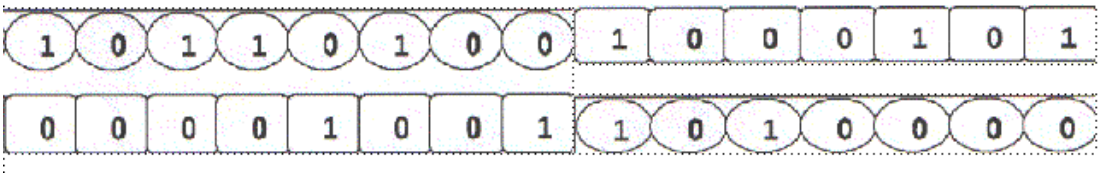

Figura 1 – Crossover con un punto de división

Las variaciones en este tipo de cruzamiento se deben al hecho de que tanto el punto de cruzamiento como el orden en el que se eligen los cromosomas padres para llevar a cabo esta operación son variables, generando así distintos cromosomas hijos. [4]

Cruzamientos más complicados se producen intercambiando porciones del material genético. Las posibilidades que ofrecen estos son prácticamente ilimitadas. La *Figura 2* ejemplifica gráficamente Crossover con dos puntos de división.

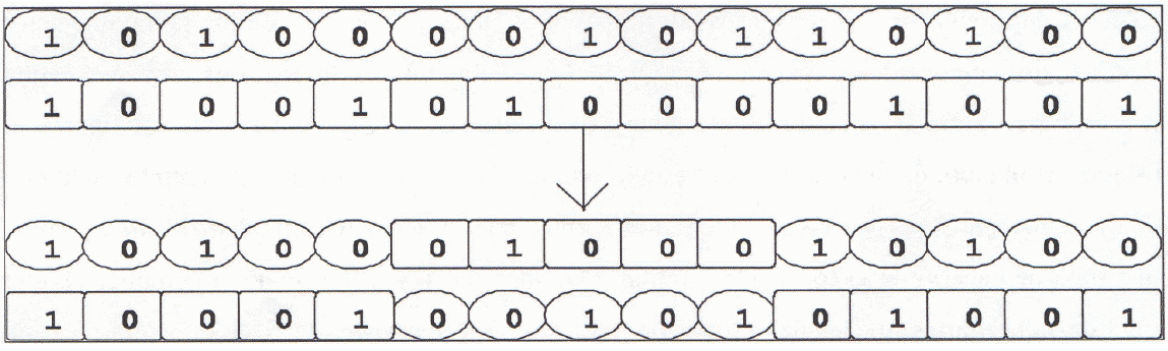

Figura 2 – Crossover con dos puntos de división

Evolución

La teoría de la selección de las especies de **Darwin** sostiene que aquellos individuos de una población que posean los caracteres más ventajosos dejarán proporcionalmente más descendencia en la siguiente generación; y si tales caracteres se deben a diferencias genéticas, que pueden transmitirse a los descendientes, tenderá a cambiar la composición genética de la población, aumentando el número de individuos con dichas características. De esta forma, la población completa de seres vivos se adapta a las circunstancias variables de su entorno. El resultado final es que los seres vivos tienden a perfeccionarse en relación con las circunstancias que los envuelven.[2]
En conclusión: la población cambia (evoluciona) hacia la figura del más fuerte.

La teoría de la selección de las especies argumenta que la evolución de los seres vivos en la naturaleza, (de la cual nosotros somos fruto), se produce gracias a la competencia y la lucha entre los individuos. Los seres vivos evolucionan porque no pueden dejar de hacerlo: los seres vivos pueden tener o no esta actitud de competencia, tal vez puedan decidir si participar o no en estas luchas contra otros, pero en cualquier caso los que no luchen morirán o se reproducirán en menor grado, por lo que un comportamiento pacífico tenderá a desaparecer. En cuanto a la vida, la explicación más sencilla es que se creó al azar y que su objetivo es obtener individuos cada vez más perfectos y poderosos.

Factores Evolutivos: Mutación y Selección Natural

La **mutación**, es un cambio genético inesperado, que aparece en uno o varios individuos de una población. Como factor evolutivo, actúa a la manera de una propuesta de cambio. En la evolución, una mutación es un suceso bastante poco común. En la mayoría de los casos las mutaciones son letales, pero en promedio, contribuyen a la diversidad genética de la especie.

La **selección natural** es un proceso por el cual los efectos ambientales conducen a un grado variable de éxito reproductivo entre los individuos de una población de organismos con características, o rasgos, diferentes y heredables. Las características que inhiben el éxito reproductivo se hacen menos frecuentes de generación en generación.

El incremento resultante en la proporción de los individuos que son reproductores eficaces mejora, a menudo, la adaptación de la población a su ambiente. De esta

manera, la selección natural tiende a mejorar la adaptación al mantener aquellas adaptaciones que resultan favorables en un entorno estable (selección estabilizadora), o bien, al favorecer adaptaciones en la dirección adecuada ante cambios ambientales (selección direccional).
La selección, entonces, orienta los cambios genéticos de las poblaciones, provocando la expansión, retracción o eliminación de determinadas variantes genéticas. Como factor evolutivo, actúa a la manera de una política de decisiones.

La selección es un factor de cambio que está determinado por las necesidades del organismo y las ofertas ambientales. Consiste en una supervivencia y/o reproducción diferencial de los individuos, en función de ciertos rasgos que les otorgan más o menos aptitud. Aunque la selección actúa directamente sobre los fenotipos (no heredables) que portan esos rasgos, indirectamente ejerce una discriminación estadística sobre los genotipos (sí heredables).

La selección es una fuerza que orienta de modo predictible la evolución, y es el único factor que hace explicable la **adaptación**.

Competición y Cooperación

El comportamiento en el mundo animal es tan solo una característica fenotípica más de su código genético. El comportamiento está "programado" en sus genes. Esta circunstancia no impide que en los comportamientos animales podamos encontrar un alto grado de complejidad.

Aún admitiendo la aseveración anterior, entramos en paradojas que a priori resultan difíciles de resolver, muchas de ellas basadas en concepciones equivocadas de la evolución. Una de ellas es la aparición de comportamientos **altruistas**.

Para que se produzca evolución es necesario que se produzcan cambios en los individuos que van a reproducirse. Estos cambios modificarán el potencial de supervivencia de los mismos. Supongamos que los cambios producen variaciones en el comportamiento del individuo. Dicho cambio, para perdurar, debe suponer un beneficio suficiente para que el individuo pueda **reproducirse** el mayor número de veces posible.

El único objetivo de los genes es el de replicarse masivamente, necesitándose que el individuo que los posea tenga mayor éxito reproductivo que sus alelos. Así pues, si un gen quiere tener éxito deberá generar estrategias que conviertan al individuo

que los posea en un individuo mejor que el resto de su entorno. Estas estrategias tenderán a proteger la supervivencia y el éxito del individuo que las posee por encima de todo, son "**estrategias egoístas**".

Desde este punto de vista la aparición de estrategias no egoístas es sencillamente imposible.

En general, existirán estrategias egoístas, producidas por genes egoístas[24] y por otro lado existirán alelos de estos genes egoístas, que llamaremos genes altruistas que producirán "**estrategias altruistas**". Las estrategias egoístas son las que permitirán que el individuo sobreviva por encima de todo, y se transmitirán con una alta probabilidad. Son buenas estrategias desde el punto de vista del individuo. Las estrategias altruistas combinadas adecuadamente con las egoístas son buenas desde el punto de vista de la especie, estrategias colaboradoras.

1.2. Vida Artificial

Vida artificial es el nombre dado a la nueva disciplina que estudia la "vida natural" con la intención de recrear los fenómenos biológicos a través de computadoras y otros medios artificiales. La vida artificial complementa el aprovechamiento analítico de la biología tradicional con un enfoque sintético que intenta crear sistemas que se comporten como los organismos vivos.

La **ALife** (artificial life) [6], tienen como objetivo la aplicación del conocimiento sobre el funcionamiento y la dinámica del fenómeno que denominamos de manera genérica: vida, y su aplicación en múltiples alternativas de investigación y desarrollo extremadamente novedosas. Este intento de reproducir fenómenos biológicos de forma artificial intenta no solo conseguir una mejor compresión del fenómeno de la vida sino también aplicar los principios de la biología en tecnología, software, medicina, robótica, etc.

Los objetivos de la ALife fueron introducidos por **Christopher Langton** [7] como *"el estudio de la vida como podría ser y no como la vida es".*

Langton definió a la Vida Artificial como el *"estudio de sistemas hechos por el hombre que exhiben comportamientos característicos de sistemas de vida naturales, tales como auto-organización, reproducción, desarrollo y evolución"*.

La idea es por consiguiente, abstraer los principios subyacentes de las organizaciones de cosas vivas e implementar en una computadora de manera de ser capaces de estudiar esos principios y probarlos. Es decir que la gran meta es encontrar o extraer la forma lógica de los seres vivos.

En el campo, de la vida como debería ser, se considera a la vida como una propiedad de la organización de la materia, antes que una propiedad de la materia organizada[7].

Mientras que la biología está ampliamente interesada en las bases materiales de la vida, la vida artificial está interesada en las bases formales de la vida. La Biología comenzó su estudio de arriba hacia abajo, observando los organismos vivos como una máquina bioquímica compleja, y trabajando de manera analítica hacia abajo (Esta referencia se realiza hacia los órganos, tejidos, células, membranas, y finalmente moléculas) en busca de los mecanismos de vida.

Langton menciona que la ALife empieza de la parte de **abajo**, considerando a un organismo como una gran población de máquinas simples, y trabaja hacia **arriba** de manera sintética (de manera similar a la construcción de grandes agregados de objetos simples gobernados por reglas que interactúan una con otra de manera no lineal).

Podemos resumir las características esenciales de un modelo de la Vida Artificial en los siguientes puntos:

1. El modelo consiste en una población de individuos simples.
2. No hay un individuo especial que dirija a todos los demás agentes.
3. Cada individuo detalla la forma en que una entidad simple reacciona a las situaciones locales de su entorno, incluyendo encuentros con otras entidades.
4. No hay reglas en el sistema que dicten un comportamiento global.
5. Cualquier comportamiento en un nivel superior a los individuos es entonces emergente.

Diferencia de Enfoques: ALife - IA

Es importante en este punto realizar algunas consideraciones que de alguna manera esclarecen las diferencias a veces ignoradas pero por cierto existentes entre lo que se denomina Vida Artificial (Alife) y la Inteligencia Artificial (IA).

Haber cursado en el último año de la carrera la materia Inteligencia Artificial, sumado a la investigación presente sobre Vida Artificial, nos permite atrevernos a enumerar estos puntos o características en los cuales se pueden demostrar estas diferencias:

Característica	Vida Artificial (ALife)	Inteligencia Artificial (IA)
Objeto de Estudio	Vida	Inteligencia
Conocimiento	Surge de la interacción de agentes dedicados a una tarea común ➔ Interacción y relación entre agentes es lo importante.	No toma en cuenta la interacción. Se basa en conceptualización y razonamiento (inferencias). Se aparta al agente y se estudia su estructura sin tener en cuenta las relaciones e interacción.
Lema	*"No hay conocimiento sin interacción"*. Todo conocimiento es local y la verdad relativa.	Conocimiento acumulado en un agente. Verdad absoluta.
Representación del Conocimiento	Conexiones. Sistemas reactivos que aprenden y tienen comportamiento adaptativo, conectando las percepciones directamente con las acciones.	Centralizado. Cognición simbólica.
Paradigma	**Bottom-Up**. De la interacción de componentes simples "emerge" un comportamiento global complejo no programado. **Método Sintético**.	**Top-Down**. Se analiza el problema descomponiéndolo en sub-problemas. **Método Analítico**.
Ciencias Relacionadas	Biología, Física, Sociología, Química.	Ingeniería,psicología, Filosofía.

Tabla 1 – Diferencias de enfoques entre Alife e IA

La ALife trata las características inesperadas. Se crea una característica inesperada cuando algo se convierte en algo **"más que lo que es la suma de sus partes".** Un ejemplo extremo de una propiedad que conozcamos es la vida en este planeta. Somos más que la suma de nuestras partes.

Una mitad de un ser humano no funciona sin la otra mitad, pero en su totalidad somos capaces tener un comportamiento complejo. La vida artificial es un intento de explicar la existencia de la vida así como la de crear algo nuevo, esto no tiene

que ser confundida con inteligencia artificial, estos campos tienen pocas cosas en común.[6]

Ejemplos de Sistemas de Vida Artificial

El problema de las presas y los predadores, "la caza"

Uno de los ejemplos más conocidos es el problema de "la caza" o también conocido como problema de los "predadores y presas". Los agentes, esto es, las presas (herbívoros), como los predadores (carnívoros) se mueven en un espacio representado en la forma de grilla. El objetivo del juego es para los predadores capturar a las presas rodeándolas para poder comérselas. El problema consiste en coordinar las acciones de los predadores para, de esta manera, puedan rodear a una presa tan rápido como sea posible, una presa rodeada se considera muerta. Lo interesante en este juego es que el problema está bien definido y aún deja abiertos muchos posibles modos de cooperación. Este ejemplo es bueno para la diferenciación de las aproximaciones cognitivas y reactivas.

Las estrategias cognitivas generalmente proceden de una manera top-down definiendo las diferentes funciones que el sistema puede ejecutar: detección de presas, fijar los equipos o grupos de predadores, alocación de roles (tomar a la presa por el norte, oeste, este o sur), reorganización de los equipos si la distribución de los cazadores no es la correcta. Todas estas funciones se definen bien y se implementan en los agentes en la forma de comportamiento adaptado que requiere un sistema de comunicación que permita el diálogo y la distribución de la decisión. Podemos suponer que las criaturas tienen objetivos y que actúan racionalmente en relación con esos objetivos señalando una criatura líder si es necesario para organizar la distribución del trabajo y coordinar las acciones.

La creación de sistemas reactivos para resolver este problema comienza por una aproximación totalmente diferente. Suponemos que los animales presas emiten una señal, cuya intensidad decrece en proporción a la distancia, y la cual juega un rol de atractor de los predadores. Entonces cuanto más cerca esté un predador de una presa más fuerte será la recepción de la señal.

Además, para que los cazadores no se encuentren todos juntos en el mismo lugar, y que puedan rodear efectivamente a una presa, acordamos que los predadores emiten una señal que actúa de repelente para otros predadores. De este modo cada predador es a la misma vez atraído por las presas y rechazado por otros

predadores. La captura de las presas de esta manera emerge de la combinación de estas diferentes reacciones a los estímulos emitidos por las presas y por los predadores.

Hormigas y Plantas

En este programa vemos un mundo de dos dimensiones en el que conviven hormigas y plantas. Las plantas crecen cuando son regadas. Las hormigas pueden comer plantas, moverse, regar plantas, pelearse y reproducirse. Las hormigas ganan energía cada vez que comen una planta. Cuando una planta ha sido suficientemente regada, será comida por la primera hormiga que pase junto a ella. No hay límite en la cantidad de plantas que una hormiga puede comer. Las hormigas pierden energía cada vez que se mueven, riegan, se pelean o se reproducen. Si una hormiga pierde toda su energía, muere. Por otra parte, cada cierto tiempo nace una planta espontáneamente.

En un combate, la hormiga ganadora se quedará con la mitad de la energía (comida) acumulada por la perdedora. Además, cuantas más veces salga vencedora una hormiga, más probabilidades tendrá de salir victoriosa de la siguiente pelea.

En primer lugar si la hormiga no posee comida, la declaramos muerta. En caso contrario, la hormiga observa si existen plantas u otras hormigas a su alrededor. Si existen plantas comestibles, se las come. Después, podrá regar las plantas que aún no son comestibles, pelear o reproducirse con sus vecinos, o simplemente seguir paseando por su mundo virtual.

En este programa tenemos 5 tipos de hormigas. La probabilidad de que una hormiga realice una determinada acción dependerá del tipo de hormiga y de los individuos con que se encuentre. Cuando dos hormigas se reproducen, se crea una nueva hormiga con características similares a las de los padres. Por ejemplo, una hormiga de tipo 1 (roja) y una hormiga de tipo 5 (verde) generarán con gran probabilidad un descendiente de tipo 3 (naranja).

Además, para que la sociedad no se estanque en un único tipo de hormiga, se permitirá, por ejemplo, que dos hormigas verdes produzcan una roja; eso sí, con una probabilidad mucho más baja. La *tabla Nº 3* ejemplifica estadísticamente el comportamiento. La hormiga roja es la más egoísta. Nunca riega, y cuando se encuentra con otra hormiga, se dedica a pelear o a reproducirse. La hormiga verde es la menos egoísta, no lucha nunca y se dedica a regar todas las plantas que

encuentra a su paso. El resto de hormigas son términos medios entre estos dos extremos. Es evidente que las hormigas del tipo verde son la base sobre la que se sustenta la sociedad. Es decir, si no hay quien riegue las plantas, éstas no crecen, la comida se termina y las hormigas mueren. [9]

Tipo de Hormiga	¿Hay Planta?	¿Hay Hormiga?	Mover	Regar	Luchar	Reproducirse
Roja	No	No	100%	0%	0%	0%
	No	Si	0%	0%	80%	20%
	Si	No	100%	0%	0%	0%
	Si	Si	0%	0%	80%	20%
Rosa	No	No	100%	0%	0%	0%
	No	Si	20%	0%	60%	20%
	Si	No	75%	25%	0%	0%
	Si	Si	20%	20%	40%	20%
Naranja	No	No	100%	0%	0%	0%
	No	Si	40%	0%	40%	20%
	Si	No	50%	50%	0%	0%
	Si	Si	20%	30%	30%	20%
Amarilla	No	No	100%	0%	0%	0%
	No	Si	60%	0%	20%	20%
	Si	No	25%	75%	0%	0%
	Si	Si	20%	40%	20%	20%
Verde	No	No	100%	0%	0%	0%
	No	Si	80%	0%	0%	20%
	Si	No	0%	100%	0%	0%
	Si	Si	0%	80%	0%	20%

Tabla 2 – Plantas y Hormigas

1.3. Sistemas Multi-Agente

Agente

Los **agentes** están por todas partes. La gente se encuentra diariamente con agentes inteligentes, agentes de información, agentes móviles, agentes asistentes personales, y algunos tipos más de agentes. Uno puede preguntarse si se puede sacar algo en claro de esta aparente anarquía. ¿Qué es lo que caracteriza a un agente? ¿Tienen algo en común? ¿Cómo podemos organizarlos para llevar a cabo tareas? [10]

Afortunadamente, los investigadores ya han definido algunos conceptos de gran ayuda. En primer lugar, es comúnmente aceptado que los agentes son entidades

dentro de un entorno, y que pueden sentir y actuar (no necesariamente en este orden). Esto significa que los agentes no son entidades aisladas, sino que son capaces de comunicarse y colaborar con otras entidades. Simplemente, los agentes que no sean capaces de trabajar conjuntamente con otros agentes (humanos incluidos) están destinados a ser virtualmente inútiles.

Una vez que los agentes están a punto para empezar a colaborar, necesitan encontrar otros agentes con los que llevar a cabo la colaboración. Ésta es una tarea fácil si saben exactamente qué agentes contactar y dónde contactarlos. No obstante, una distribución estática de los agentes es muy difícil que exista: la gente está normalmente en movimiento y no siempre está disponible para interactuar con los demás. Esto también es verdad para los sistemas multi-agentes dinámicos: los agentes necesitan soporte para encontrar a otros agentes.

El término agente es difícil de definir. Los agentes son normalmente definidos como entidades con atributos considerados útiles en un dominio particular. Éste es el caso de los agentes inteligentes, donde los agentes son vistos como entidades que emulan procesos mentales o simulan un comportamiento racional; asistentes personales, donde los agentes son entidades que ayudan a los usuarios a realizar una tarea; agentes móviles, donde las entidades son capaces de vagar por una red para conseguir sus objetivos; agentes de información, donde los agentes filtran y organizan de forma coherente datos dispersos y no relacionados; y agentes autónomos, donde los agentes son capaces de cumplir acciones no supervisadas.

Un aspecto de los agentes que es ampliamente mencionado en la literatura es la noción de agente como una entidad interactiva que existe como parte de un entorno compartido con otros agentes. Esta definición de agente tomada de las descripciones dadas por varios autores, que describen los agentes como entidades conceptuales que perciben y actúan de una forma reactiva en un ambiente dónde otros agentes existen y interactúan basados en una comunicación y representación del conocimiento común.

La gran cantidad de definiciones sobre agente obliga a los autores de esta tesis a restringir o acotar un marco de definiciones que sea coherente con nuestra idea formada acerca de que es un agente y acerca del tipo de agentes que necesitamos para modelar en nuestro Framework. Es por eso que a continuación definiremos al agente y haremos referencia al mismo de acuerdo con las conclusiones a las que hemos abordado sobre el tema.

De esta manera, un Agente es una entidad física o virtual, posee las siguientes características:

- Actúa en un medio ambiente.
- Puede comunicarse directamente con otros agentes.
- Está manejado por un conjunto de tendencias (en la forma de objetivos individuales o funciones de satisfacción / supervivencia las cuales trata de optimizar).
- Posee recursos propios.
- Es capaz de percibir su medio ambiente (pero en una extensión limitada).
- Tiene una representación parcial de su entorno (quizá ninguna).
- Posee habilidades y ofrece servicios.
- Su comportamiento tiende hacia la satisfacción de sus objetivos. Tomando en cuenta los recursos y habilidades disponibles y dependiendo de su percepción, su representación y la comunicación que recibe.

Los agentes son capaces de **actuar**, y no sólo razonar, como los sistemas clásicos de la IA. El concepto de acción está basado en el hecho que los agentes llevan a cabo acciones que modificarán el entorno de los agentes, y de este modo las decisiones futuras. También se pueden comunicar con otros agentes, y este es uno de los caminos principales de interacción de los agentes.
Los agentes están dotados de autonomía. Esto quiere decir que no están comandados directamente por el usuario (u otro agente), pero si por un conjunto de tendencias, las cuales pueden tomar la forma de objetivos individuales a ser cumplidos o funciones de satisfacción o supervivencia las cuales trata de optimizar. El agente está activo, puede acceder o rechazar pedidos de otros agentes.

Los agentes tienen una representación parcial de su entorno, esto es, no tienen una percepción global de lo que sucede a su alrededor. Y este es el caso de cualquier emprendimiento humano de gran escala (la fabricación de un avión por ejemplo),

en el cual nadie sabe todos los detalles del proyecto, cada especialista tiene sólo un punto de vista parcial que corresponde al área de su especialidad.

Agentes Cognitivos y Reactivos

Estos dos conceptos se deben a dos escuelas de pensamiento. La primera, la escuela 'cognitiva', está representada en el dominio de la inteligencia artificial distribuida (DAI), ya que sus orígenes descansan en el enfoque de la comunicación y cooperación asociados a los sistemas expertos clásicos. En este enfoque un sistema multi-agente está constituido de un numero pequeño de agentes 'inteligentes'. Cada agente tiene una base de conocimiento disponible que comprende todos los datos y el know-how requerido para llevar a cabo la lista de tareas y las interacciones con los otros agentes y su entorno.
En contraposición, la otra tendencia, la escuela 'reactiva' afirma que no es necesario que los agentes sean inteligentes individualmente para que el sistema global se comporte de manera inteligente. Mecanismos de reacción a eventos pueden resolver problemas categorizados como complejos.

Los agentes reactivos pueden ser dirigidos por mecanismos de motivación que lo hacen avanzar hacia el cumplimiento de una tarea, como por ejemplo: satisfacer una necesidad interna (como por ejemplo, mantener su nivel de energía). Estos agentes pueden por ejemplo ser construidos de manera que respondan sólo a estímulos del ambiente, que sus comportamientos sean guiados íntegramente por el estado local del mundo en el cual están inmersos.

Los agentes cognitivos, debido a su sofisticación y su capacidad de razonamiento sobre el entorno que lo rodea, pueden operar en una forma relativamente independiente. Las tareas que ejecutan son complejas con relación a la mayoría de las facultades de los agentes reactivos. Pueden, de esta manera, resolver problemas complejos en una forma individual. Sus representaciones internas y los mecanismos de inferencia disponibles le permiten funcionar en forma independiente de otros agentes, y le dan gran flexibilidad en la expresión de su comportamiento.

Por el contrario, la estructura de los agentes reactivos impone comportamientos más rígidos. Por ésta razón, los agentes reactivos no son potentes en forma individual. Pero su potencia surge de la capacidad que poseen de formar grupos, esto es, colonias capaces de adaptarse a su entorno. Por esto los agentes reactivos son interesantes, no tanto en el nivel individual sino en el nivel de poblaciones y

con respecto a las capacidades de adaptación y evolución la cual emerge de las interacciones entre los miembros.

Sistema Multi-Agente

Habiendo definido ya que es para nosotros un agente, podemos entonces referirnos a su comportamiento y a la interacción entre ellos. Así, el próximo paso será describir lo que entendemos por Sistema Multi-Agente.

Los sistemas multi-agente obedecen a los principios que gobiernan el fenómeno caótico[25]. Cualquier modificación de las condiciones iniciales, aunque sean infinitesimales, cualquier introducción de variables aleatorias, aunque sean restringidas, tiene efectos que son amplificados por la interacción entre los agentes, y no es posible predecir cual será el estado preciso de los agentes luego de transcurrido un cierto tiempo.
Esto no quiere decir que no hay leyes que gobiernan estos tipos de sistemas. Al contrario, es posible observar efectos de la auto-organización, y efectos que causan la aparición de estructuras emergentes. De hecho, con un poco de experiencia, es posible construir sistemas que produzcan un comportamiento deseado en una situación estable, mientras se demuestra la capacidad de adaptación a disturbios que no fueron previstos al comienzo.

Se han propuesto varias definiciones, procedentes de diferentes disciplinas, para el término sistema multi-agente (MAS: Multi-Agent System). Una de las primeras definiciones dadas indica que un sistema multi-agente es una red poco acoplada de entidades capaces de solucionar problemas, que trabajan conjuntamente para encontrar respuesta a problemas que están más allá de la capacidad y el conocimiento individual de cada entidad[11]. Más recientemente, se ha dado al termino sistema multi-agente un significado más general, y actualmente es usado para definir todos los tipos de sistemas compuestos por múltiples componentes autónomos que poseen las siguientes características[12]:

- Cada agente tiene capacidad para solucionar parcialmente el problema
- No hay un sistema global de control
- Los datos no están centralizados
- La computación es asíncrona

Uno de los factores actuales (y sin discusión uno de los más importantes) que promueven la investigación en MAS, es la creciente popularidad de Internet, que

proporciona la base para un entorno o ambiente abierto donde los agentes interactúan para conseguir sus objetivos individuales o colectivos. Para interactuar en este entorno, los agentes tienen que superar dos problemas: tienen que ser capaces de encontrarse (los agentes pueden aparecer, desaparecer o moverse en cualquier momento) unos a otros; y tienen que ser capaces de interactuar[13].

Es necesario un mecanismo para avisar, encontrar, fusionar, usar, presentar, gestionar y actualizar los servicios y la información de los agentes. Se propuso la noción de agentes intermedios (middle agents)[14] para proporcionar estas necesidades. Los agentes intermedios son entidades a las que otros agentes comunican sus capacidades y que ni proveen ni solicitan servicios desde el punto de vista de la transacción que se está considerando en ese momento. La ventaja de los agentes intermedios es que permiten a los MAS operar de una forma robusta a pesar de tener que afrontar la aparición, desaparición y movilidad de los agentes.

Interacciones entre Agentes

Para un agente, interactuar con otro es por un lado la fuente de su poder y el origen de sus problemas. Para un agente, el otro es simultáneamente lo mejor y la peor de las cosas.

Manejar el problema de la interacción significa proveer a un agente no sólo con la descripción de los mecanismos necesarios que permiten a una agente interactuar, sino también analizar y diseñar las diferentes formas de interacción que un agente puede ejecutar para completar sus tareas y cumplir sus objetivos. Primero de todo los agentes tienen que ser capaces de utilizar la comunicación no sólo para transmitir información sino, más importante, para inducir un comportamiento específico en otros agentes. Comunicarse es entonces una manera de acción la cuál en vez de aplicar un cambio en el ambiente, tiende a modificar el estado mental del receptor. Por ejemplo, pedirle a alguien que lleve a cabo tareas tiende a crear en los demás una intención para completar la tarea, y de esta forma constituye un camino para conseguir un objetivo sin realizar la tarea uno mismo.

Las diferentes formas de interacción son: colaboración y coordinación de acciones.
En los sistemas multi-agente la forma más común de interacción estudiada es la cooperación. Es posible hablar de cooperación en ambos sentidos: agentes cognitivos y reactivos si sólo miramos los resultados de las acciones y no las intenciones de los agentes.

Cooperación = Colaboración + Coordinación de acciones + Resolución de conflictos

La capacidad de interacción es una de las más importantes de los agentes[15]. Dicho de otra forma, los agentes interaccionan de forma recurrente para compartir información y realizar las tareas para conseguir sus objetivos. Los investigadores en lenguajes de comunicación entre agentes mencionan tres elementos clave para conseguir la interacción multi-agente[16]:

- Un lenguaje y un protocolo de comunicación común.
- Un formato común del contenido de la comunicación

El término 'sistemas multi-agente' (o MASs) se aplica a sistemas que comprenden los siguientes elementos:

- Un entorno o ambiente, E, esto es, un espacio que generalmente tiene volumen.

- Un conjunto de objetos, O. Estos objetos están situados, es decir es posible asociar en un determinado momento a cualquier objeto con una posición en E. Estos objetos son pasivos y pueden ser percibidos, creados, destruidos y modificados por los agentes.

- Un grupo de Agentes, A. Son objetos específicos ($A \subseteq O$), representan las entidades activas del sistema.

- Un conjunto de relaciones, R. Relaciona objetos (y de esta manera, agentes) con otros objetos.

- Un conjunto de operaciones, Op, hacen posible a los agentes de A percibir, producir, consumir, transformar y manipular objetos de O.

- Operadores con la tarea de representar la aplicación de estas operaciones y la reacción del mundo ante estos intentos de modificación, los cuales daremos en llamar las leyes del universo.

Cuando los agentes están situados, E generalmente es un espacio métrico, y los agentes son capaces de percibir su entrono, esto es, reconocer objetos situados en el medio ambiente a través del funcionamiento de sus capacidades de percepción y

acción, es decir, de transformación del estado del sistema modificando la posición de, y las relaciones existentes entre los objetos.

La mayoría de los sistemas multi-agente reactivos consideran que el concepto de entorno o hábitat es fundamental para la coordinación de las acciones entre varios agentes.

La creación de sistemas multi-agente requiere la definición simultánea de la estructura de los agentes y su entorno o hábitat, las acciones que los agentes deben llevar a cabo dentro del entorno.

Arquitecturas de Agentes

Los agentes constituyen caso especial de organizaciones. El termino generalmente empleado para describir la organización interna de un agente es el de arquitectura, por su analogía con la estructura de las computadoras. Un agente puede efectivamente ser comparado con una computadora, donde sus órganos de percepción y acción y su sistema de deliberación puede parecerse a la entrada-salida, a la memoria de almacenamiento y a las unidades de computación de una computadora.

Es claro que un agente cognitivo tiene una arquitectura más elaborada que un agente reactivo. En la arquitectura de un agente, encontramos parámetros generales que son válidos para el análisis estructural de organizaciones artificiales: tipo de metodología, estructuras de subordinación, pareamiento y constitución.
En medio de la totalidad de las posibles arquitecturas que se obtienen por la variación de estos parámetros, solo un pequeño número a dado origen a implementaciones que sean capaces de categorizarse. Así podemos hablar de arquitecturas basadas en módulos horizontales, arquitecturas basadas en tableros, arquitecturas basadas en subsumpción, arquitecturas basadas en tareas competitivas y arquitecturas basadas en sistemas de producción.

Arquitectura Modular Horizontal

La mayor parte de las arquitecturas propuestas para la definición de agentes cognitivos se basan en el concepto global de módulos horizontales enlazados por conexiones pre-establecidas.

Las arquitecturas modulares horizontales son concebidas como un conjunto de módulos, cada uno efectuando una función horizontal específica.
Los módulos mas ampliamente usados son:

- Funciones de motor y perspectiva
- Envío e interpretación de comunicaciones
- Creencias basadas en la comprensión del ambiente y de otros agentes
- Manejo de obligaciones
- Experiencia en el dominio de habilidades
- Manejo de objetivos y tareas de decisión
- Planeamiento de acciones

En este tipo de arquitecturas todos los enlaces son fijos, esto es, el modo de circulación de datos está predefinido por el diseñador.

En la fase ascendente, las señales ingresan desde el exterior a través de los sensores y son filtradas de manera de obtener información cada vez más abstractas hasta poder integrarse con el modelo del agente. La función más alta es efectuada por el módulo de decisión de tareas, que decide de acuerdo a los datos recibidos y a los objetivos propios. La fase descendiente corresponde a la implementación de decisiones. El módulo planificador determina las acciones que necesitan ser efectuadas para lograr el objetivo seleccionado. Estas se transmiten luego al módulo de ejecución.

La siguiente figura muestra un ejemplo típico de esta arquitectura, caracterizada por un flujo de datos que primero asciende y luego desciende:

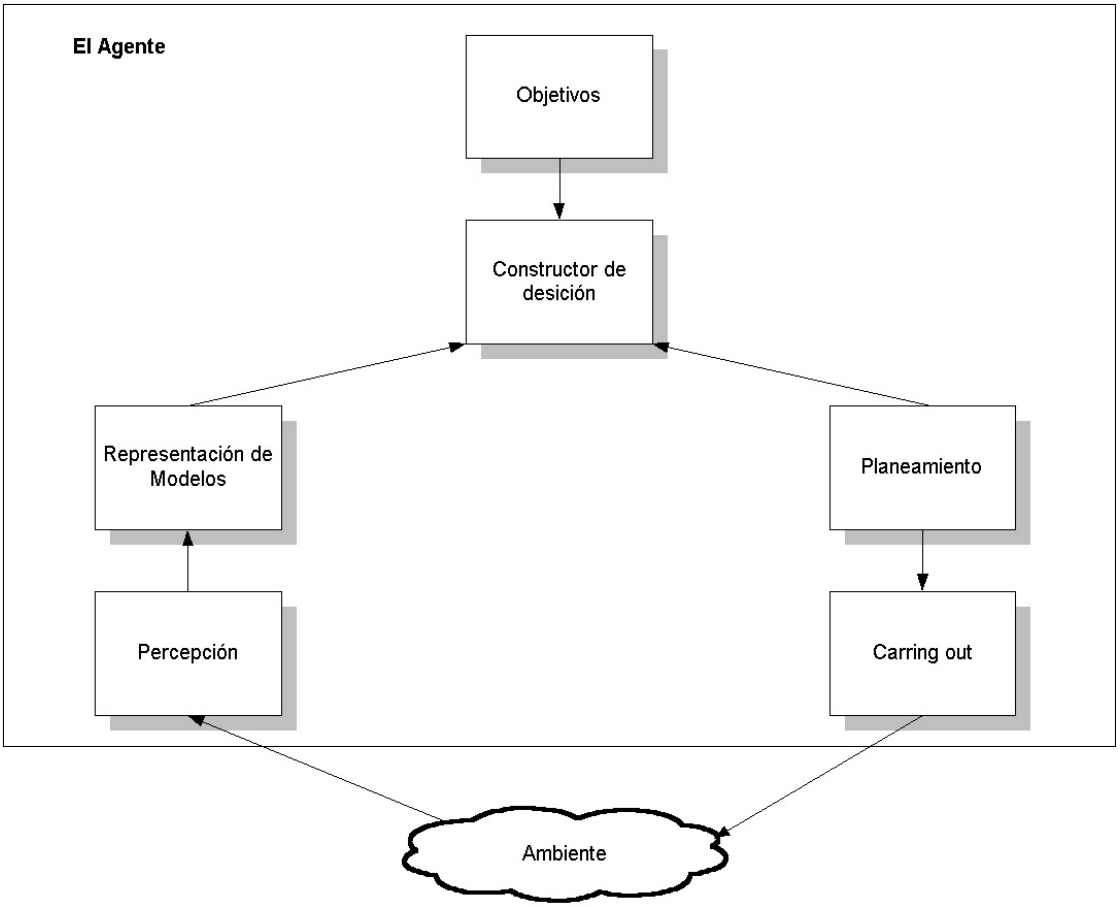

Figura 3 – Arquitectura Modular - Horizontal

Arquitectura basada en tablero

La arquitectura basada en tablero es una de las más ampliamente usadas en sistemas multi-agentes cognitivos.

Este modelo está basado en la división de módulos independientes que no se comunican datos directamente pero que actúan recíprocamente e indirectamente por compartir datos. Estos módulos, llamados Base de Conocimiento o KSs (Knowledge Sources), trabajan en un espacio que incluye todos los elementos requeridos para resolver un problema.

La arquitectura de un sistema basado en tablero comprende tres subsistemas:

- ***Bases de Conocimientos***

- La ***base compartida (el tablero),*** que comprende el estado parcial de un problema en el proceso de solución – hipótesis, resultados intermedios, y, en forma más general, todos los intercambios de datos entre las bases de conocimiento. Estas bases son separadas en jerarquías (conceptual, partitiva, causal, etc.), que estructuran el modelo del área de aplicación como el espacio para hipótesis / soluciones.

- Un ***dispositivo de control***, que maneja conflictos de acceso entre bases de conocimiento. Esto último interviene de una manera "oportunista", es decir, sin ser provocado por un sistema de control centralizado.

En esta arquitectura, el control debe, en esencia, decidir que se hará en lo inmediato, esto es, determinar que base de conocimiento necesita ser activada.
Por su mecanismo de control centralizado, su carencia de memoria local y su manera de localidad de datos, estos sistemas son respetados actualmente como una arquitectura práctica para la creación de sistemas inteligentes, y más particularmente, para implementar las estructura interna de agentes cognitivos.

La principal desventaja de esta arquitectura es su relativa ineficiencia, debido al carácter altamente expresivo de su control.

En la Figura Nº 4 se muestra un ejemplo esquemático de la arquitectura basada en Tablero.

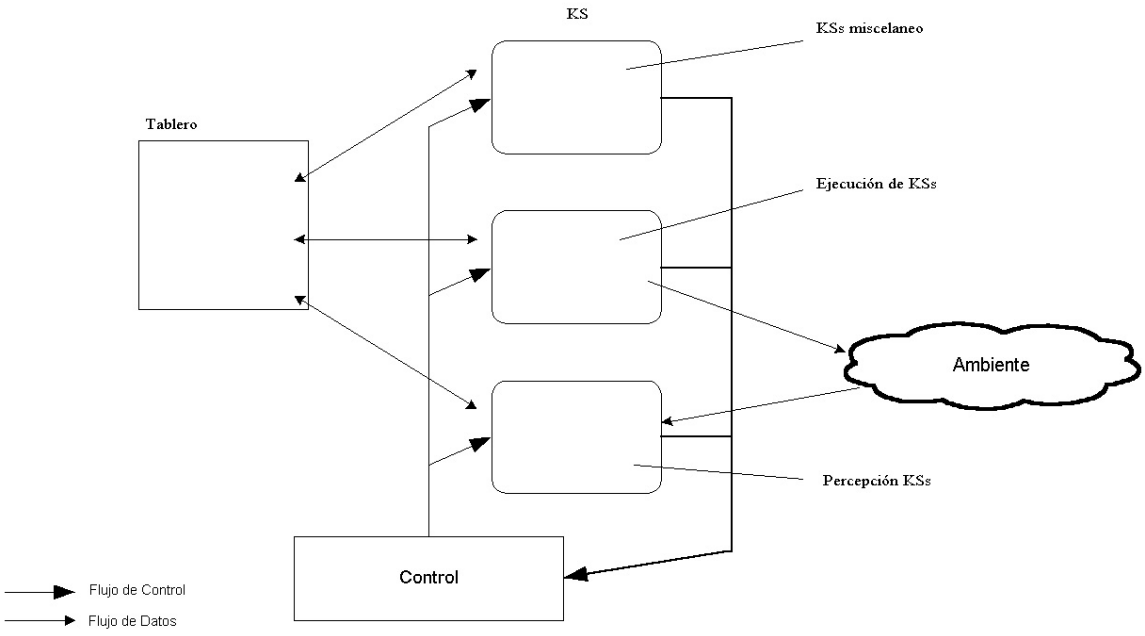

Figura 4 – Arquitectura Basada en Tablero

Arquitectura de Subsumpción

En contraste a la arquitectura modular jerárquica, que divide un agente en módulos horizontales, la arquitectura de subsumpción divide un agente en módulos verticales, cada uno de ellos responsable para un muy limitado tipo de comportamiento.

Las interacciones entre módulos son fijas, y se efectúan por intermedio de una relación de dominio, definidas en la etapa de diseño. Los módulos llevan a cabo sus tareas en paralelo, pero si dos módulos están en conflicto (es decir, producen resultados contradictorios), entonces sólo los datos proporcionados por el modulo dominante serán tenidos en cuenta. Si el módulo inferior produce un resultado, mientras el módulo dominante no está trabajando, este efecto será tenido en cuenta, pero si el módulo dominante también produce datos, esos último tendrán prioridad.
Por lo tanto es posible construir sistemas relativamente complejos usando esta arquitectura. Sin embargo, una vez más es el diseñador quien define la

construcción de los módulos y las relaciones de dominio existentes entre ellos. La siguiente figura muestra un ejemplo característico de la arquitectura de subsumpción para un robot explorador.

Esta técnica es utilizada principalmente para describir agentes reactivos, aunque con algunas consideraciones especiales podría servir también para la descripción de agentes cognitivos.

En la Figura 5 se observa un modelo de Robot explorador basado en arquitectura de subsumpción.

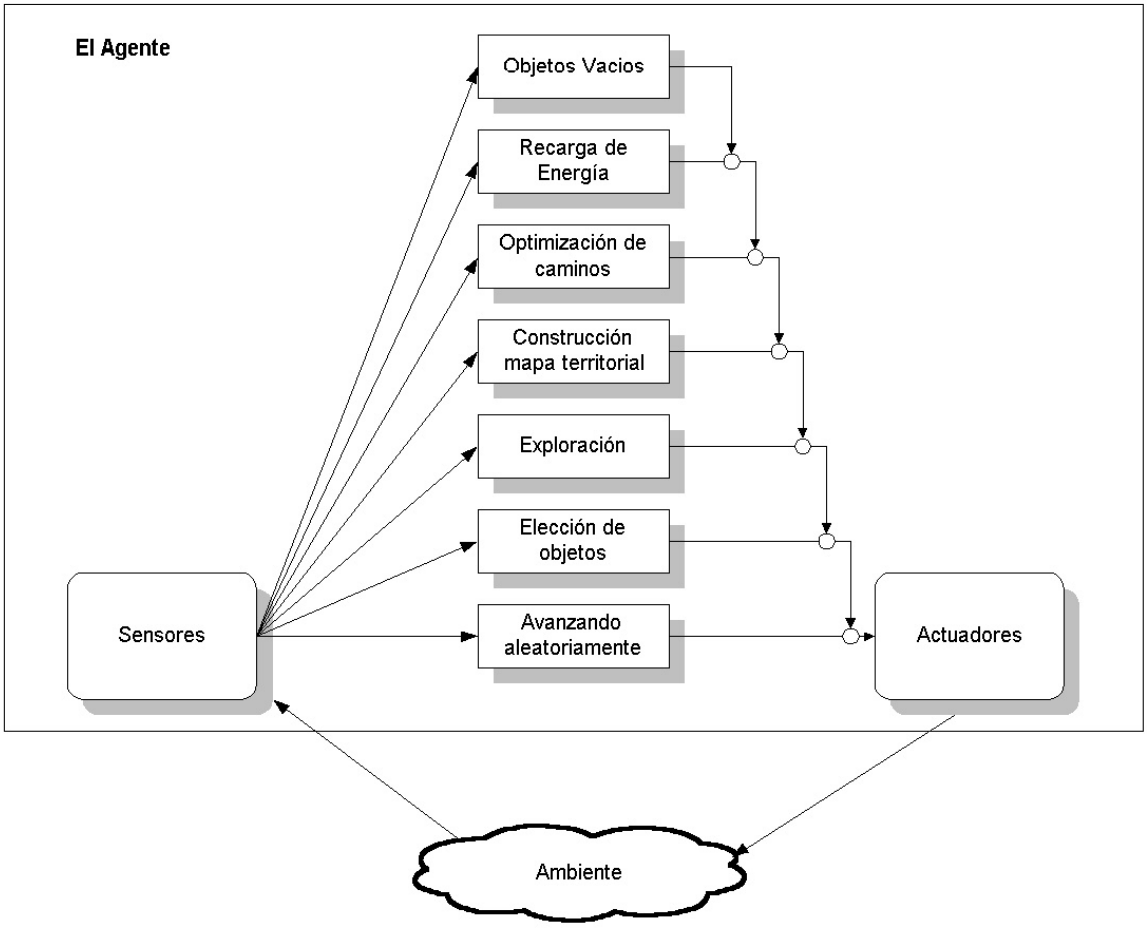

Figura 5 – Arquitectura de Subsumpción

Tareas Competitivas

Mientras que en los dos arquitecturas modulares previas los enlaces son fijos, otras arquitecturas se han propuesto intentando introducir una cierta mutabilidad en los enlaces entre módulos, además de obtener mayor flexibilidad en la elección de una selección de módulo. Estos módulos representan funciones verticales, a veces llamadas comportamientos o tareas.

En la estructura de tareas competitivas, un agente está construido de un estructura de tareas, donde sólo una de ellas puede ser activada en un mismo instante de tiempo. Estas tareas se encuentran en competición para su selección a través de un mecanismo de decisión que depende de varios parámetros: el peso acordado para la tarea en un momento dado, el contexto de la aplicación, la entrada de datos desde el exterior, etc. El módulo de selección recepciona la lectura para los varios parámetros de tareas desde cada una de las ellas, y selecciona aquella tarea que resulte tener el valor más alto de la función de evaluación aplicada a sus parámetros.

Una vez seleccionada, la tarea es activada, y la tarea actual es desactivada. Pero una desactivación de la tarea no significa eliminar definitivamente la tarea, sino que permanece para continuar en la competición, ejercitando una actividad de monitoreo que consiste en recibir datos desde los sensores y calcular los valores de los parámetros de selección que dependen de ella.

Otras arquitecturas se han propuesto con mas o menos el mismo principio. Todas ellas proponen un mecanismo de selección de la acción para agentes reactivos para organizar su comportamiento, en respuesta a los estímulos del ambiente que reciben y a los manejos internos que los motivan (necesidad de energía, reproducción, etc.).

En la siguiente figura se observa un esquema que describe la Arquitectura de Tareas Competitivas.

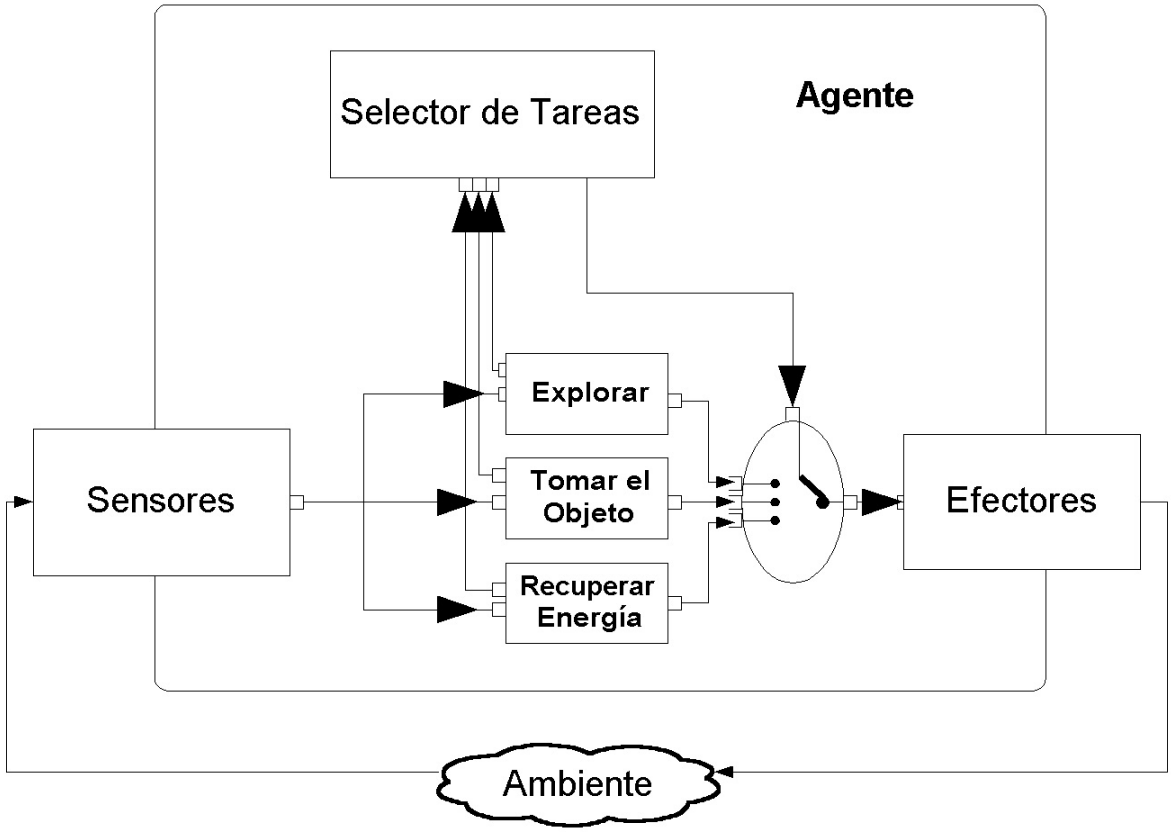

Figura 6 – Arquitectura de Tareas Competitivas

Sistemas de Producción

Un sistema de producción (o sistema basado en reglas) se define como la combinación de una base de datos, una regla de producción base y un intérprete, el motor de inferencia. Auque existe una amplia variedad de sintaxis para la definición de reglas de producción, generalmente siguen la siguiente forma:

If <lista de condiciones> then <lista de acciones>

Donde *<lista de condiciones>* se asocia con los elementos de la base de datos y *<lista de acciones>* abarca acciones elementales tales como agregar o eliminar elementos de la base de datos.

Algunas acciones pueden activar comandos de ejecución para el agente directamente. Cuando una regla puede validar alguna condición de la lista, ejecuta

la acción correspondiente. Si varias reglas pueden ser activadas, entonces se dice que están en conflicto, y que es allí donde el motor de inferencia activa la regla con la mayor prioridad.

En el contexto de un sistema multiagente, cada agente es representado en la forma de un sistema de producción, equipado con funciones de interpretación y ejecución.

En la siguiente ilustración se muestra el esquema que representa la arquitectura basada en Sistemas de Producción.

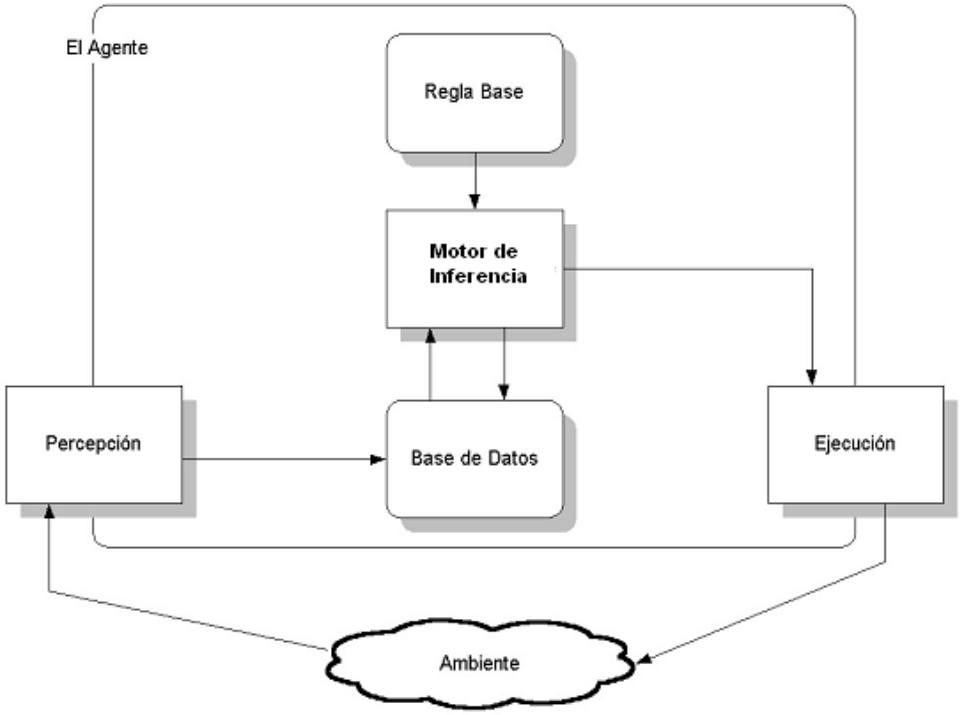

Figura 7 – Arquitectura basada en Sistemas de Producción

1.4. Sistemas Complejos

El término de "**Sistema complejo**" y "**Complejidad**" se prestan a muchas confusiones. Según algunos autores hay hasta 30 definiciones diferentes del concepto de complejidad. Por ejemplo para el padre de la "vida artificial'' Chris Langton, la Complejidad es la ciencia que trata de describir los sistemas que se encuentran en el "borde del caos''. Algunas definiciones aunque interesantes son algo oscuras y suelen formularse desde un punto de vista específico y están dirigidas hacia alguna rama en concreto de la ciencia. El concepto de "Complejidad" parece que no es lo mismo para un informático, para un físico, un biólogo o un economista.

Supóngase ya un sistema complejo. Por más que se conozcan acabadamente todos sus subsistemas constituyentes, en realidad ello no alcanza para conocer la función verdaderamente fundamental del sistema. ¿Por qué? Porque el sistema no es solamente suma ingenua de sus partes. Para ser un sistema, debe ser más que la suma trivial. Una bicicleta y un ciclista forman un sistema con una función más allá de la suma de las funciones aisladas de ambos subsistemas. Cuanto mayor sea la diferencia entre la suma ingenua de las partes y la operación total, tanto más significativa es la complejidad del sistema. Los sistemas complejos muestran una dinámica propia que los hace acercarse a (aunque también alejarse de) diversos ordenamientos posibles, ordenamientos que implican a veces autoorganizaciones inesperadas para un observador desinformado.

La complejidad, con sus aportes característicos, enriquece todo lo que se sabe de casos simples.

Si intentamos adoptar un punto de vista más genérico y, a su vez más estricto, un sistema complejo puede definirse como un sistema compuesto por una gran cantidad de componentes simples que interaccionan localmente, intercambian información con el entorno y tienen capacidad para adaptar su comportamiento como consecuencia de las interacciones, dando lugar a algún tipo de comportamiento global que no se puede explicar a partir de las propiedades de un sólo individuo sino como resultado de las interacciones de un colectivo. El elemento fundamental radica en que las interacciones simples entre los componentes pueden generar comportamientos complejos en el sistema en su conjunto. En consecuencia nos encontramos con el paradigma consistente en que la complejidad del conjunto es más que la suma de los componentes.

El problema es que muchos de los sistemas complejos no pueden ser analizados matemáticamente, pudiendo sí ser estudiados mediante experimentos numéricos. Los sistemas que tienen esa característica son denominados **irreducibles**, siendo a través de **simulaciones** la única manera de estudiar su evolución.

La técnica radica en diseñar los componentes, especificando la estructura y las interacciones posibles junto a las funciones de comportamiento ante cada situación considerada. Luego se procede a conectar el comportamiento de los componentes siguiendo alguna lógica de ordenamiento temporal. Finalmente se hace evolucionar la simulación y se estudian los resultados tratando de encontrar alguna dinámica emergente.

En este punto del análisis resulta de interés especificar las características básicas que hacen que un sistema sea calificado como "complejo". En este sentido hablamos de sistemas complejos cuando nos encontramos, como ya dijimos, con sistemas que poseen gran cantidad de componentes, cuyas interacciones tienen características aleatorias lo que genera desequilibrios en cuanto a que no todos los componentes logran su comportamiento óptimo (fenómeno de frustración). Estas características hacen que no sea esperable que el sistema llegue a un óptimo global en el cual las interacciones no generen cambios en los componentes. De aquí se deduce que la dinámica general de un sistema complejo evoluciona en constante desequilibrio empujado por las funciones de comportamiento hacia un equilibrio global cambiante con la evolución de las interacciones. Asemejándose tal comportamiento a una dinámica que bordea el caos. Por esto último nos referiremos más adelante a lo que es llamado Teoría del Caos.

Algunos ejemplos de sistemas "complejos" son:

- El cerebro
- Un ordenador
- Un hormiguero
- Un sistema social ó económico

Complejidad en Biología

Atendiendo mas a nuestras intenciones en esta tesis, es importante observar la complejidad desde aquellos campos a los que nos dedicaremos en el trabajo.

Gran parte de la Vida Artificial que se desarrolla hoy día se asienta en trabajos precedentes de la biología teórica y se enmarca en el terreno más amplio de las ciencias de la complejidad.[17]

El sistema viviente más simple que conocemos, la célula bacteriana, es una pequeña maquinaria de extremada complejidad y eficacia. La biología está repleta de sistemas complejos: los miles de genes regulándose unos a otros en el interior de una célula, las redes de células y moléculas que sirven de intermediarios en la respuesta inmune, los casi millones de millones de neuronas que integran a las redes neuronales que son las que gobiernan al comportamiento y al aprendizaje, las tramas ecológicas que tejen entre sí a las muchísimas especies co-evolucionantes. Como ejemplo de complejidad valga la red autorreguladora de un genoma (o sea del juego completo de genes de un organismo) que ofrece una explicación de cómo una autoorganización ("anti-caos") puede llegar a gobernar el desarrollo de un organismo biológico.

La complejidad como tal es materia de estudios increíblemente reveladores y la biología se amolda a una nueva Ciencia de las Ciencias, cuyo capítulo más importante es el de la Ciencia de la Complejidad o de los Sistemas Complejos. La complejidad, con sus aportes característicos, enriquece todo lo que se sabe de casos simples. La física y la química estudian sobre todo lo comparativamente simple; necesitan, sin embargo, introducirse también en lo complejo, en el campo de los signos, que resalta sobre todo dentro del campo biológico.

Una solución para afrontar la complejidad: el pensamiento por niveles

Esta noción de niveles es fundamental para entender las emergentes “ciencias de la complejidad”, es decir, aquellas que investigan cómo surgen fenómenos complejos de componentes e interacciones simples[18]. Los nuevos proyectos de investigación en caos, auto organización, sistemas adaptativos, dinámicas no lineales, y vida artificial son parte de este creciente interés por los sistemas complejos que se ha extendido, junto con la publicación de diversos libros sobre el tema, desde la comunidad científica hasta la cultura popular. La investigación de sistemas complejos aborda algunos de los temas más profundos de la ciencia y la filosofía, tales como el orden contra el caos, el azar contra la determinación y el análisis contra la síntesis. Para muchos, el estudio de la complejidad no es solamente una nueva ciencia, sino una nueva manera de pensar sobre toda la ciencia, un cambio fundamental de los paradigmas que han dominado el pensamiento científico

durante los últimos trescientos años. A pesar de que los investigadores de la complejidad no se han extendido particularmente sobre la noción de niveles, ésta es una de las ideas centrales de las ciencias de la complejidad, fundamental para ayudar a los inexpertos a alcanzar un entendimiento de las mismas. Al enfatizar esta noción, se espera que la gente modifique su visión de los sistemas y utilice los niveles como un marco para analizar sistemas desde múltiples perspectivas. Se intenta que ello permita a las personas desarrollar mejores explicaciones causales de las interacciones y relaciones entre los elementos de los sistemas.

De hecho, la visión emergente de niveles es una poderosa herramienta para entender algunos de los temas más antiguos de la ciencia. Algunos de los mayores avances y controversias en el campo de la biología evolutiva dependen de una cuestión de niveles. Por ejemplo, ¿es correcto pensar en variación y selección al nivel del gen, del organismo o de las especies? De igual manera, hoy en día, muchas investigaciones sobre la naturaleza de la mente se centran en la idea de niveles. Según Minsky, la mente surge de las interacciones entre una compleja sociedad de agentes que se organizan en una variedad de estructuras. Hofstadter la compara con una colonia de hormigas; así como el comportamiento de una colonia surge de las interacciones entre hormigas individuales, la mente surge de interacciones entre "hormigas cognitivas".

No hay duda de que la gente tiene dificultad para entender el sentido emergente de los niveles. Sin embargo, existen razones para confiar en la posibilidad de ayudar a las personas a superar sus confusiones al respecto. Hay indicios de que la gente puede relacionarse con la noción de niveles emergentes, mucho mejor que con otros conceptos difíciles[19] [20]

Lo que algunos autores consideran complejo, otros lo consideran simple: la complejidad suele ser "observador-dependiente". En general se puede afirmar, según Leibniz, que mucho depende del punto de vista del observador. El sistema solar es complejo si se lo mira desde la tierra pero se simplifica si el observador imagina estar ubicado en el sol.

Teoría de la Complejidad

La moderna teoría de la complejidad[22] se basa en principios informacionales y computacionales que intentan abordar con cierto nivel de abstracción campos tales como el estudio de la naturaleza y consecuencias de las interacciones y no-linealidades de sistemas con muchas variables, con muchos objetos y que

presentan múltiples objetivos. Incluye tópicos de teoría general de sistemas, vida artificial, autómatas celulares, caos, valles de atracción en redes neuronales, criticalidad, computación evolucionaria, algoritmos genéticos, fractales, computación paralela, sistemas auto-organizantes, sistemas adaptivos compartimentalizados, sistemas dinámicos, inteligencia artificial, complejidad en biología.

Teoría del Caos

Como una primera aproximación podríamos definir al Caos como una conducta de un sistema complejo que aparece como impredecible y falsamente como aleatoria, cuando en realidad tiene un patrón subyacente. Es extraordinariamente sensible a pequeños cambios en las condiciones iniciales. El caos es una palabra especializada. Caos es un subconjunto de dinámicas dentro de la complejidad. En biología, la zona lindera entre el orden y el caos es frecuentemente propicia para el surgimiento de autoorganizaciones[23].

La Teoría del Caos intenta explicar el aparente desorden evidente de una forma ordenada. Muchos acontecimientos que parecen aleatorios se pueden representar por un método simple de cálculo que, cuando se itera, produce resultados complejos. La mayoría de procesos naturales son caóticos, al igual que muchos procesos del hombre.

Diferentes algoritmos cuando los iteramos producen diversos valores de complejidad y de cambios en un sistema caótico, estos tienen diversos niveles del impacto. Cada acontecimiento tiene constantemente diversos efectos en el establecimiento del siguiente paso. Un ejemplo famoso es el efecto mariposa, se dice que el golpeo de las alas de la mariposa puede afectar a la ruta de un huracán en la otra cara del planeta, por otra parte éste no puede tener efecto algunos, y es esta carencia evidente de calma en la predicción es lo que crea lo que llamamos efecto aleatorio. Es solamente posible si hacer un pronóstico en un sistema caótico si todas las entradas de información son conocidas y las reglas del sistema también son conocidas. Si solamente las reglas son conocidas, se puede reproducir el comportamiento, pero no la ocurrencia exacta que había sido producida con anterioridad. Por ejemplo, podemos crear un árbol en una computadora, pero a menos que no sepamos como era la semilla del árbol origen, será muy difícil reproducir ese árbol, mas aun así ese árbol es un nuevo árbol, no es el mismo.

La teoría del caos trata los modelos. Un sistema caótico trata siempre los diferentes pasos repetidos en modelos irregulares o, a veces, se producen modelos casi regulares.

Finalmente, y como conclusión de nuestra idea de Caos, podemos decir que en situaciones caóticas se presenta la circunstancia donde pequeñas causas provocan grandes efectos, realmente amplificando el estímulo que desencadena el cambio. Y es esta, en definitiva, la gran ventaja del Caos.

Auto-Organización (Self-Organization)

El término **auto-organización** (más precisamente "sistemas auto-organizativos") fue definido por Farley y Clark del laboratorio de Lincoln en 1954 como: "*Un sistema auto-organizativo es un sistema que cambia su estructura básica como función de su experiencia y el entorno*".

Esta definición se relaciona claramente con los términos: redes neuronales, algoritmos genéticos, control adaptativo.
Un sistema auto-organizativo posee tres características principales (o funciones):

- Affect
- Telos
- Effect

Un agente robótico observa su entorno (affect), utilizando estas observaciones decide que hacer luego (telos) y ejecuta acciones según esa decisión (effect).

La auto-organización se encuentra en múltiples campos de investigación: Biología (sociedades de insectos, ecosistemas), química (termodinámica), ciencias de la computación (redes neuronales, lógica difusa), geología (movimientos tectónicos), sociología (comunicación y migración), economía (sistemas socio-espaciales).

Algunos investigadores, como Nicolis y Prigogine, enmarcan a la auto-organización en sistemas caóticos, afirman que la auto-organización enfatiza la coordinación de procesos a gran escala en muchos niveles.

Otros, como Kauffman, creen que la auto-organización, una "propiedad inherente de cualquier sistema complejo", puede ser responsable de la evolución biológica

junto con la selección. Sus modelos de computadora sugieren que ciertos sistemas biológicos complejos tienden hacia la auto-organización.

La auto-organización posee tres características:

- Primero, un sistema auto-organizativo puede llevar a cabo tareas complejas con poco comportamiento individual y simple.

- En segundo lugar, un cambio en el ambiente puede influir en el mismo sistema de manera de que el mismo realice tareas diferentes sin cambios en las características de comportamiento.

- Por último, cualquier diferencia pequeña en el comportamiento individual puede influir en el comportamiento del sistema. De esta forma, la complejidad de la sociedad del sistema es compatible con individuos idénticos y simples, siempre y cuando la comunicación entre los miembros provea el mecanismo amplificador necesario.

Y es justamente la auto-organización que se presenta como protagonista y posible solución de los temas plantados por la ciencia hoy día. Algunas moléculas complicadas se auto-organizan. Un ejemplo claro es la molécula de la hemoglobina. Consiste en una proteína que se pliega sobre un átomo de hierro para así poder atraer a un átomo de oxígeno. Hoy sabemos como fabricar la hemoglobina, pero nada más. Afortunadamente, con solo fabricarla ella misma se pliega y atrapa al átomo de hierro, y a su vez este atrae a uno de oxígeno.

Por esto, pensamos que a partir de una molécula, esta se auto-organiza de tal manera que crea a un ser vivo. Pocos afirmarían que este comportamiento es predecible con solo observar la molécula inicial. Por ello decimos que es un **comportamiento emergente**. Este término se aplica a todo hecho que evoluciona de una manera imposible de predecir al principio, ya que ni siquiera sabemos si el hecho evolucionará (usándose "evolucionar" como sinónimo de "cambiar con el tiempo").

Comportamiento Emergente

El comportamiento emergente consiste en el fenómeno de producirse un comportamiento complejo a partir de una serie de reglas simples.

Un ejemplo ilustrativo del comportamiento emergente es el caso del experimento desarrollado por Cristopher Langton en la Universidad de Michigan. Langton simuló un entorno artificial de una colonia de hormigas en una rejilla compuesta por celdillas. Las reglas asignadas a las hormigas fueron: si una hormiga penetra en una celdilla blanca, continúa desplazándose hacia adelante. Si se introduce en una celdilla azul, ésta adopta el color amarillo, y la hormiga girará a la izquierda; y si penetra en una celdilla amarilla, ésta se tornará azul, y la hormiga girará a la derecha.

En un primer momento, las hormigas erraron sin rumbo definido, pero en poco tiempo las hormigas se encontraron unas a otras formando una fila. El aspecto interesante de este experimento es que no se había programado explícitamente que las hormigas adoptaran un comportamiento social de este tipo, sino que se produjo espontáneamente. Este tipo de comportamiento es habitual en experimentos de vida artificial.

El comportamiento emergente es un concepto clave de la Vida Artificial. La vida natural emerge de interacciones organizadas de un gran numero de moléculas no vivientes, sin ningún control global responsable del comportamiento de cada parte. Antes, cada parte es un comportamiento propio, y la vida es el comportamiento que emerge desde fuera de todas las interacciones locales entre comportamientos individuales. Esta es una determinación local, distribuida, hacia-arriba, del comportamiento que emplea la Vida Artificial en su enfoque metodológico primario para la generación de comportamientos similares a la vida[7].

Sistemas Descentralizados

Los sistemas descentralizados son aquellos que se organizan sin que exista un organizador.

Los sistemas descentralizados se basan en la premisa de que no hay ningún subsistema central que concentre la toma de decisiones y dicte el curso de acción a seguir. El plan de las acciones se decide en forma colectiva por varios subsistemas distribuidos en diferentes lugares físicos o en el mismo lugar.

Este tipo de sistemas brindan un enfoque bottom-up es decir que la resolución de los problemas emerge de la interacción local de las entidades.

1.5. Framework

Que es un Framework?

Los Frameworks de aplicaciones orientadas a objetos se observan comúnmente como una tecnología útil para lograr reusabilidad en sistemas de software.

La definición más comúnmente usada dice que un Framework es una arquitectura de soft reusable, consistente en un conjunto de clases abstractas y/o concretas que modelan la forma en que colaboran las instancias del dominio.

Otra definición común es que un Framework es un esqueleto de una aplicación que puede ser hecha a medida por un desarrollador de aplicaciones.

La primer definición describe la estructura de un Framework, mientras que la segunda describe su propósito.

Se los puede ver como un tipo de arquitectura específica del dominio.

Los Frameworks son importantes y los serán más en un futuro. Será la base subyacente de toda aplicación futura dentro del dominio.

Un Framework modela las características generales del dominio del problema.

Beneficios - Para qué sirven los Frameworks?

Los beneficios de un Framework radican en que provee un esqueleto general y reusable de clases y patrones de comportamiento para un dominio determinado, y confiando en este soporte nuevas aplicaciones que pueden ser desarrolladas de manera directa y flexible, con ahorros adicionales de esfuerzo de diseño y tiempo. Pero estos beneficios son sólo una cara de la moneda; estos beneficios deberían imponerse durante el diseño para asegurar esa calidad los productos de software se desarrollen.

- Reducir el costo de desarrollo y mantenimiento de una aplicación.
- Se reutiliza no sólo código sino también diseño.
- Contienen la experiencia de muchos diseñadores dentro del dominio.

Cómo desarrollar un Framework?

Un punto importante que se debe tener en cuenta para que un Framework pueda resultar útil y en particular para que el que intentamos desarrollar en esta tesis lo sea, es estudiar, investigar y llevar a la práctica la manera de encarar el desarrollo del mismo.

Para esto es necesario tener presentes algunas cuestiones:

- Concreto → General:
  Desarrollar aplicaciones concretas del dominio o estudiar ejemplos. Factorizar y Generalizar.

- Comprender el dominio:
  Encontrar problemas recurrentes del dominio.

- Identificar **hot-spots**:
  Puntos en el Framework donde aparecerán las variaciones. Particularidad de cada aplicación.

El proceso de construir una Framework es altamente complejo. Requiere de un pormenorizado esfuerzo para capturar las abstracciones necesarias del dominio y expresarlas en una estructura (Framework) poderosa y comprensible para desarrolladores de aplicación.

En particular todos los Frameworks requieren de un análisis del dominio y una ingeniería del dominio, es por esto que hay que pagar un alto costo antes de ver los beneficios.

La metodología preferida consiste de desarrollo reiterativo, comenzando con unos ejemplos y aplicando consecutivos refinamientos a la estructura hasta alcanzar un estado razonable de madurez.

La flexibilidad es un deseable e importante factor de calidad en la evolución de software actual. Esta propiedad define la capacidad de un sistema software determinado para resolver con poco trabajo los cambios que se den en la especificación del problema, produciendo un impacto bajo sobre los componentes implementados anteriormente. De esta manera, los sistemas son capaces de evolucionar y abordar variaciones diferentes de un problema determinado. La falta

de flexibilidad puede convertirse en un punto crítico si se quiere escalar el sistema. Los desarrolladores deberían pensar no solamente en el objetivo del sistema durante las fases de diseño sino deberían considerar también futuras variaciones del mismo.

Análisis de Frameworks existentes para ALife

En la presente sección analizaremos algunos Frameworks existentes para obtener una primera aproximación de los problemas que se encuentran en estos tipos de sistemas de vida artificial.
Se analizarán tres Frameworks existentes, los cuales son: Evo, FraMaS y Bubble.

Evo

Evo fue diseñado con el objetivo de proveerle a los investigadores de las ciencias biológicas una herramienta para facilitarle el estudio de la evolución del comportamiento.
Evo es un Framework de objetos reusables que permiten a los investigadores estudiar sistemas complejos de agentes independientes que interactúan unos con otros y con su ambiente.

Está construido sobre el sistema de simulación Swarm. Swarm es un paquete de software desarrollado por el *Instituto de Santa Fe para el estudio de Sistemas Complejos, New México, USA*, cuyo propósito es la simulación de sistemas complejos. Swarm provee funcionalidad como por ejemplo: administración de tareas, administración de memoria, widgets GUI, generadores de números random, una librería de colecciones, etc.

Una simulación Evo consta de una **población de agentes** "viviendo" en un ambiente. Un programa agente se construye de comportamientos individuales. Cuando dos agentes se reproducen el comportamiento del nuevo agente es heredado de sus ancestros. El investigador es responsable de especificar: acciones que puede ejecutar el agente, sentidos que posee, que información pueden observar en otros agentes de la población. El desarrollador implementa los individuos en la población subclasificando el objeto Agent. El objeto Agent provee comportamiento por defecto para reproducirse con otros agentes, moverse a través del ambiente y observar otros agentes.

Los agentes poseen un conjunto de **características heredables**. Cuando se construye una simulación Evo el desarrollador debe decidir que características poseerán los agentes y el rango de valores permitido de las mismas.

Los "**comportamientos**" se construyen en base a *instrucciones, sentidos y observaciones*. El desarrollador debe especificar las instrucciones los sentidos, etc. De los cuales consistirán los comportamientos.

Una **instrucción** es una acción simple que el agente ejecuta, por ejemplo: comer.

Los **sentidos** permiten al agente observar su medio ambiente.

Las **observaciones** permiten al agente observar a otros agentes. Un agente observable esta constituido por sus características que pueden ser observadas por otro agente. Por ejemplo si una clase de agente tiene un atributo llamado "age" y queremos que otros agente puedan observar ese atributo entonces debemos especificar esa característica como **observable**. La habilidad de los individuos de observar características de otros individuos provee la oportunidad de que ocurra el fenómeno de selección sexual donde un individuo "elige" reproducirse con otro porque poseen características particulares.

Los **mapas de recursos** son objetos que representan alguna distribución de recursos en el espacio. Los recursos pueden cambiar a través del tiempo por lo tanto Evo permite que los mapas de recursos puedan ser actualizados. Por ejemplo si un desarrollador quiere que el medio ambiente contenga comida, debe subclasificar la clase ResourceMap y crear una clase por ejemplo: FoodMap. Para hacer que el recurso cambie a través del tiempo (ejemplo: crecimiento de comida), se debe especificar un intervalo de actualización para la instancia de FoodMap. Evo maneja la administración de los mapas de recursos y llamará automáticamente al método de actualización de cada mapa de recurso según su intervalo de actualización. La Clase Environment maneja la administración de la población y todos los mapas de recursos. Subclasificando la clase Environment el desarrollador puede dar a la simulación una noción de espacio. Por ejemplo, la clase EnvironmentDiscrete2D provee un modelo de espacio que es una grilla bi-dimensional.

Cada agente tiene un genoma de 2 **cromosomas**: el cromosoma de característica y el cromosoma de comportamiento.

Cuando dos agentes se **reproducen**, sus genomas son combinados para formar los genomas de dos nuevos individuos, utilizando el operador reproductivo *crossover*. Durante la reproducción, la lista de características de un ancestro es combinada con la lista de características del segundo ancestro formando dos nuevos cromosomas de características para los dos nuevos individuos. Sucede de manera similar con la lista de comportamientos.

Durante la reproducción, hay una posibilidad estadística de que ocurra una **mutación**. Una mutación es un cambio inesperado o aleatorio, en biología se lo suele denominar "error de copia", este fenómeno a veces produce seres mejores adaptados con lo cual poseen mas chances de sobrevivir y por ende reproducirse.

FraMaS

FraMaS es el nombre del Framework propuesto para SMA y es el acrónimo de **Fra**mework para **S**istemas **M**ulti-**a**gente.

La construcción del Framework fue realizada utilizando un enfoque bottom-up. Se desarrolló un SMA en Java desde el que se identificó un conjunto de componentes abstractos –y sus relaciones, a fin de ser reutilizados. La idea es motivada por la complejidad que este tipo de sistemas presenta y por la gran cantidad de componentes –y comportamientos, concretos y abstractos que pueden ser reutilizados.

En particular se diseñó un agente que ayuda al usuario en la administración de sus reuniones y tareas. Primero se construyó un sistema de agenda simple al que se le adicionaron responsabilidades.

El diseño de cada agente está basado en una estructura de clases que permite adicionar responsabilidades dinámicamente. Es decir, partir de un diseño con tareas simples y luego "cubrirlas" con nuevos comportamientos. Esta forma de agregar funcionalidad a las tareas existentes es también llamada wrapper.

El diseño del Framework permite adicionar responsabilidades a los objetos, sin cambiar el mensaje de llamada. Se observa en la figura el esquema de wrapper o "decoración" de la tarea1. Más adelante se presenta una aplicación de este diseño.

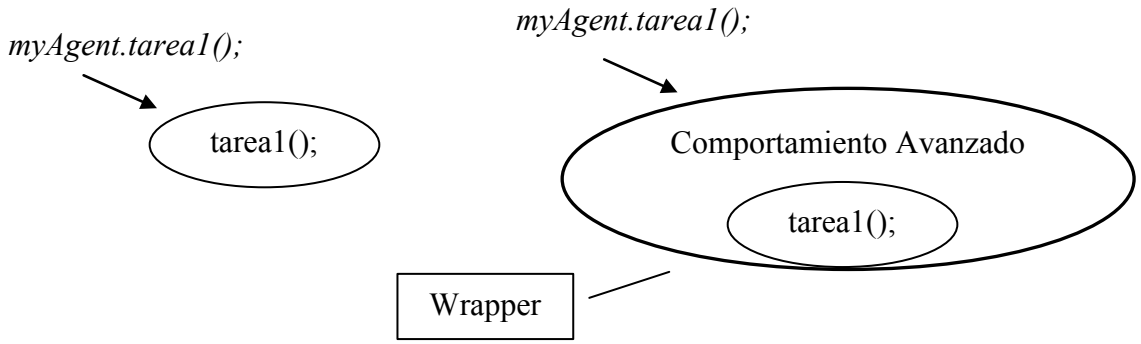

Figura 8 – Esquema de Wrapper

Cada tipo de agente es construido de la siguiente manera. Primero, se identifica, en términos generales, que será un agente en el sistema a construir. Segundo, se identifican las tareas básicas y tercero, el "comportamiento avanzado" que tendrá el agente.

Por ejemplo, en una aplicación de un asistente personal para agendas, cada agente conocerá las preferencias del usuario y sugerirá de acuerdo a esas preferencias (que pueden cambiar a lo largo del tiempo).

El conjunto de acciones básicas en una agenda para la administración de reuniones son: ingresar y borrar una reunión, mostrar reuniones almacenadas, etc. (cabe mencionar que esto no es considerado un agente aún: sino que es el comportamiento básico que tendrá).
Se continua con la especificación del comportamiento avanzado. El agente agenda requiere: la capacidad de aprender las preferencias del usuario (por ejemplo usando un razonador basado en casos), negociar las reuniones con otras agendas y actuar autónomamente, es decir, que el usuario no inicie explícitamente todas sus acciones. El Framework posee una clase principal llamada *Agent*. FraMaS posee la clase *BasicAgentAction* –hereda de *Agent* (*ver figura 10*), con el conjunto de acciones que el agente realiza y los métodos, ya implementados, para agregar y borrar acciones. Desde BasicAgentAction hereda la clase que implementa el comportamiento básico (clase Calendar en el ejemplo). Se extiende la funcionalidad de las tareas con el objetivo de adicionar comportamientos inherentes a agentes. El pattern Decorator [27] provee la estructura de clases necesaria para permitir que estos comportamientos sean adicionados y alterados dinámicamente.

A continuación, se detalla un ejemplo que describe su funcionamiento.

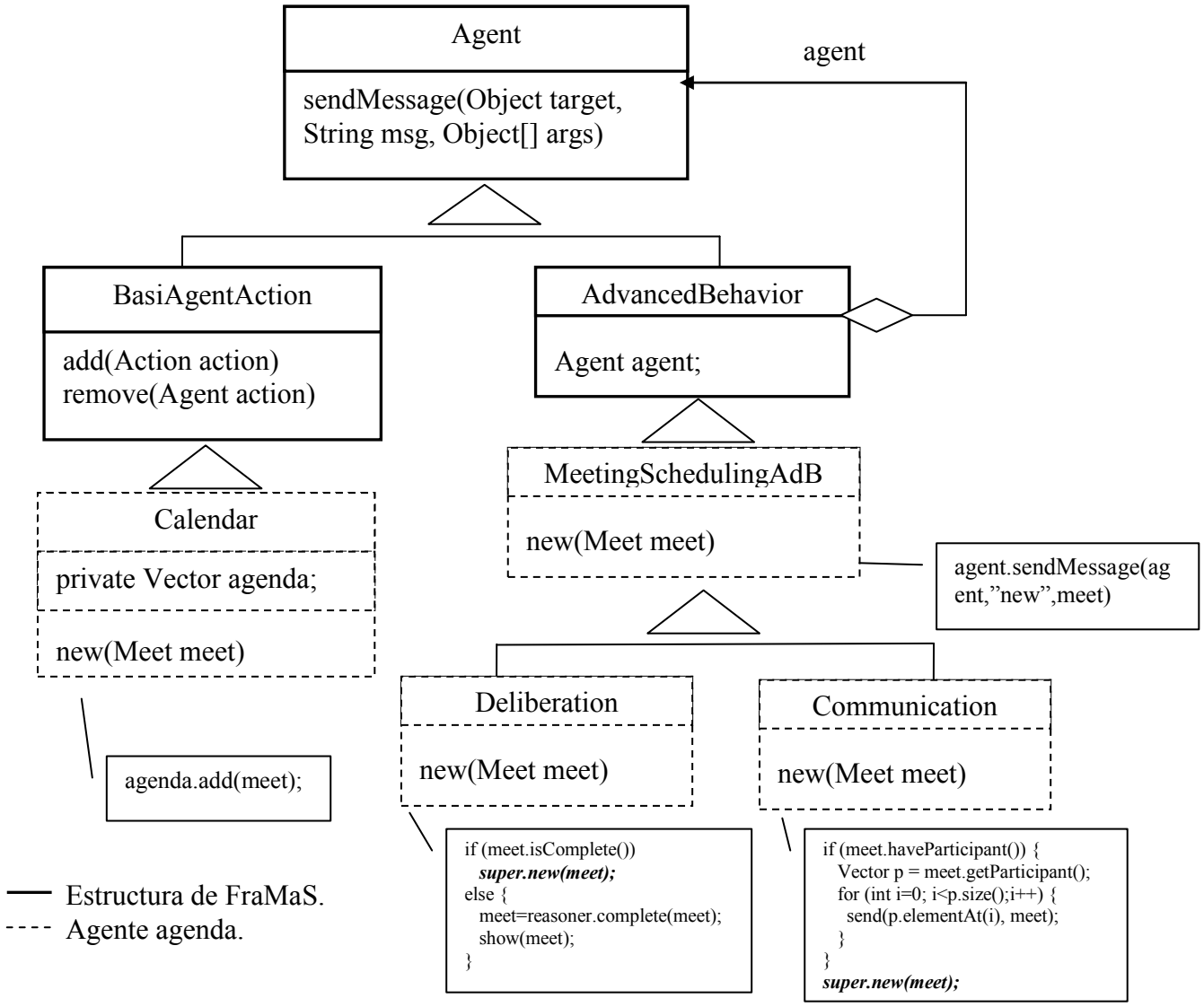

Figura 9 – Estructura de un agente FraMas

En la aplicación del agente agenda cada tarea tiene una fecha, hora, duración, lugar y tema; si es una reunión tendrá además una lista de participantes. Se requiere que el agente posea la capacidad de aprender las preferencias del usuario, en cuanto a: horarios usuales para cada tarea y reunión, participantes con los que trata determinados temas, lugares de reunión, etc. Como así también la capacidad de negociar las reuniones con los participantes (con la mínima intervención del usuario). Esto es expresado diciendo que los comportamientos de la agenda son "decorados" por otros comportamientos (avanzados). Análogamente se puede seguir adicionando responsabilidades a las tareas básicas. Lo más importante es que la agregación de wrappers es trasparente, es decir, sin modificar la interface existente. Además, la estructura de control ya está implementada en el Framework, por lo que sólo hay que diseñar el comportamiento particular que se desea.

Bubble

Bubble es un Framework Multi-agente implementado en Java, originalmente concebido y diseñado para simular el movimiento de burbujas de gas en un ambiente de fluidos. Los elementos básicos del sistema son agentes reactivos descriptos por un estado interno y un conjunto de tareas ejecutables.

La interacción entre estos agentes reactivos se realiza a través de eventos que los agentes producen y reciben. Los agentes están equipados con sensores (como filtros) que son registrados para "escuchar" determinados tipos de eventos.

El comportamiento de un agente reactivo está definido a través de tareas utilizando un estilo condición-acción, es decir que una tarea es un módulo compuesto por una serie de acciones a ser ejecutadas por el agente cuando determinadas condiciones se cumplen.

La siguiente figura muestra un diagrama del modelo conceptual soportado por el Framework, ilustrando un flujo de evento típico entre agentes dentro de un container y el rol que juegan los sensores en el proceso.

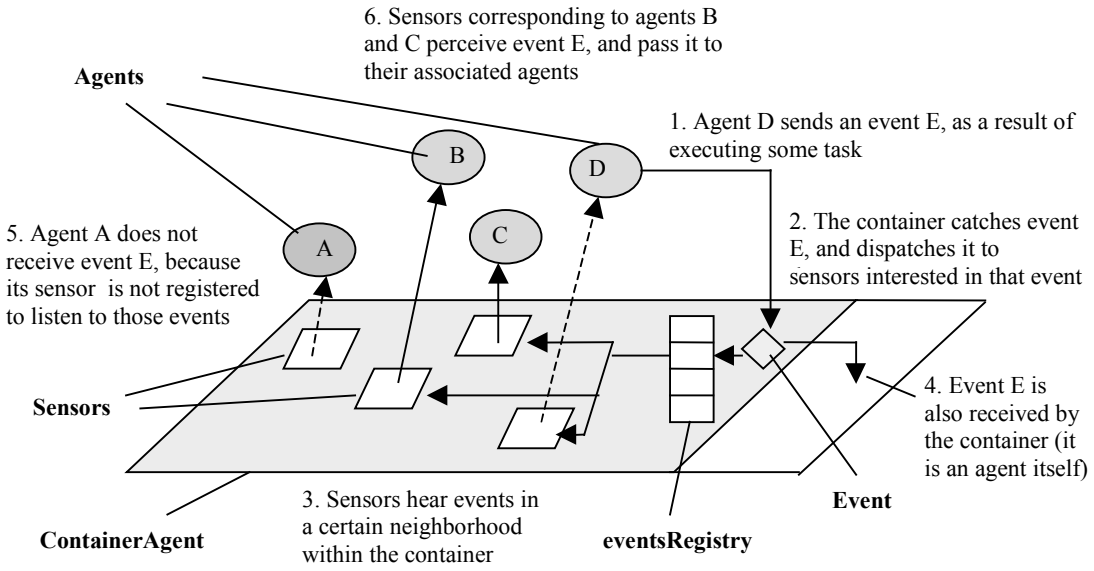

Figura 10– Arquitectura Bubble

La **comunicación** entre diferentes componentes en Bubble se lleva a cabo a través de eventos. Cada agente puede enlazarse a un agente-contenedor, y este contenedor es el encargado de recolectar y despachar los eventos entrantes a los

sensores **registrados** dentro de él. Un evento representa una notificación de cualquier cambio ocurrido en el sistema.

Los **sensores** son los responsables de la recepción y transmisión condicional (filtrado) de eventos. Los agentes-contenedor derivan los eventos recibidos de los agentes a los sensores de los mismos.

Todos los agentes del Framework pueden ejecutar un conjunto de **tareas**. Una tarea está compuesta por uno o más procedimientos con un conjunto de parámetros de entrada y salida. Las tareas son activadas por **condiciones predefinidas**, las cuales pueden estar relacionadas con el estado interno del agente o a eventos entrantes. De este modo el comportamiento de agente está concebido como un conjunto de **tareas competitivas**, donde sólo una tarea es activada por vez. Cuando una tarea se ejecuta, puede generar eventos salientes (hacia el contenedor) y/ó afectar el estado del agente.
Diferentes tareas pueden ser asignadas dinámicamente a un agente. Estas tareas compiten para ejecutar según sus **prioridades** y los **requerimientos de activación**.

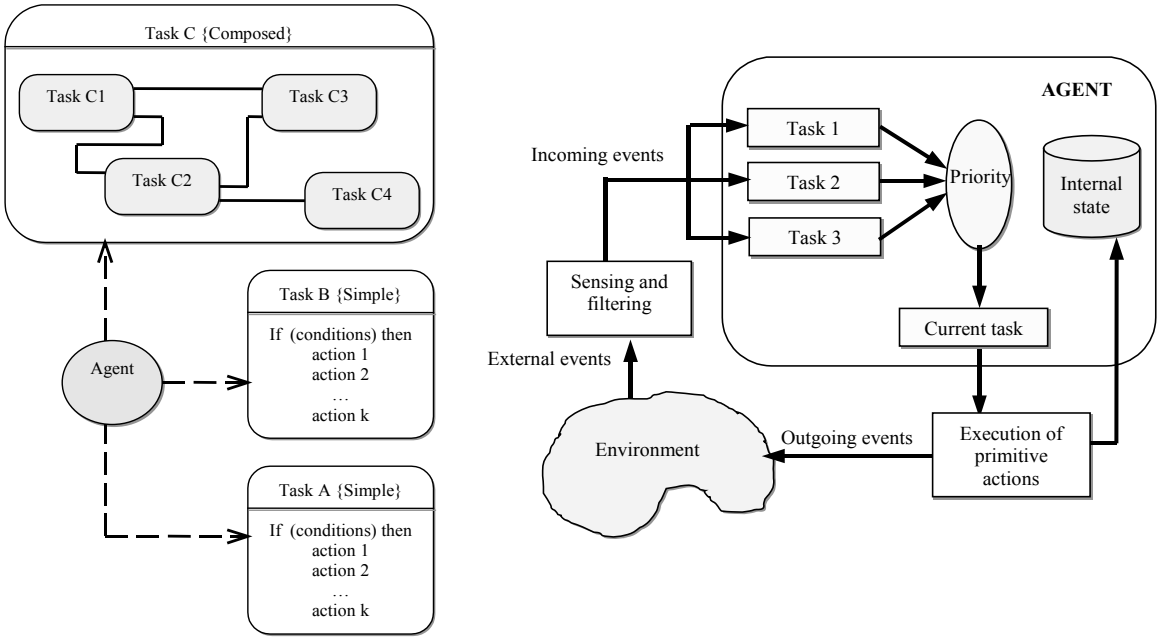

Figura 11 – Tareas Competitivas en Bubble

Resumen del Análisis de Frameworks existentes

Framework	Características destacables
Evo	• Reproducción entre agentes: Características y Comportamientos heredables. • Arquitectura del Agente: Sentidos, Instrucciones y Observaciones. • Genética: Representación en forma de cromosomas de las características y comportamientos. Crossover y Mutación. • Mapa de Recursos: Modelan la distribución de recursos en el ambiente.
FraMaS	• Arquitectura del Agente: Agregado de comportamientos a través de Wrapper's. • El Framework provee comportamientos básicos a extender por los desarrolladores.
Bubble	• Arquitectura del Agente: Reactivos. La interacción se realiza a través de eventos que inciden en los sensores de los agentes. • Modelo de Registro de sensores a eventos. • Agente Container para administrar el flujo de eventos, registro de sensores-eventos. • Modelo de Tareas Competitivas: Las tareas de un agente compiten para tomar el control de la ejecución. Se activan según las condiciones internas más los estímulos externos.

Tabla 3 – Características relevantes de los Frameworks analizados

Capítulo 2: Caracterización del Framework

En primer término se enumeran las observaciones del Marco Teórico, donde se extraen las ideas principales de cada una de las secciones identificando los puntos que utilizamos para el desarrollo del Framework.

*Se plantean luego las **características del dominio** que servirán como base para la posterior **caracterización del Framework**.*

*Se enumeran finalmente las **herramientas** provistas por el Framework.*

2.1. Extracciones del Marco Teórico

Se describen a continuación los principales conceptos observados en el Marco Teórico y que constituyen la base teórica para desarrollar el Framework.

Vida Natural

De este sección extraemos varias ideas útiles para el desarrollo del Framework. Por un lado los conceptos básicos de vida natural, genes, ADN, reproducción, competición y cooperación y por otro lado la mecánica de resolver los problemas de forma **sintética** en vez de analítica. Los componentes que modelizan características de seres vives se apoyan sobre estos conceptos teóricos.

Vida Artificial

De los ejemplos de sistemas de Vida Artificial podemos extraer que la técnica utilizada mayormente es la de definir **reglas básicas**, sin pensar en el comportamiento global y luego experimentar con la interacción de numerosos componentes que implementan esas reglas básicas. Lo interesante de estos sistemas es ver la interacción y el patrón global surgido. Es aquí donde surge nuestra primera decisión importante y es la de inclinarnos por agentes poco inteligentes en forma individual pero potentes en forma grupal.

Sistemas Multi-agentes

Diferenciamos dos grandes grupos de pensamiento en cuanto a tipos de agentes (reactivos y cognitivos) para luego decidir y enfocar nuestro trabajo a uno de ellos: los **agentes reactivos**.

Conociendo las arquitecturas de sistemas multi-agentes más difundidas. Es en este punto en que elegimos una de ellas, que a nuestro criterio es la que mejor se adapta a los agentes reactivos: la **arquitectura de tareas competitivas**.

Sistemas Complejos

"El pensamiento por niveles", además de su aporte intelectual, constituye una herramienta conceptual que será base para algunos de los procedimientos que pondremos en práctica al desarrollar el Framework. En el Capítulo 6, sección 6.7.2.1, se utiliza esta técnica para modelar Objetos Intermedios.

Framework

De los Frameworks analizados extraemos algunos elementos, como por ejemplo:

Evo: Nos interesa el modelo de **reproducción** de agentes y **mapas de recursos**.

FraMas: Su esquema de wrappers que utiliza permite definir puntos extensibles en el Framework. Esta idea nos ayudará a marcar nuestros propios **hot-spots**.

Bubble: Observamos una de las implementaciones de **tareas competitivas**.

2.2. Características del Dominio

El dominio presenta muchas características entre las cuales podemos encontrar:

- ✓ Gran cantidad de agentes interactuando entre ellos y el medio ambiente.
- ✓ Los agentes poseen un conocimiento acotado de su entorno local, o casi ningún conocimiento.

- ✓ El medio ambiente es un factor clave en este tipo de sistemas, puesto que fija las restricciones en las interacciones, provee recursos a los agentes, etc.

- ✓ Los agentes son autónomos y poseen objetivos que intentan cumplir. Es decir el comportamiento de los agentes está guiado por el o los objetivo/s que poseen.

- ✓ Fruto de cumplir estos objetivos, los agentes, generalmente entran en colaboración con otros agentes para satisfacer sus necesidades y cumplir mejor sus objetivos.

- ✓ Como consecuencia de las interacciones entre agentes y ambiente surge un comportamiento emergente no programado.

- ✓ Los agentes poseen las características de las entidades 'vivas': reproducción, percepción del ambiente o fenómenos que en él ocurren, metabolismo, ciclo vital, evolución, comportamiento social o grupal, acción, reacción a los estímulos externos, etc.

- ✓ En estos sistemas no hay un control centralizado sino más bien descentralizado, es decir todo comportamiento general es por ende emergente y no previsto. Ninguna entidad dicta el curso de las acciones, éstas se desarrollan en una forma no determinística.

- ✓ Como toda entidad 'viva' los agentes evolucionan de manera de que las próximas generaciones se adaptan de 'mejor' manera a las condiciones cambiantes impuestas por el ambiente. De manera similar como ocurre en la realidad que las especies de animales evolucionan, sobreviviendo las más aptas a las condiciones externas (Adaptación).

2.3. Caracterización del Framework

Enunciadas las características del dominio que resultan de interés a modelar, encontramos que un Framework para mundos virtuales debe contar con los siguientes componentes:

Agentes

Son objetos autónomos, activos y que persiguen ciertos objetivos. **Presentan las características de los seres vivos**.

Estas criaturas poseen un comportamiento orientado a cumplir los objetivos y a perpetuar el grupo de criaturas o colonia de la que forman parte.

Características

- Perciben el medio que lo rodea (entorno local) pero en un alcance acotado (limitado).
- Reaccionan a estímulos externos actuando en consecuencia y, en ocasiones, modificando el ambiente.
- Nacen, crecen, se reproducen y mueren.
- Realizan tareas que permiten satisfacer sus objetivos.
- Buscan recursos que necesitan para poder realizar sus tareas.
- Interactúan con el ambiente, otras criaturas y/u objetos de la simulación.
- Muchas de las criaturas poseen movimiento. Es decir se mueven para conseguir lo que necesitan, trasladándose por el Ambiente.

Ambiente

El ambiente es el medio donde conviven todos los elementos u objetos de la simulación. Es un gran contenedor de objetos.

Características

- En él ocurren eventos o situaciones que lo modifican (ej: incendios, lluvias, terremotos, etc..).
- Posee determinadas condiciones dependiendo de muchos factores (ej: zonas frías, cálidas, desérticas).
- Posee una estructura o topología que determina su forma geográfica.

- Puede ser modificado por los elementos de la simulación.

- Provee los recursos necesarios a las criaturas para que estas puedan sobrevivir.

Relaciones, interacciones, comunicación

Son las formas en que las criaturas, objetos inanimados y ambiente se relacionarán. Los elementos de la simulación no son objetos aislados, colaboran o se relacionan con otros objetos. El Framework se enfocará en las interacciones que las criaturas mantienen con otras criaturas, objetos inanimados y con el ambiente. Estas interacciones serán **locales** y **descentralizadas**, es decir que no habrá una entidad central que dicte el curso global de la simulación.

Objetos no "vivos"

Son objetos o elementos que se encuentran en el ambiente y se diferencian de las criaturas en que no están "vivos". Sirven como recursos para las criaturas.

Eventos y paso del tiempo

Son situaciones que pueden ocurrir en el ambiente, afectándolo no sólo a él sino también a los objetos que se encuentran en el ambiente (incluidas las criaturas).

Ciertos eventos funcionan como estímulos para las criaturas o para un grupo de ellas, las cuales reaccionan ante ese estímulo. El paso del tiempo también es importante ya que gracias a él las criaturas "crecen" cambiando su forma de comportamiento a lo largo del tiempo. El ambiente también sufre cambios debido al paso del tiempo.

Grupos, Colonias

Conjunto de criaturas que a través de sus interacciones o características comunes forman una entidad u organización de más alto nivel no programada.

Un grupo o colonia se puede ver como varias criaturas que entran en colaboración y/o cooperación para alcanzar objetivos comunes o grupales. De esta manera si se cumplen los objetivos del grupo, cada individuo perteneciente al mismo se verá satisfecho.

2.4. Herramientas provistas por el Framework

El Framework proveerá dos categorías de herramientas:

1. **Herramientas para recolectar y facilitar el análisis de la información generada por la simulación.**

Estas herramientas se dividen en dos grupos:

- Recolección de información: Se proveen mecanismos para capturar la información que va surgiendo de la ejecución de la simulación. El usuario decidirá que información será capturada y en que intervalos de tiempo.

- Análisis y deducción de información: el Framework esta construido sobre la base del análisis en la interacción entre agentes, clasificación de relaciones surgidas entre agentes y detección de propiedades comunes. Estos elementos son un valor agregado al Framework para la ayuda en la toma de decisiones.

2. **Herramientas de configuración de la simulación.**

- El objetivo de estos componentes es diseñar la simulación y sus elementos antes de iniciar la ejecución.

- Se proveen interfaces con el usuario para facilitar la tarea de creación y configuración de componentes.

- El estado creado por el usuario se utilizará como punto de partida para la ejecución de la simulación.

En el capítulo 6, Figura 22 se observan estas herramientas en forma general.

Capítulo 3: Agente

*En primer lugar se analizan y enumeran las **características de los sistemas "vivos"** que se incluirán en el Framework y que servirán de base para el diseño del componente agente.*

*En base a estas características, en la sección **arquitectura del agente,** se modelan las mismas y se define la estructura interna del agente en base al **modelo de tareas competitivas** adaptándolo a nuestras necesidades.*

*Una vez detallado el modelo general se explican los **componentes del agente,** detallando para cada uno su definición y comportamiento. La sección de **interacción entre componentes** tiene como objetivo esclarecer la colaboración entre los mismos dilucidando el funcionamiento global del agente. En esta sección se observa el flujo de distintos elementos.*

*En la última sección, **diseño del componente agente**, se presenta el diseño de clases y su descripción. Este diseño se presenta en forma de capas para su mejor comprensión.*

*Finalmente se describen los **puntos extensibles del Framework** identificados en este componente y que el usuario del Framework puede aprovechar para su adaptación y extensión.*

3.1. Características de sistemas "vivos"

En el Framework que se desarrolla se modelan entidades que presentan las propiedades de los sistemas "vivos", es por esto que deberán analizarse las características de dichos sistemas. La presente sección se divide en dos partes: características generales y características específicas.

Características generales

Esta categoría de características están presentes en toda entidad "viva". Entre las principales características de estos sistemas podemos encontrar:

Tendencia a preservar la "vida":

Las entidades "vivas" persiguen un objetivo muy claro que es mantenerse con "vida". Para esto necesitan satisfacer todas las necesidades relacionadas con esta meta.

Reproducción:

El hecho de buscar reproducirse obedece a un objetivo de más alto nivel que involucra la supervivencia del tipo de entidad de la que forma parte (comúnmente conocido con el nombre de **especie**).

Percepción del ambiente:

Estas entidades deben percibir el ambiente que les rodea para poder actuar en su entorno, para esto tienen en cuenta no sólo su estado interno sino también el estado del entorno local donde actúan.

Acciones y Tareas:

Las entidades "vivas" actúan de manera tal que modifican el ambiente que las rodea y así también el comportamiento de otras entidades ya sea explícitamente (interactuando) o implícitamente ante cambios en el ambiente.

Dichas entidades cuentan también con la capacidad de realizar distintas tareas que se realizan en pos de cumplir los objetivos que la entidad "viva" posee.

Características específicas

Este conjunto de características están relacionadas con lo específico del dominio que son las entidades biológicas, es por este motivo que se incluye este apartado analizando los items pertinentes a una entidad biológica.

Entre estas características encontramos:

- ✓ Obtener alimento para subsistir y tener energía para desarrollar las demás tareas.

- ✓ Se reproduce para perpetuar su **especie**.

- ✓ Duerme, como proceso compensador de la pérdida de energía debido a la ejecución de acciones.

- ✓ En ciertos casos construye un "hogar" o nido para resguardarse de sus predadores, poner huevos, obtener calor, etc.
  Es decir que busca mejores condiciones de vida.

- ✓ En algunos casos buscan **similares** a él, entidades organizadas en grupos.

- ✓ Aprende en función de su interacción con el medio que lo rodea, acumula experiencia.

- ✓ Se deteriora y luego de un tiempo muere.

Genética

En las entidades vivas toda la información requerida para **crear** un ser con las mismas características se encuentra en los genes.

El conjunto de genes transmite de generación en generación características que sirvieron a los antepasados en la supervivencia dentro del entorno donde se encontraban.

- ➢ Genotipo
  El Genotipo almacena la información genética del individuo y especifica como construir una entidad con dicho genotipo, es decir contiene la materia prima necesaria para crear un Fenotipo o agente en la simulación.

- Fenotipo
  El análogo a este concepto es la entidad viva ya creada. No sólo con el genotipo se consigue la construcción de una entidad ya que el medio ambiente (causas externas presentes en el ambiente) también influye en la construcción de dicha entidad.

Especies

Dos entidades vivas pertenecen a la misma **especie** si comparten ciertas características comunes en sus **genes**. Esta habilidad es innata en la naturaleza y permite que todas las entidades distingan a otras de su misma especie y más específicamente reconocer la especie de otras entidades. Esta condición es necesaria para el comportamiento grupal.

Una especie es una entidad de nivel superior que agrupa características comunes de las entidades que pertenecen a ella. La misma no determina el comportamiento de las entidades pertenecientes sino que establece las **tareas posibles** que podrán realizar dichas entidades, la estructura del genotipo, etc.

Reproducción

Para lograr el objetivo de perpetuar la especie, las entidades deben reproducirse es decir generar nuevos individuos a partir de los ya existentes.

La reproducción se logra combinando el material genético de dos entidades de la misma especie. Una vez que se combina el código genético de dos entidades se crea un nuevo **genotipo.**

La reproducción es un **factor evolutivo** importante ya que combina el material genético de los individuos mejor adaptados al entorno.

Movimiento

En el mundo biológico otra característica importante a tener en cuenta es la capacidad de movimiento de las entidades. Esta capacidad corresponde a la habilidad de desplazarse por el ambiente modificando su posición y su contexto local.

3.2. Arquitectura del agente

En esta sección se enumeran los factores que se tendrán en cuenta y como se modelarán el conjunto de características anteriormente mencionadas.

La primera decisión importante es que **modelo de agentes** utilizar, en el marco teórico se estudio las dos categorías en que se dividían los agentes: **cognitivos** y **reactivos**. Como el presente Framework se desarrolla en el dominio de mundos virtuales biológicos se toma la opción de **agentes reactivos**.

Esta aproximación no requiere que el agente sea potente en forma individual, es decir no requiere que tenga un esquema de representación del conocimiento elaborado como por ejemplo: sistemas basados en reglas como la IA clásica. La potencia de dichos agentes reside en forma grupal, por lo que a nivel individual se modelará un mecanismo basado en reacción a estímulos externos.

Un aspecto importante a tener en cuenta son las **características específicas** a entidades biológicas.

En cuanto a genética se refiere, se modelará un mecanismo para la representación de un agente en forma de "genes" y luego un proceso de interpretación de ese **genotipo** que conjuntamente con el medio ambiente formen un agente o **fenotipo**.

Otro aspecto es la capacidad de los agentes de reconocer otros agentes **similares** es decir reconocimiento de **tipos de agentes**.

Como ultimo punto importante a los aspectos biológicos se modelará un ambiente Bi-Dimensional (2D). Los agentes se podrán mover en cualquier dirección dentro de los 360 grados posibles. Es decir que se moverán en ángulos relativos a la posición actual.

También el movimiento forma parte de su comportamiento y podrá estar definido, en algunos casos, a nivel de especie. Las agentes tendrán distintas **estrategias de movimientos**, (ejemplo: movimientos en forma de grupo o individuales).

Estructura interna del agente: Tareas competitivas

La arquitectura de representación interna del agente que se elige es la de **tareas competitivas** (ver *Capítulo 1, Sección 1.3*) ya que cumple con la mayor parte de las necesidades de este tipo de agentes, a saber:

- Un agente puede ejecutar distintas tareas.
- El agente debe evaluar que tarea le conviene ante determinada situación externa e interna.
- Se ejecuta la tarea seleccionada actuando de acuerdo a ella en el ambiente.

Se extiende este modelo con la idea de **objetivos** del agente, es decir que la decisión de la tarea a ejecutar no sólo se hace sobre la base de las percepciones del entorno que lo rodea sino también con el cumplimiento de sus objetivos.

Esquema General del Agente:

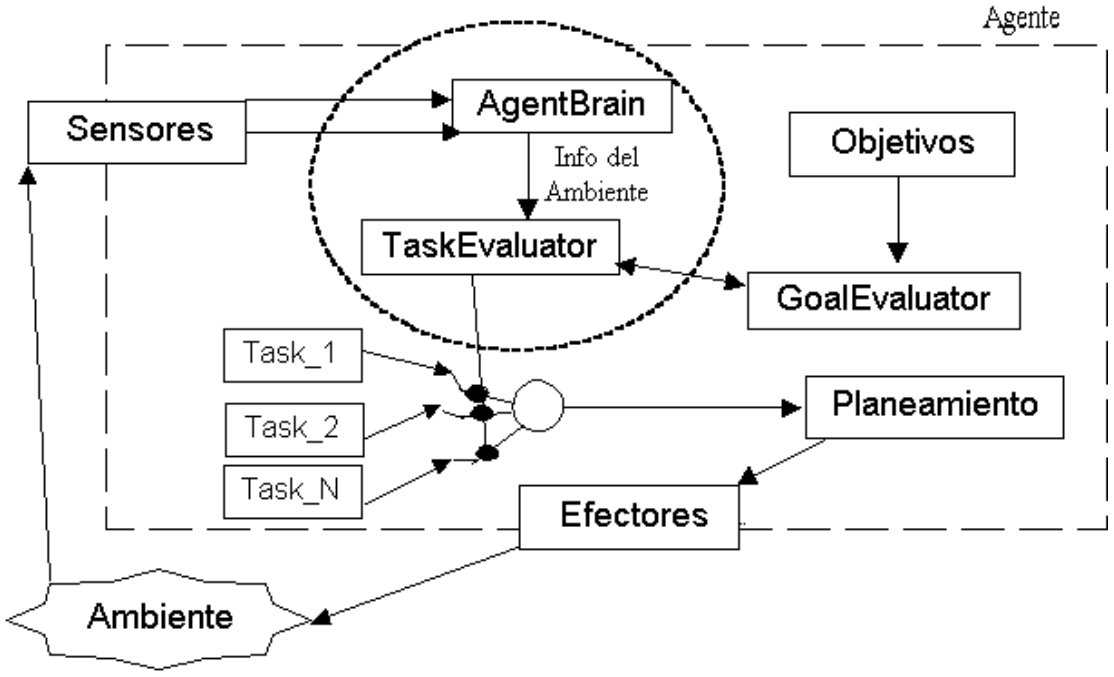

Figura 12 – Modelo general del agente

En forma general el funcionamiento del agente es como se describe a continuación:

1. El agente puede actuar por decisión propia o por la recepción de un estimulo o evento externo.

2. Antes de actuar el agente percibe su entorno local.

3. Evalúa los objetivos en base a esta información y su estado interno.

4. En base a los objetivos que se deben cumplir se elige la tarea que más satisface las necesidades actuales.

5. Se procede a ejecutar la tarea seleccionada.

6. Las acciones que ejecuta la tarea seleccionada son enviadas a la unidad que planifica y organiza los efectores que actúan en el ambiente.

Nota:
Este modelo también se utiliza en otros Frameworks como por ejemplo: **Bubble** analizado en el *Capítulo 2*.

3.3. Componentes del agente

A continuación se detallan los componentes que forman parte del agente, descriptos, la mayor parte de ellos, en la *figura 12 - modelo general del agente.*

Sensores

Captan la información del exterior (evalúa el ambiente en cada momento), la codifica y la envía al módulo que unifica las señales **(AgentBrain)** de todos los sensores.

Puede verse al sensor como un **Adapter**, ya que transforma la información externa en una clase de información o señal interna entendible por el agente.

Se comporta como una estrategia de codificación, la cuál toma como entrada una **señal externa** y la convierte en una **señal interna unificada (USignals)** entendible por el agente.

Señales Externas al Agente

Son las señales que existen fuera del agente, es decir que circulan en el ambiente. Cada tipo de sensor puede interpretar y codificar un tipo de señal externa. Por ejemplo: El sensor visual sólo podrá captar e interpretar la señal lumínica, el sensor auditivo es compatible con la señal sonora.

Es decir que la existencia de un tipo de sensor se debe a que en el exterior hay una señal que requiere ser interpretada como portadora de información valiosa para el agente para comprender su entorno.

Señales internas - USignals

Como en los seres vivos las señales externas son asimiladas por la entidad que las percibe y codificadas en un formato interno común.

El objetivo de este tipo de señales USignals es la de ser ese formato común que fluye en el interior del agente. Estas señales son producidas por los sensores. Modelan la señal nerviosa que se genera internamente en los seres vivos luego de percibir algún aspecto del exterior.

Los sensores se organizan en una Jerarquía:

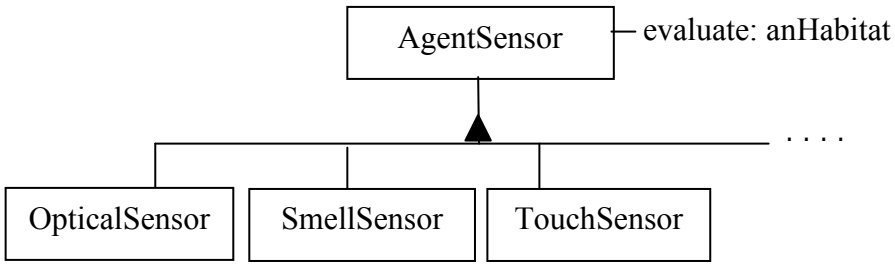

Figura 13 – Jerarquía de sensores

Un aspecto importante está referido a definir una cierta prioridad entre los sensores, de acuerdo a diversos factores que lo rodean y que hacen que la información captada por alguno de ellos sea más importante que otra. Por ejemplo, en la noche, la información recibida de los sensores visuales podría ser escasa o menos importante que la información obtenida por los sensores auditivos y/o olfativos. La importancia de la señal de los sensores puede ser modelada con dos enfoques diferentes:

Centralizado

A través de prioridades, con la existencia de un módulo que determina, ante determinadas situaciones referidas al contexto del agente, que importancia asignarle a cada sensor.

Descentralizado

Donde cada sensor percibe y emite la señal de acuerdo a las condiciones externas. Por ejemplo: la señal del sensor óptico será más escasa en la noche que en el día, por lo tanto al **AgentBrain** le llegará menos información óptica, que luego se verá reflejada en la poca importancia que le dará el **Evaluador de Tareas (TaskEvaluator)** en el momento de decidir que tarea ejecutar.

Como conclusión de estos dos planteos, decidimos que el enfoque **descentralizado** es el más apropiado, pues modela de manera más fiel la realidad, y además respetamos la idea central del Framework de resolver los problemas en forma descentralizada y control distribuido.

En la siguiente figura se observa como el **AgentBrain** recolecta la información de los diferentes sensores, este esquema también muestra el **flujo de la información** que circula internamente en el agente.

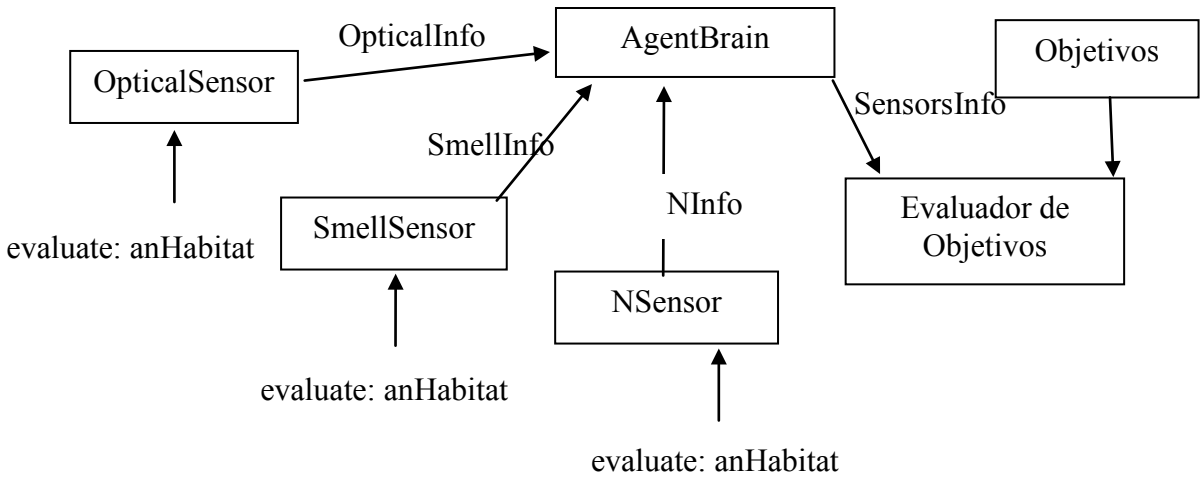

Figura 14 – Flujo de información de los sensores

AgentBrain

En este componente se almacena la información proveniente del exterior y codificada por los sensores. Es decir se mantiene una imagen o modelo de un instante de tiempo del entorno local del agente.

El **AgentBrain** es el encargado de proveer el servicio de acceso de la información recolectada por los sensores. El **GoalEvaluator** y **TaskEvaluator** hacen uso de este componente para cumplir con sus objetivos.

SensorHandlers

Representan la "zona del cerebro" donde incide la señal unificada generada por el sensor correspondiente. Son un parte interna del **AgentBrain**.

Para recibir las USignals (señales generadas por los sensores) el AgentBrain se encuentra provisto de SensorHandlers los cuales son responsables de interpretar las USignals generadas por el sensor correspondiente, es decir que cada tipo de Sensor posee un SensorHandler asociado que "sabe" como interpretar la USignal generada por dicho Sensor. Además el SensorHandler es responsable de almacenar la USignal de forma adecuada dentro del AgentBrain.

Como consecuencia a estar asociados a un tipo de sensor, se genera una jerarquía paralela a la de Sensores, *(ver figura 15)*.

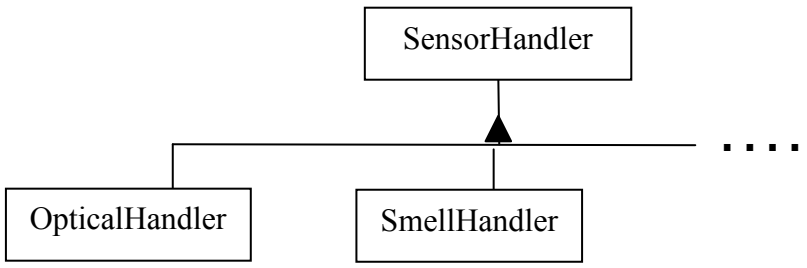

Figura 15 – Jerarquía de SensorHandlers

Objetivos

Los objetivos son premisas que guían el comportamiento de cada agente.
Un objetivo se puede ver como una función que representa el grado de satisfacción que le produce al agente un estado dado (estado externo + estado interno).

De esta forma el agente actuará de manera de maximizar sus objetivos.

A grandes rasgos y a los efectos de que el Framework sea extensible, los objetivos se dividen en dos grandes grupos:

- Específicos de entidades vivas:
  Son objetivos innatos a todo ser vivo como por ejemplo: "mantenerse vivo".

- Específicos de la aplicación:
  En esta categoría entran los objetivos provistos por los usuarios del Framework que son específicos de la aplicación.

Análogamente como ocurre en el mundo real, los objetivos poseen prioridades. Dichas prioridades establecen un orden de cumplimiento de objetivos según las condiciones externas (del medio ambiente) y las internas (estado interno del agente).

Un ejemplo del mundo real: ante una amenaza el objetivo "seguir con vida" es más importante que "encontrar comida".

GoalEvaluator

Este componente es el encargado de supervisar el cumplimiento de los objetivos.

El GoalEvaluator va marcando en cada paso de la simulación los objetivos que hay que cumplir (los que están por debajo de un umbral serán los que hay que cumplir en el próximo paso, y no así los que están por encima), esto guiará el comportamiento del agente a través del tiempo.

Sobre la base de un umbral, las prioridades y las condiciones externas e internas se evalúa que objetivos son los que necesitan satisfacerse en primer término.

Este componente toma información del medio ambiente, ya asimilada (AgentBrain) y del estado interno del agente para determinar que objetivos cumplir.

Tareas

Las tareas son unidades componentes del agente que determinan su comportamiento. Es por esto que son un factor a tener en cuenta en el momento del desarrollo de un agente.

Una tarea es una habilidad que posee el agente para llevar a cabo cierto tipo de acciones. Es un conjunto de acciones cuya finalidad es resolver un problema determinado.

Una tarea se ejecuta como respuesta a las necesidades del agente, ya sea por el cumplimiento de objetivos o en respuesta inmediata a estímulos externos.

Las tareas toman como entrada la información del ambiente y el estado interno del agente y las acciones que ellas contienen están dirigidas, entre otras cosas:

- Modificar el ambiente, generando eventos.
- Modificar el estado interno del agente.
- Modificar el estado de otro agente o agentes.

Las tareas entran en competencia y la que mejor se ajusta a las condiciones será la que se ejecutará en un instante, es decir que en un instante de tiempo hay sólo una tarea activa.

El conjunto de tareas que "sabe" realizar el agente define el **rol del agente,** categorización o especialización. Es decir que se tiende a tener agentes especializados en determinadas tareas y no de propósito general.

Para aclarar este último punto tomamos un ejemplo del mundo real,
Colonia de hormigas. En una colonia de hormigas hay distintos tipo de hormigas o roles:

- ✓ *Recolectoras:* Transportan distintos elementos (hojas, insectos) hacia el nido.

- ✓ *Soldados:* Aseguran la seguridad de la colonia.

- ✓ *Cultivadoras:* Transforman los elementos recolectados por las recolectoras en hongos que luego serán el alimento para la colonia.

TaskEvaluator

El Evaluador de Tareas es el encargado de decidir que tarea debe realizar el agente en cada paso de la simulación. Para lograr esto el evaluador debe tener en cuenta los objetivos del agente y la información recibida desde los sensores y así seleccionar la tarea que mejor se ajuste a ese estado.

Una vez que el evaluador elige que tarea ejecutar, el agente termina de realizar la tarea actual y se pone a ejecutar la tarea seleccionada.

La tarea seleccionada deberá ser la que mejor satisfaga los objetivos elegidos y también deberá ser realizable, es decir que el entorno local externo permita la ejecución de dicha tarea. Por ejemplo: si la tarea seleccionada es *"caminar hacia delante"* pero en el ambiente hay un obstáculo entonces no se podrá realizar dicha tarea.

Planificador

Este componente del agente se encarga de organizar el orden de activación de los **efectores** de manera de llevar a cabo la acción especificada por la tarea que se encuentra en ejecución.

Es decir que el objetivo principal de este componente es armar un plan que coordine a los efectores indicados para plasmar en el ambiente la acción que envía la tarea ejecutante.

Efectores

Son los elementos con los cuales el agente se comunica con el ambiente, el punto de contacto entre el agente y el hábitat del mismo.

Los efectores reciben las ordenes del planificador y actúan directamente sobre el ambiente, modificándolo, generando eventos ó influyendo en el estado interno de otro agente o conjunto de agentes.

Se pueden ver como la contrapartida de los **sensores**. Mientras que los sensores perciben el ambiente y obtienen la **entrada** del agente, los **efectores** son elementos que plasman la **salida** del agente en la simulación.

3.4. Interacción entre componentes

En esta sección se detallan las diferentes relaciones entre los componentes del agente. La articulación de estas relaciones formaran una idea más cabal de cómo se comporta cada una de dichas componentes y que colaboradores posee para cumplir con sus tareas.

Objetivos – Tareas

Para satisfacer ciertos objetivos debe haber ciertas tareas.
Por ejemplo: Si hay un objetivo "incrementar energía" entonces debe haber una tarea "comer" o "buscar comida".

Si el agente no posee las tareas que satisfagan los objetivos tendrá que encontrar, a través de la **comunicación** (*ver Capítulo 5*) algún agente que le provea de dicha tarea o trabajo, es decir tendrá que colaborar con otro agente o conjunto de agentes para satisfacer sus objetivos.

Se tendrá que definir entonces la relación entre tareas y objetivos,
Por ejemplo: *La tarea **X** cumple en un **Y%** el objetivo **Z**.*

Señales – Sensores - AgentBrain

El Flujo de Señales que sirven de entrada al agente puede graficarse de la siguiente manera:

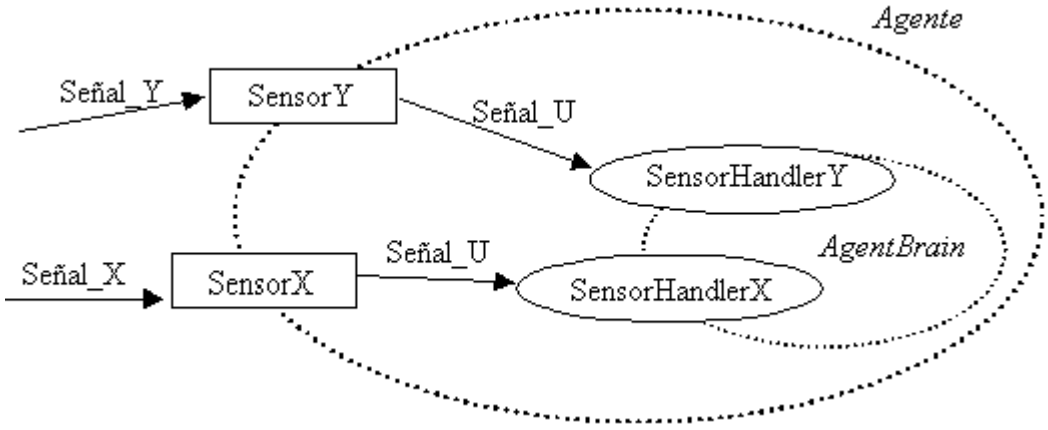

Figura 16 – Flujo de señales desde el ambiente

Como se puede observar en la *Figura Nº 16* cada tipo de señal tiene un sensor que la "comprende", es decir sabe interpretarla y codificarla en el lenguaje interno del agente (USignals = Señal_U) cada señal unificada a su vez va acompañada de un **SensorHandler** que sabe interpretarla y almacenarla en el AgentBrain.

Tareas – Planificador - Efectores

La relación entre estos tres componentes hace que el agente realice efectivamente las acciones sobre el ambiente.

Una tarea consta de una secuencia de acciones. Cada acción puede involucrar intervenir directa o indirectamente en el ambiente por lo que se necesita una influencia sobre el mismo.

Para llevar a cabo efectivamente las acciones que influyen en el ambiente se necesita organizar y ejecutar o activar las partes del agente que tienen contacto con el mismo de manera de lograr los efectos deseados.
De esta manera para cumplir una acción especificada en la tarea activa el **planificador** organiza a los efectores y planea la secuencia y forma en que serán activados para influir adecuadamente en el ambiente.

3.5. Diseño del Componente agente

El componente **agente** del Framework se divide principalmente en tres capas:

VRAgent : Modela la estructura interna general del agente.

VRAgentSensor: Especifica la conexión del agente con el ambiente.

VRAgentType: Modela los aspectos biológicos específicos del dominio.

Notar que las dos primeras capas corresponden a las características generales en tanto que la tercera a características específicas del dominio.

VRAgent

Este conjunto de Clases modela la estructura interna general de cada agente, independientemente de su tipo. Es decir la arquitectura general de los agentes en cuanto a funcionamiento de **objetivos** y **tareas** se refiere. Se observa en el siguiente diagrama la conexión entre tareas y objetivos.

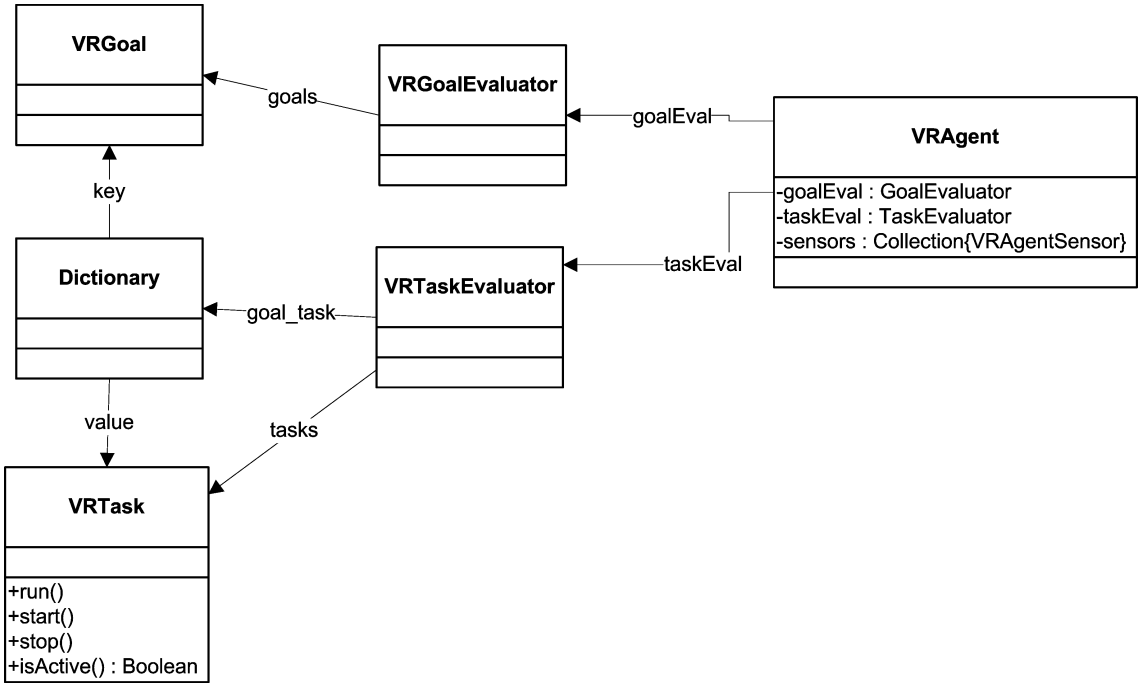

Diagrama 1 – Diseño de la capa VRAgent del Agente

Descripción de cada Clase

- ✓ *VRAgent:*
  Clase que establece la estructura general de todo agente. Define la arquitectura elegida y que se utilizará como soporte en el Framework.

- ✓ *VRGoalEvaluator:*
  Como se discutió en puntos anteriores esta clase representa el evaluador de objetivos que a partir del estado interno y las percepciones del agente determinará que objetivos deberán cumplirse en los próximos pasos de la simulación.

- ✓ *VRTaskEvaluator:*
  Evaluador de tareas que en base a los objetivos a cumplir, las percepciones y el estado interno determinará cuál es la **"mejor tarea"** que se adapta a los requerimientos.

- ✓ *VRAgentGoalWrapper:*
  Este tipo de objetos agregan comportamiento adicional a los objetivos sin afectar su comportamiento. Por ejemplo: serán responsables de mantener el umbral tolerable por el objetivo, la prioridad del mismo para facilitar las tareas de scheduling de objetivos.

- ✓ *VRAgentTaskWrapper:*
  Ídem el Wrapper anterior pero para la tarea. Contendrá por ejemplo: recursos que necesita la tarea, etc.

- ✓ *VRAgentTask:*
  Representa una tarea dentro del Framework.
  Determina la estructura general que tendrá que tener una tarea para poder ser administrada por el Framework.
  Encapsula un conjunto de acciones que se ven como una unidad lógica.

- ✓ *VRAgentGoal:*
  Representa un objetivo. Establece el protocolo a implementar por cualquier objetivo en el Framework.
  Es una premisa que, en base a la información del ambiente y el estado interno del agente, determina en cuanto grado se adapta esa situación a

la necesidad del agente. Puede verse como una función de fitness que mide una característica particular del agente.

VRAgentSensor

Este conjunto de Clases muestra la conexión del agente con el medio ambiente *(referiremos más sobre esto en capítulos subsiguientes: Capítulo 4 y 5).* También se aprecia la relación que hay con las señales y el AgentBrain como repositorio de información externa.

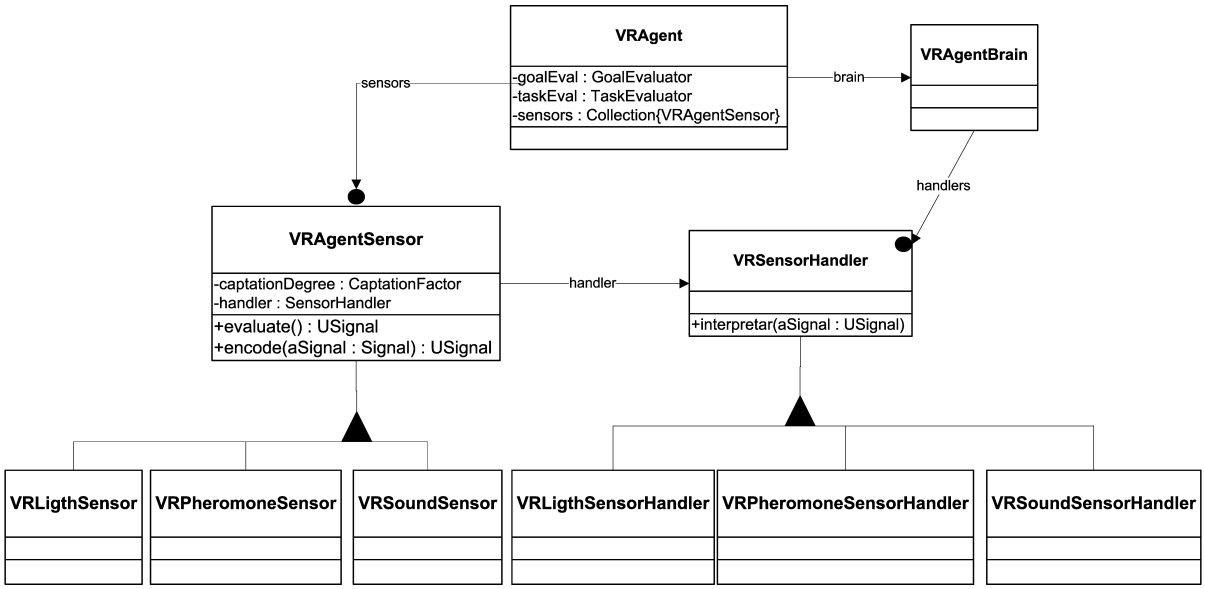

Diagrama 2 – Diseño de la capa VRAgentSensor del Agente

Descripción de cada Clase

- ✓ *VRAgentSensor:*
  Su objetivo principal es el de proveer al agente una forma de asimilar cierto tipo de señal externa e interpretarla de manera que dicho agente pueda extraer información útil del medio ambiente que lo rodea.
  Transforma una señal del tipo que comprende a una señal interna al agente.

- ✓ *VRAgentSensorHandler:*
  Es una parte del VRAgentBrain y se encarga de interpretar las señales codificadas por el VRAgentSensor correspondiente.

- ✓ *VRUSignal:*
  Señal interna que transporta en forma unificada la información del exterior.

- ✓ *VRSignal:*
  Señales externas que existen en el exterior.

VRAgentType

El siguiente conjunto de clases modela el concepto de Tipo de Agentes o Especies de agentes. Las características generales y comunes que comparten todos los agentes "vivos".

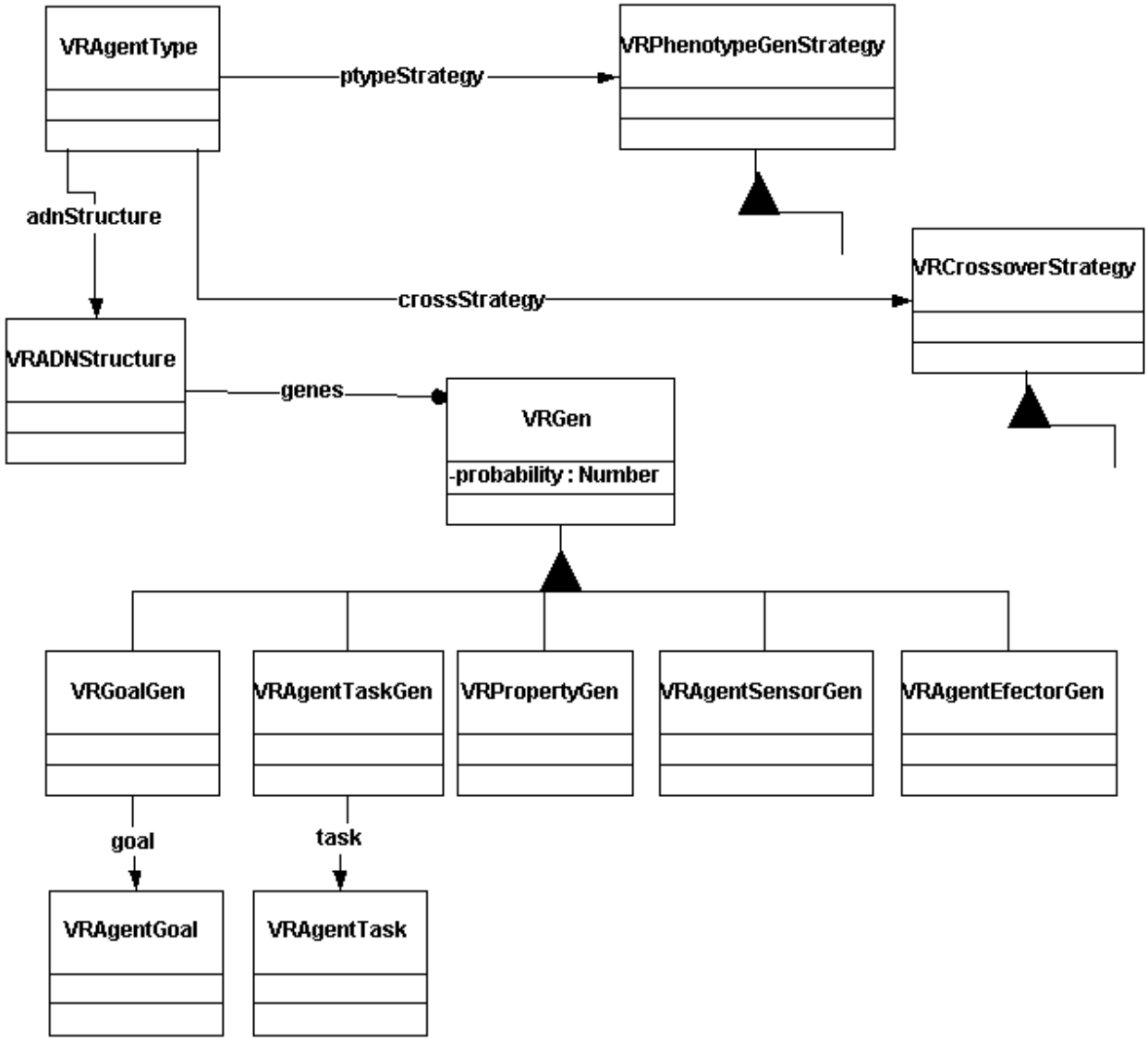

Diagrama 3 – Diseño de la capa VRAgentType del Agente

Descripción de cada Clase

- ✓ *VRAgentType:*
  Define un tipo especifico de agente. Establece la estructura del genotipo que tendrán los agentes del tipo o especie , tareas posibles, objetivos innatos al tipo, etc.

- ✓ *VRPhenotypeGenStrategy:*
  Es una estrategia de generación de un **fenotipo** a partir de una estructura de ADN.

- ✓ *VRCrossoverStrategy:*
  Es una estrategia de cruzamiento o reproducción. Establece los pasos a seguir para crear dos individuos de un tipo determinado a partir de sus dos **genotipos**.

- ✓ *VRGen:*
  Superclase de todos los genes posibles. Determina una característica puntual del agente que lo posee. Hay diferentes tipos de genes, algunos establecen los objetivos, otros las tareas, otros simples valores que se transmiten de generación en generación.
  El conjunto de VRGen's forma el **genotipo** del agente que lo contiene.

- ✓ *VRADNStructure:*
  Define la estructura u organización del conjunto de VRGen que determinan que un agente pertenece al tipo.

3.6. Hot Spots: Puntos Extensibles en el Framework

Los puntos en los cuales los usuarios del Framework podrán extenderlo son:

- ✓ *VRAgentGoal:*
  Extendiendo esta clase y creando una jerarquía de objetivos se podrá proveer a los agentes una guía de comportamiento más especifica de la aplicación.

- ✓ *VRAgentTask:*
  Creando una jerarquía de tareas el usuario podrá agregar nuevas funciones que sabrán ejecutar los agentes.

- ✓ *VRAgentSensor:*
  Extendiendo esta clase podrán agregarse nuevas formas de captar información o nuevos tipo de información existente en el ambiente.
  El mecanismo principal es proveer un algoritmo de transformación de una señal externa (VRSignal) a una VRUSignal interna al agente.
  Además de esta clase el usuario deberá proveer el correspondiente VRSensorHandler que interpreta la VRUSignal generada.

- ✓ *VRSignal:*
  Respetando el protocolo de esta clase se podrán agregar nuevos tipos de señales.

Lo que el Framework con respecto al agente, y por lo tanto el usuario debería utilizar como soporte son los siguientes puntos:

- ✓ Manejo de Señales (externas e internas): Transporte por el ambiente, incidencia en los sensores correspondientes.

- ✓ Scheduling de Tareas: Evaluación de la mejor tarea en base a los objetivos, administración de la ejecución de las mismas.

- ✓ Mecanismos de evaluación de objetivos y tareas que guían el comportamiento de los agentes.

- ✓ Soporte para la simulación.

Capítulo 4: Ambiente

*Se describen las características principales del **componente ambiente** y se enumeran las propiedades que el ambiente debe proveer al mundo virtual.*

*Se presenta la "**estructura física** del ambiente" y sus elementos principales (**recursos**, **componentes estáticos** y **dinámicos**). En particular, se detallan los mecanismos de representación para recursos y **señales**. Entre ellos la utilización de **layers** para configurar estos elementos.*

Se diseñan las clases de objetos para modelar el ambiente con su correspondiente descripción.

En los mundos artificiales y de acuerdo a nuestra investigación previa, podemos observar que los Ambientes son pocas veces estudiados.

En general, en los sistemas multi-agentes y en los dominios de vida artificial, el investigador enfoca su esfuerzo alrededor de las entidades activas, los agentes.
Son pocos los trabajos que se dedican a tratar el tema en forma exclusiva. Algunos de ellos, los encargados de dar los primeros pasos respecto al tema, definen técnicas de evolución para ambientes y las diferentes maneras de interacción entre ellos.

Para ellos, los ambientes no sólo son vistos como soporte dinámico donde poblaciones o animales evolucionan, sino que son entidades por sí mismo. Estas entidades permiten elaborar modelos de comportamientos particulares. Estos ambientes pueden evolucionar siguiendo las diferentes intersecciones de sus objetos y agentes.

Algunos van incluso más allá, investigando acerca de la relación entre ambientes, multi-ambientes, sub-ambientes y meta-ambientes.

Toda entidad viva está intrínsicamente relacionada con su ambiente, variando su forma de interacción y consecuente adaptación que lo define como un individuo distinto del medio.

El contexto o hábitat donde se desarrollan los hechos de un ecosistema no sólo influyen sobre el comportamiento de las criaturas que lo habitan sino que es el encargado de "contener" todos los objetos que forman el ecosistema. Dentro del ambiente también se desarrollan ciertos "hechos" o eventos, en algunos casos es importante que sean detectados por los objetos que forman parte del ecosistema, como por ejemplo: lluvias, la llegada de la noche, la llegada del día, los cambios de temperatura, etc.

Todos los organismos interactúan y captan energía del ambiente, los cambios de éste influyen directa o indirectamente en el curso de la evolución. El ambiente ejerce su influencia sobre los organismos.

En esta etapa del trabajo, debemos tener en cuenta cuestiones que pueden ocasionar resultados no esperados para la realización del Framework, y que son inherentes al proceso de mapeo de un ambiente real en contraste con un ambiente ideal en una simulación. Usualmente, un ambiente simulado es ideal para el sujeto (agente) bajo prueba, y esto no debiera pasar en nuestro caso. Es decir, debemos intentar no caer en el error de dotar al Framework con la capacidad de soportar un ambiente que favorezca o sea el ideal para sus habitantes, sino que debemos esforzarnos para que la realidad se refleje con la mayor certeza posible en la simulación.

Para poder entender y dotar al Framework con los elementos necesarios para proveer la funcionalidad asignada a un ambiente debemos estudiar con detalle todas las características que posee un ambiente para el tipo de mundos virtuales que nos interesa. Así, debemos responder desde una simple pregunta de qué es el ambiente hasta saber cuales son todos sus componentes y funcionalidades comunes que necesitaremos tener en cuenta.

4.1. ¿Qué es el ambiente?

El ambiente es la representación del hábitat donde "viven" los agente.

4.2. Características del ambiente

- ✓ En él ocurren eventos o situaciones que lo modifican.
- ✓ Posee determinadas condiciones dependiendo de muchos factores (zonas frías, cálidas, desérticas, etc.)
- ✓ Posee una estructura o topología que determina su forma geográfica.
- ✓ Puede ser modificado por los elementos de la simulación.
- ✓ Provee los recursos a las criaturas para que puedan sobrevivir.

4.3. ¿Qué necesitamos del ambiente?

Básicamente:

- ✓ Que sirva de hábitat para la vida de los agentes.
- ✓ Que sea contenedor de los elementos que componen el hábitat: comida, obstáculos, fenómenos climáticos, etc.
- ✓ Que pueda comunicarse con el agente para transmitirle los eventos que se producen en él y que afectarán al agente (clima, luz, características del terreno, etc.)

Para el modelado del Ambiente del Framework se deben considerar los siguientes aspectos:

- ✓ Modelado del "espacio físico" del ambiente.
- ✓ Elementos que contiene el ambiente:
  - Recursos: distribución, administración y provisión de los mismos al agente.
  - Características Estáticas – Consultas del agente.
  - Características Dinámicas: Señales – Estímulos para los agentes.
- ✓ Relación con el agente

4.4. Espacio Físico

Para responder al primer punto expresado anteriormente y considerando el aspecto del modelado del "espacio físico" del ambiente, tenemos que definir y diseñar aquellos items que nos permiten ver el ambiente como el lugar físico en el que se llevarán a cabo todas las actividades del Mundo Virtual.

En primer lugar, debemos dar un marco dimensional definido para poder trabajar de acuerdo al mismo. De esta manera, y como primer decisión respecto al ambiente que pretendemos modelar, definimos al ambiente de tamaño fijo en dos dimensiones (2D).

El usuario podrá definir las dimensiones del ambiente que quedarán configuradas en la clase VRHabitat. Luego, todos los objetos que intervengan en el mundo virtual deberán estar "contenidos" en el espacio que abarca el ambiente.

4.5. Elementos del ambiente

Los elementos principales que posee el ambiente son los **recursos** (ej: energía, agua, oxígeno), las **características estáticas** (ej: obstáculos, relieve del terreno) y las **características dinámicas** (ej: temperatura, presión, acidez, grado de contaminación).

Estos elementos se configuran a través de **layers** o capas. Cada layer se compone de un conjunto de Slices representados por figuras geométricas cuyas áreas se corresponden con algún área del ambiente. De acuerdo al tipo de elemento que represente, será de un tipo de layer determinado.

Se querrá por supuesto, definir cada una de las características comunes a un ambiente. Por lo tanto el usuario podrá crear nuevos layers para las características que le interese definir. El usuario no tendrá límites en sumar características al ambiente que quiere modelar, pero si deberá corresponder cada una de estas propiedades con un tipo de los layers propuestos en el Framework.

Es importante recordar aquí que cada uno de los slices que representen un elemento del ambiente, debe enmarcarse dentro del margen delimitado del ambiente y definido de antemano.

Esta forma de corresponder cada layer con una característica presente en el ambiente, ya sea estática, de recursos o dinámica, puede verse como la tercer dimensión que nos representa el ambiente. Estas capas nos permiten de alguna manera representar un modelo que simule un mundo virtual en sus tres dimensiones.

Con esta idea, podemos decir que en el modelo de la simulación que instancia el Framework, existirán tantos layers distintos como elementos diferentes que existan en el ambiente, cada uno por supuesto diseñados a través del tipo del layer correspondiente al tipo de elemento que se modele.

Así, podrá definir tres tipos de layers:

- Los layers estáticos (VRStaticHabitatLayer), que se componen de slices que representan sectores con valores comunes de la característica que representa el layer. La conexión de estos layers con los agentes es pasiva, es decir, será el agente quien será el interesado en preguntar el estado de ese layer.

- Los layers dinámicos o de señales (VRSignalHabitatLayer) se componen de un conjunto de señales que se mapean en slices dinámicos cada vez que se emiten. Es decir, la señal se crea y su onda de propagación forma un slice que es representado por una figura geométrica dentro del rango del tamaño del ambiente. El resto del lugar del layer que no tenga señales activas quedará con una señal de valor nulo o por defecto.

- Los layers de recursos (VRResourceHabitatLayer) que representen la distribución de los recursos que el ambiente posee y que están a disposición de los agentes cuando estos lo requieran. Puede verse que la dimensión o área ocupada por el recurso se corresponde con un slice dentro del layer.

La *figura Nº 17* muestra gráficamente el modelo por capas para representar las características del ambiente. La distribución de los recursos, señales y las características estáticas se representan a través de diferentes layers.

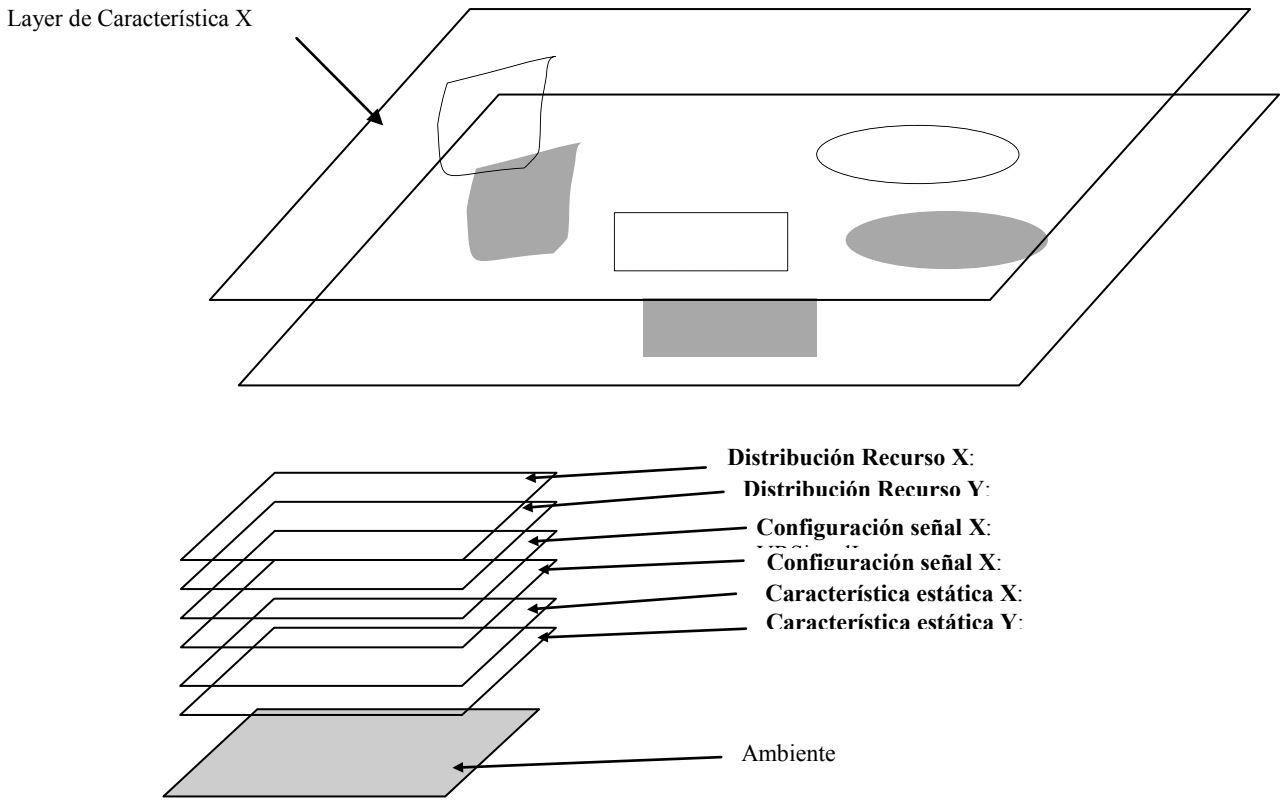

Figura 17 – Elementos del ambiente a través de Layers

4.6. Los recursos en el ambiente

Mencionábamos anteriormente a los **recursos** como uno de los elementos que componen el ambiente. Debemos acentuar aquí el énfasis en describir la importancia y el rol fundamental que los recursos cumplen dentro del mismo.

La cantidad de vida que cada ambiente puede sustentar está limitada por sus recursos básicos: la afluencia de energía, minerales y agua. La productividad sostenida de un ecosistema requiere suficiente energía para que se sinteticen nuevos productos y también para reciclar completamente los residuos de los viejos.

El mundo contiene una gran diversidad de condiciones físicas, las cuales crean una amplia variedad de ambientes.

En cada parte del ambiente habitable, los diferentes organismos compiten por comida, espacio, luz, calor, agua, aire y abrigo.

Debemos contemplar una adecuada representación para los recursos del ambiente, de manera que pueda reflejarse en la simulación un comportamiento similar al que se lleva a cabo en un ambiente real.

Habiendo analizado las maneras con las que diferentes Frameworks implementan el manejo de recursos, nuestras decisiones se orientan a diseñar este aspecto con la idea de mapas de recursos.

Los mapas de recursos son objetos que representan alguna distribución de recursos en el espacio.

En nuestro caso, nuestra manera de diseñar los elementos del ambiente a través de layers se corresponde perfectamente con la idea de mapas de recursos. El layer del tipo VRResourceHabitatLayer puede verse como un ResourceMap (Framework Evo analizado en *Capitulo 2*) que representa la distribución de sus objetos recursos (VRResource) dentro del ambiente.

Continuando con la idea descripta anteriormente, para el caso de los recursos, podemos decir que existirán tantos layers de recursos diferentes como tipos de recursos distintos se modelen.

Un VRResourceHabitatLayer contendrá entonces objetos del tipo VRResource representando la distribución de este tipo de recursos en el ambiente.

Los recursos y su actualización

Las características del medio ambiente no son eternas (en nuestro planeta se presentan por ciclos definidos).

Igual sucede con la vida, la evolución adapta a un ser vivo a un determinado ambiente, con respuestas acordes a los valores de entrada del ambiente (temperatura, humedad, presión, calidad del terreno, etc.), respuestas como estructura corporal, comportamientos o tácticas de supervivencia.

En un ambiente de agentes biológicos los recursos tienden a degradarse o cambiar su estado a través del tiempo. Para hacer que un recurso cambie a través del tiempo, se debe especificar un intervalo de actualización para ese recurso. En consecuencia se hace necesario que la infraestructura de la simulación aporte el

mecanismo adecuado para actualizar el estado de los recursos y, por ende, la presencia de una entidad dedicada a realizar esta tarea.

Por esto último se definen eventos especiales tendientes a actualizar el estado de los recursos para reflejar el paso del tiempo. Hablamos entonces de intervalos de actualización y de una entidad dedicada a realizar esta tarea: VRResourceStateUpdater.

Este componente se puede ver como un caso especial del Administrador de sucesos y eventos apuntado a administración de recursos.

4.7. Elementos dinámicos en el ambiente

Así como mencionábamos los recursos como elementos fundamentales para la productividad sostenida del ecosistema, debemos ahora realizar consideraciones similares para otro tipo de elementos que deben ser modelados si se quiere simular un mundo virtual.

Nos referimos a aquellos componentes del ambiente que tienen un comportamiento algo diferente de los recursos, ya que, a cambio de ser actores pasivos en cuanto a su relación con los agentes, su manera de actuar consiste en influir de diferentes manera en los agentes que habitan en el ambiente.

A pesar de un comportamiento algo diferente al de los recursos, e intentando mantener cierta coherencia en el diseño del Framework, pensamos que la distribución de estos componentes dinámicos en el ambiente puede comportarse de igual manera que las características estáticas y los recursos. Veremos sí, mas adelante, que la diferencia consistirá en la tasa de actualización, la variabilidad y la relación con el agente que tienen estos elementos.

En el caso de este tipo de layers, cada vez que se genera una señal se deberá informar a los agentes comprendidos dentro del espacio que abarca el slice dinámico que se ha creado, es decir, la conexión de estos layer con los agentes es activa, ya que es el mismo layer el que brinda los datos al agente.

Las señales

Debemos encontrar una estructura adecuada para representar estas señales que se generan en el ambiente y que generalmente afectan al resto de los elementos del

mundo virtual. Al igual que los recursos y que los elementos estáticos, las señales también deben caracterizarse por el especio físico que ocupan y de esta manera saber a cuales elementos afecta y en que medida lo hace.

La clase VRSignal entonces representará una señal en el ambiente y su distribución en él estará modelada a través de los layer dinámicos (VRSignalHabitatLayer).
Un VRSignalHabitatLayer contendrá entonces objetos del tipo VRSignal representando la distribución de señales en el ambiente.

La señal se crea y su onda de propagación forma un slice que es representado por una figura geométrica dentro del rango del tamaño del ambiente. El resto del lugar del layer que no tenga señales activas quedará con una señal de valor nulo o por defecto.

Quizá aquí podamos hablar también de mapas, en particular de mapas de señales, aunque en este caso la configuración y el snapshot del layer que representa el mapa tendrá seguramente mayor volatilidad en cuanto a la permanencia que los layers de recursos y, mas aún, que los layers estáticos.

Las señales y su generación

Es necesario proveer la estructura necesaria para simular la continua generación de fenómenos en el ambiente que hemos llamado señales o elementos dinámicos. Así, se definen eventos orientados a simular la generación y dispersión de las señales que ocurren en el ambiente. Hablamos entonces de objetos VRSignalGenerator, cuya funcionalidad será básicamente "crear" señales de distintos tipos, aportarles sus características particulares y distribuirlas adecuadamente en el espacio físico del ambiente a través de sus correspondientes layers.

Este componente también puede verse como un caso especial del Administrador de sucesos y eventos apuntado a la generación y administración de señales.

4.7. Diseño del Componente ambiente

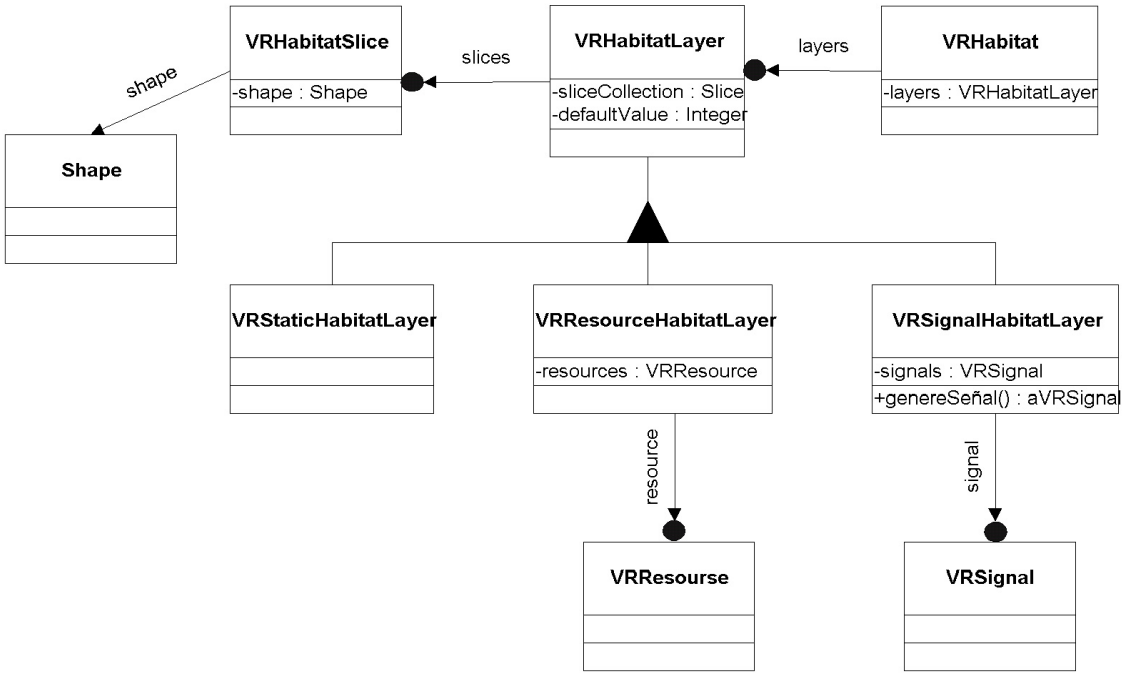

Diagrama 4 – Diseño del componente VRHabitat

Descripción de cada Clase

- *VRHabitat*

  Representa el espacio que ocupa el mundo virtual. Es un gran contenedor del resto de los objetos y básicamente define los límites del lugar físico a utilizar.

- *VRHabitatLayer*

  Representa la distribución en el ambiente de sus diferentes elementos.

  Es la superclase del resto de los tipos de layers.

- *VRStaticHabitatLayer*

  Es un tipo de Layer que representa sus características estáticas. A través de este tipo de Layer, el usuario puede configurar aquellas características que no sufren cambios significativos durante la vida del mundo virtual, como por ejemplo la presencia de obstáculos.

- *VRResourceHabitatLayer*

  Es un tipo de Layer que representa la distribución de los recursos en el ambiente. Cada layer de este tipo posee una colección de objetos **VRResource**.

- *VRSignalHabitatLayer*

  Es un tipo de Layer que representa la generación de señales en el ambiente. Cada layer de este tipo posee una colección de objetos **VRSignal**.

- *VRResource*

  Objetos que representan recursos del ambiente y que le son necesarios a los agentes.

- *VRSignal*

  Objeto que representa una señal generada en el ambiente y que puede afectar a los agentes.

- *VRHabitatSlice*

  Representa una porción del espacio físico del ambiente. La unión de todos los slice de un Layer forman la superficie ocupada por el VRHabitat. Cada Layer se divide en slices.

  Cada VRHabitatSlice utiliza un objeto Shape para representar su área.

Capítulo 5: Relaciones

Desarrollamos en este capítulo las diferentes relaciones entre los principales componentes del Framework.

Habiendo tratado los componentes principales en forma individual, se intenta aquí explicar las relaciones que ocurren entre ellos y unificar los conceptos en común que estos componentes tienen.

Se detectan las diferentes interacciones entre los componentes. Se profundiza luego analizando cada una de estas interacciones detallando que elementos entran en juego y como se relacionan.

Se grafican además los ciclos surgidos en las relaciones y al finalizar el capítulo se diseñan las clases y se describen sus propósitos.

Enumeramos a continuación las interacciones que surgen de un primer análisis sobre el tema. Mas adelante nos referimos específicamente a cada una de ellas:

- ✓ Ambiente – Agente

  Esta relación se origina como consecuencia de la generación de señales en el ambiente y que influyen en el comportamiento y estado de los agentes.

- ✓ Agente – Ambiente

  Recíprocamente esta relación viene dada por la generación de señales desde el agente hacia el ambiente a través de sus efectores.

- ✓ Agente – Agente

  Este tipo de relación se refieren a la comunicación e interacción entre agentes.

- ✓ Agente – Recursos

  Relaciones que modelan la necesidad de los agente de utilizar recursos para llevar a cabo sus tareas.

La *figura Nº 18* muestra un esquema general de los elementos de contacto que intervienen en la relación de estos principales componentes: los layers del Ambiente con los sensores del Agente.

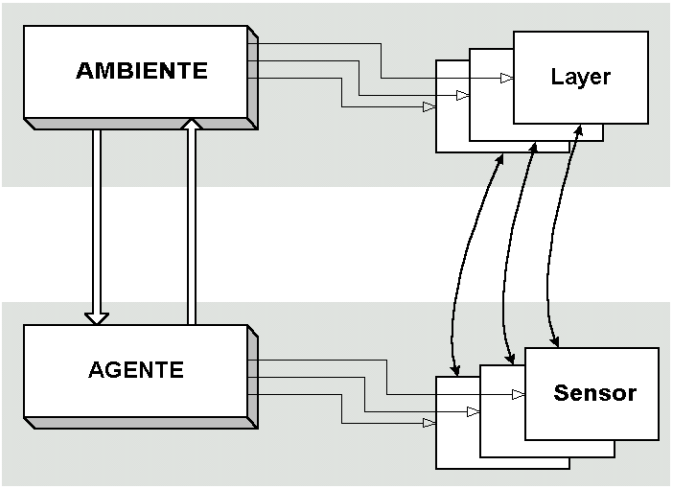

Figura 18 – Relación entre componentes

5.1. Relación Ambiente – Agente

Dijimos ya que toda entidad viva está intrínsicamente relacionada con su ambiente y que los eventos que en él suceden influyen de forma directa o indirecta en los agentes que lo habitan.

Esta relación se modela en el Framework a través de dos jerarquías que mantienen cierta correspondencia. Una de ellas desde el lado del ambiente y otra desde el agente.

- ✓ VRSignal

  Generadas desde el lado del Ambiente. Representan las distintas señales que circulan en el ambiente y potencialmente pueden interesar y/o afectar a los agentes.

- ✓ VRAgentSensor

  Los agentes poseen distintos tipos de sensores para los distintos tipos de señales que le interesan.

Es necesaria la correspondiente existencia de una VRSignal con su respectivo VRAgentSensor. Por ejemplo, existen VRLightSignal y su correspondiente VROpticalSensor.

Los distintos VRAgentSensor estarán conectados con su correspondiente VRSignalHabitatLayer, que mantiene los tipos de señales que ese sensor puede captar.

A diferencia del Framework Bubble, en el Framework que modelamos no existe un registro de señales – sensores, sino que la relación está dada por la correspondencia entre VRAgentSensor con su VRSignalHabitatLayer.

Básicamente el mecanismo de notificación de señales a los sensores se realiza de la siguiente manera:

- ✓ Se genera una señal en su layer correspondiente.
- ✓ Se obtienen los agentes afectados por el área de propagación de la señal (VRHabitat consulta a VRSimulationEngine)
- ✓ Se informa a este conjunto de agentes de la señal que los afecta.
- ✓ Aquellos que posean el sensor adecuado actuarán en consecuencia.

La *figura Nº 19* muestra un esquema general de la relación entre el Agente y el ambiente a través las señales (VRSignal) del lado del ambiente y de los sensores (VRSensor) del lado del Agente.

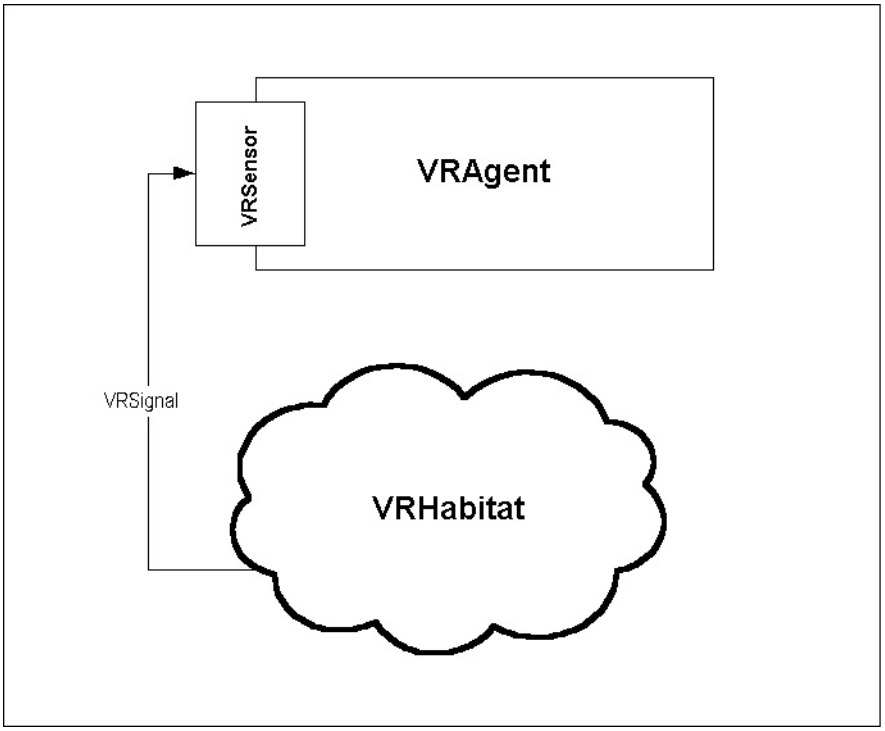

Figura 19 – Ciclo VRHabitat – VRSignal – VRAgent

5.2. Relación Agente – Ambiente

Recíprocamente, esta relación viene dada por la generación de señales desde el agente hacia el ambiente a través de sus efectores.

El agente modifica o influye en el ambiente de varias formas, las más significativas son:

- Consumiendo sus recursos

  El agente necesita recurso para llevar a cabo sus tareas. El consumo de esos recursos constituye una modificación en los elementos del ambiente. Esta acción, sumada a la propia degradación de los recursos conllevan la desaparición y posterior auto-renovación de los mismos.

- Modificación directa del ambiente

  Todos los objetos que intervengan en el mundo virtual deberán estar "contenidos" en el espacio que abarca el ambiente. Los agentes, como tales, pueden producir, con su presencia, modificaciones que

impactan directamente en el ambiente y lo modifican. Por ejemplo, recolección de objetos, cambios en la configuración del ambiente debido a su movimiento, etc. Incluso un mismo agente puede ser visto por otro como un obstáculo, siendo el caso este en el que ese agente pasa a ser "parte" del ambiente.

- Generación de señales

Así como el ambiente influye en el agente a través de señales, el agente influye sobre el ambiente generando nuevas señales.
El efecto que produce el agente sobre el ambiente es generalmente menor que su contraparte, pero no por esto debemos obviarla.
Ejemplos de este caso lo constituyen las señales sonoras y señales de comunicación (ver Relación Agente-Agente).

La *figura Nº 20* muestra un esquema general de la relación entre el Agente y el ambiente a través las señales (VRSignal) del lado del ambiente y de los efectores (VREfector) del lado del Agente.

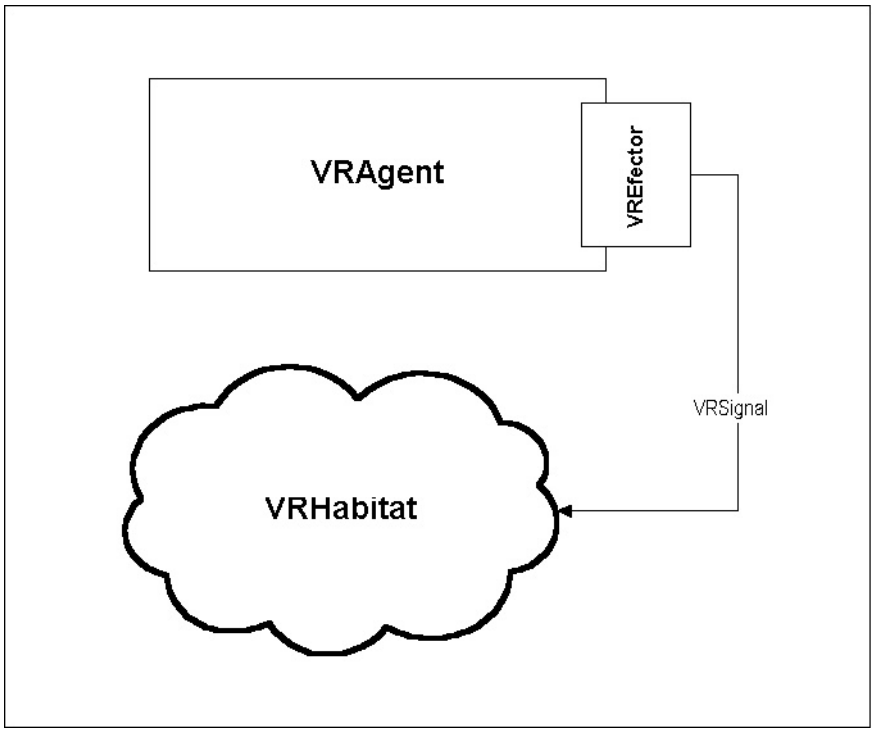

Figura 20 – Ciclo VRAgent – VRSignal – VRHabitat

5.3. Relación Agente – Agente

La forma de comunicarse y relacionarse entre agentes viene dada por la generación de señales y eventos a través de los efectores y captación de las mismas por los sensores del agente receptor.

Si un agente quiere comunicarse con otro agente o conjunto de agentes, este proceso de comunicación básicamente consiste en un ciclo de generación de señal o evento por parte de un agente y consumición de dicha señal por los sensores de los agentes receptores.

En este punto se distingue un nuevo tipo de señal (VRComunicationSignal) destinada a modelar la comunicación entre agentes. Esta señal respeta el protocolo del mecanismo de señales dentro del ambiente, pero posee además estado y comportamiento propio acerca de la transferencia de información entre dos agentes. Así, VRComunicationSignal extiende la funcionalidad de la clase VRSignal.

De igual manera que las demás señales y manteniendo una coherencia las mismas serán percibidas por sensores específicos en el agente receptor, aquél agente que no posea este sensor sencillamente no "entenderá" dicha señal o lenguaje. Esto modela el hecho de que en el mundo real hay especies o tipos de criaturas que poseen distintas formas de comunicarse.

Nota:
En el C*apitulo 6 – Infraestructura para la simulación* nos referimos a tipos de interacción entre agentes.

La *figura Nº 21* muestra un esquema general de la relación entre Agentes y Ambiente a través de señales, sensores y efectores.

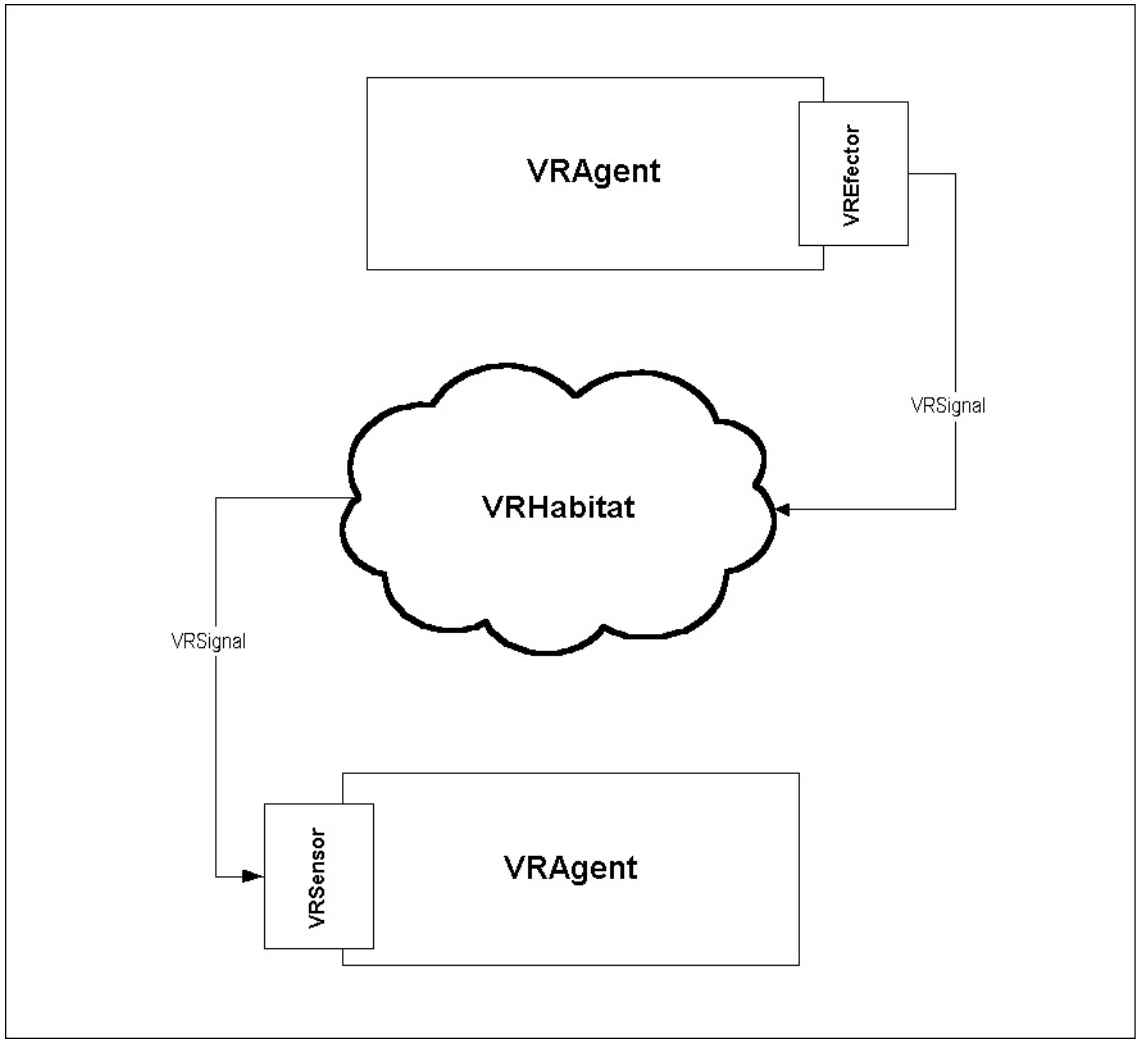

Figura 21 – Ciclo VRAgent – VRHabitat – VRAgent

5.4. Relación Agente – Recursos

Los agentes necesitan recursos para poder cumplir sus tareas.

Este componente es el encargado de proveer los mecanismos para administrar los requerimientos de recursos, coordinar la ubicación y la correspondiente asignación de los mismos.

Un VRResourceProvider será el objeto receptor de los pedidos de recursos de los agentes. Cuando un agente necesita obtener un recurso del ambiente convierte su requerimiento en un VRRequiredResource que recibirá el VRResourceProvider.

A través del manejo de una lista interna, el VRResurceProvider se encargará de despachar uno a uno los VRRequiredResource.

El VRResourceProvider localizará los recursos solicitados interactuando con los layers que representan los mapas de recursos que posee el ambiente (*ver Capítulo 4*).

Dentro de este proceso de localización de recursos el VRResourceProvider debe incluir además un mecanismo de respuesta en aquellos casos que no se ubiquen los recursos solicitados. Al agente le llegará una notificación de "no se encuentra el recurso", el agente deberá actuar según esta situación, llegando incluso a no ejecutar la tarea especifica para la cual necesitaba el o los recursos.

5.5. Diseño del Componente

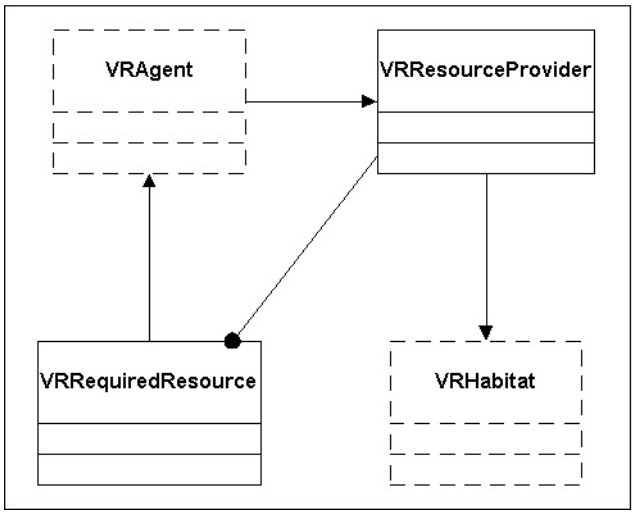

Diagrama 5 – Diseño de los Recursos del Framework

VRResourceProvider: Se encarga de proveer los recursos a los agentes que lo requieran. Posee una lista interna de los recursos solicitados. Colabora con los Layers de Recursos del Hábitat para ubicar y obtener los recursos.

VRRequiredResource: Representa un recurso requerido por un VRAgent.

Capítulo 6: Infraestructura para la simulación

El objetivo de este capítulo es analizar y diseñar los componentes necesarios para soportar una simulación en ejecución.

Se comienza analizando las necesidades para el soporte de una simulación. Se detectan y definen los componentes principales.

Se detalla cada componente encontrado y se realiza el diagrama de clases de cada uno de ellos:

Coordinador de la simulación: *coordina el acceso a recursos compartidos, controla la ejecución de la simulación, brindar una visión única de acceso a la simulación.*

Configurador de la simulación: *constituye una herramienta de ayuda para que el usuario pueda configurar fácilmente los distintos aspectos de la simulación*

Administrador de eventos y sucesos: *mantiene un conjunto de eventos y suceso y a su vez controla la activación de los mismos.*

Scheduling de recursos: *este componente es el responsable de administrar los eventos relacionados con la administración de recursos.*

Recolector de información: *es aquí donde se registrarán los distintos elementos de información a ser recolectados mientras la simulación corre.*

Deducción de información: *calcula la información útil que surja de la simulación y ayuda al usuario en el proceso de toma de decisión.*

6.1. Simulación

Una simulación puede ser definida como una representación de una función, operación o características de un proceso o sistema a través del uso de otro.

La simulación es una herramienta importante para estudiar el comportamiento de los sistemas. Existen además, dos conceptos relacionados con la simulación que son: el análisis de sistemas y la construcción de modelos.

El análisis de sistemas consiste en definir los límites del sistema que se va a modelar, identificar los elementos más importantes y sus interacciones y el entorno de cada uno de ellos, agrupándolos jerárquicamente.

Una vez analizado el sistema, se pasa a la construcción del modelo, que refleje la esencia del sistema original, aunque simplificándolo, sin alterar sus funciones principales.

Después de analizar el sistema y modelarlo, se utiliza la simulación para estudiar el comportamiento del sistema a través de la observación del modelo.

Con la simulación se pueden perseguir objetivos muy diferentes: determinar si las suposiciones sobre el funcionamiento del modelo son válidas, si los subsistemas identificados se corresponden con un funcionamiento real, efectuar predicciones sobre el comportamiento futuro del sistema, estudiar comportamientos erróneos del sistema, etc.

Patrones de comportamiento pueden emerger en el curso de ejecución de una simulación. Estos pueden ser completamente diferentes de una corrida a otra. Acciones repetitivas pueden ser utilizadas para operar una simulación, pero no deberían ser dirigidas a cualquier objetivo global específico.

Podemos identificar características de las simulaciones en general:

- ✓ Requieren ciertas configuraciones por parte del usuario antes de comenzar su ejecución.

- ✓ El objetivo es obtener información que surge de las configuraciones del usuario mas la ejecución de la simulación.

- ✓ Presencia de eventos y sucesos con variados grados de dependencia que al darse en forma simultánea necesitan cierta estructura que brinde soporte en la ejecución.

- ✓ Presencia de elementos requeridos por los agentes para llevar a cabo sus tareas, por ejemplo: recursos.

Analizando estas características surgen los siguientes componentes:

- Coordinador de la simulación
- Configurador de la simulación
- Administrador de eventos y sucesos
- Scheduling de recursos
- Recolector de información
- Deducción de información

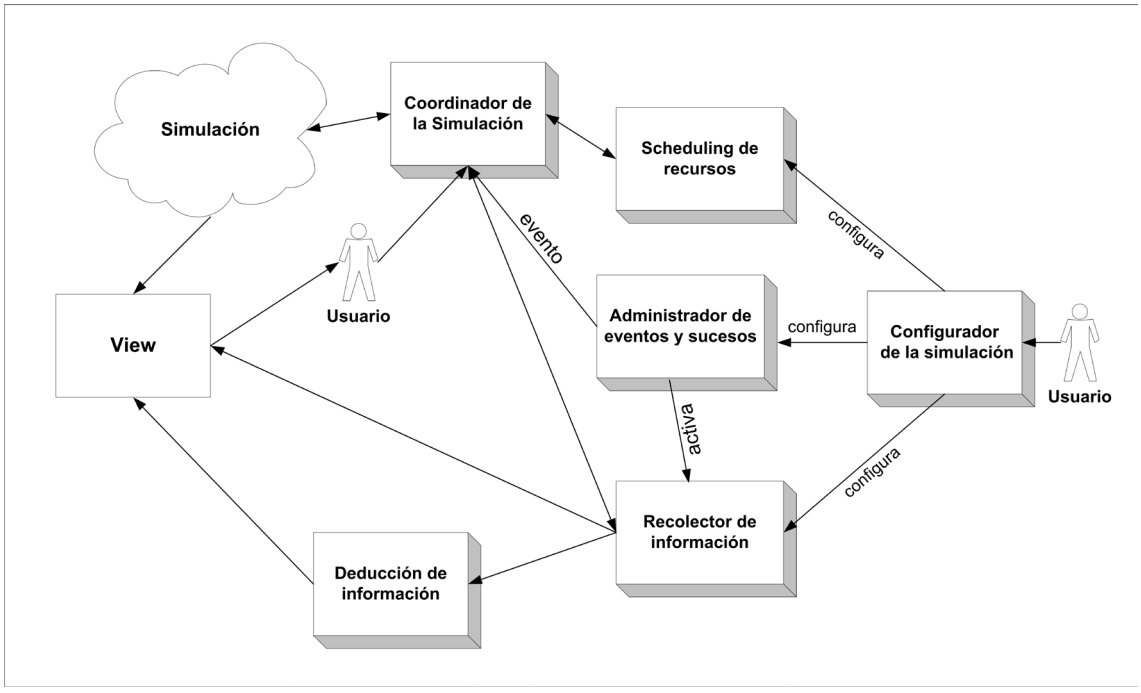

Figura 22 – Infraestructura para la simulación - Componentes

6.2. Coordinador de la simulación

Los agentes funcionan como un objeto autónomo, y a su vez necesitan interactuar y coordinarse con el resto de los componentes instanciados en la simulación.

El ambiente puede verse como un recurso único que es compartido y requerido concurrentemente. Constituye un punto crítico en la simulación que puede convertirse en un "cuello de botella" durante la ejecución.

Es necesario, además, contar con un protocolo de start, pause, resume y stop. Para lograr esto se debe tener una visión de la simulación como un objeto de alto nivel que sea capaz de recibir mensajes.

En consecuencia, los objetivos principales del Coordinador de la simulación son:

- ✓ Coordinar el acceso a recursos compartidos
- ✓ Controlar la ejecución de la simulación
- ✓ Brindar una visión única de acceso a la simulación.

6.2.1. Diseño del componente

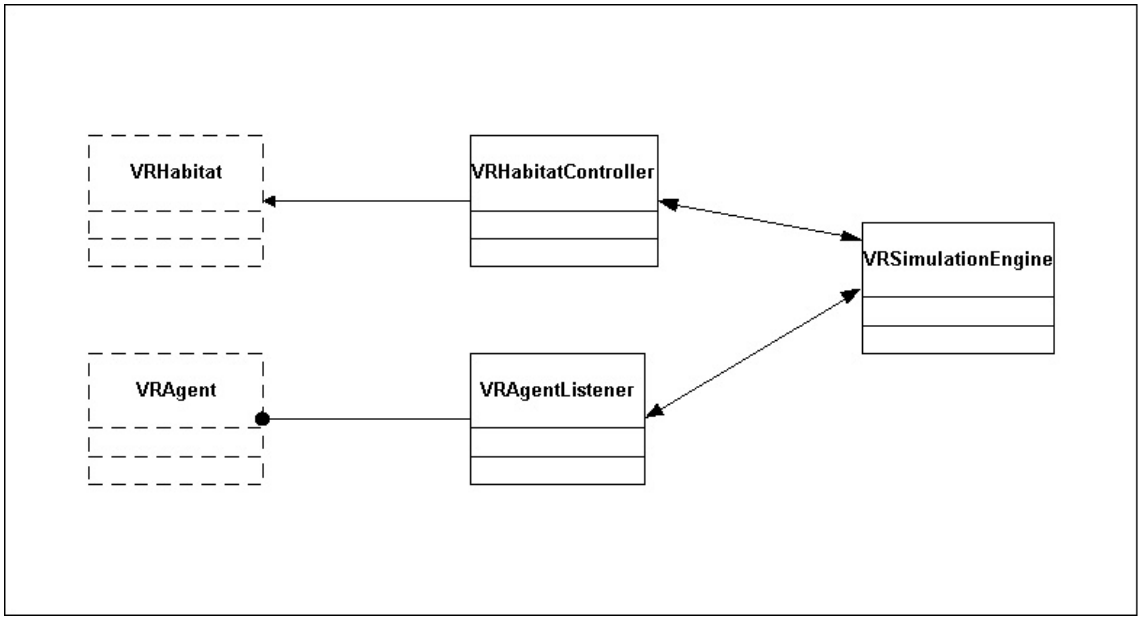

Diagrama 6 – Diseño del coordinador de la simulación

VRSimulationEngine: El objetivo principal de esta clase es la de administrar en forma general todo el proceso de simulación. Actúa como mediador y conoce a quien delegar cada aspecto particular de la simulación. Este componente es un a "puerta de entrada" a la simulación, es decir, es el punto de contacto con cualquier componente externo que necesite utilizar de alguna forma la simulación. Por ejemplo:

- ✓ Componente Interfaz Gráfica: para mostrar la simulación.
- ✓ Componente Recolector de Información: para obtener datos a procesar o analizar.

Básicamente se encargará de clasificar los mensajes desde y hacia los diferentes "modelos parciales", en forma similar a un dispatcher.

Nota:
VRSimulationEngine es una instanciación del *pattern* ***Mediator*** ya que su objetivo principal es el de recibir peticiones y derivarlas donde corresponda.

VRAgentListener: Esta clase es la encargada de registrar los diferentes sucesos generados desde los agentes e informárselos adecuadamente al VRSimulationEngine. Su funcionamiento se asemeja a un dependiente o listener del Agente.

VRHabitatController: Clase encargada de administrar los aspectos referentes a la conexión con el modelo de hábitat o ambiente. De igual manera que VRAgentListener informa a VRSimulationEngine de los sucesos del ambiente.

Tanto el VRAgentListener como el VRHabitatController son instanciaciones del *pattern* ***Observer***. Dependen u observan el estado de otro objeto sin incidir directamente sobre su estado.

6.3. Configurador de la simulación

Constituye una herramienta de ayuda para que el usuario pueda configurar fácilmente los distintos aspectos de la simulación. El usuario podrá a través de él armar la simulación en tiempo de diseño.

El configurador de la simulación provee la **vista diseño** de la simulación.

El coordinador de la simulación provee la **vista ejecución** de la simulación.

De aquí se extraerá la información necesaria para construir la infraestructura de la simulación antes de iniciar la ejecución.

Los objetos de esta componente están orientados a mantener el estado que poseerán otros componentes en tiempo de ejecución:

- ✓ Para Sheduling de Recursos mantendrá: mapa de los recursos, intervalo de tiempos, políticas de actualización, etc.

- ✓ Para el Administrador de Eventos y Sucesos, mantendrá una lista que poseerá los eventos y sucesos que deben dispararse durante la simulación y la lógica apropiada para llevar a cabo la ejecución de los mismos.

- ✓ Para el Recolector de Información mantendrá los diferentes items de información que serán calculados.

6.3.1. Diseño del componente

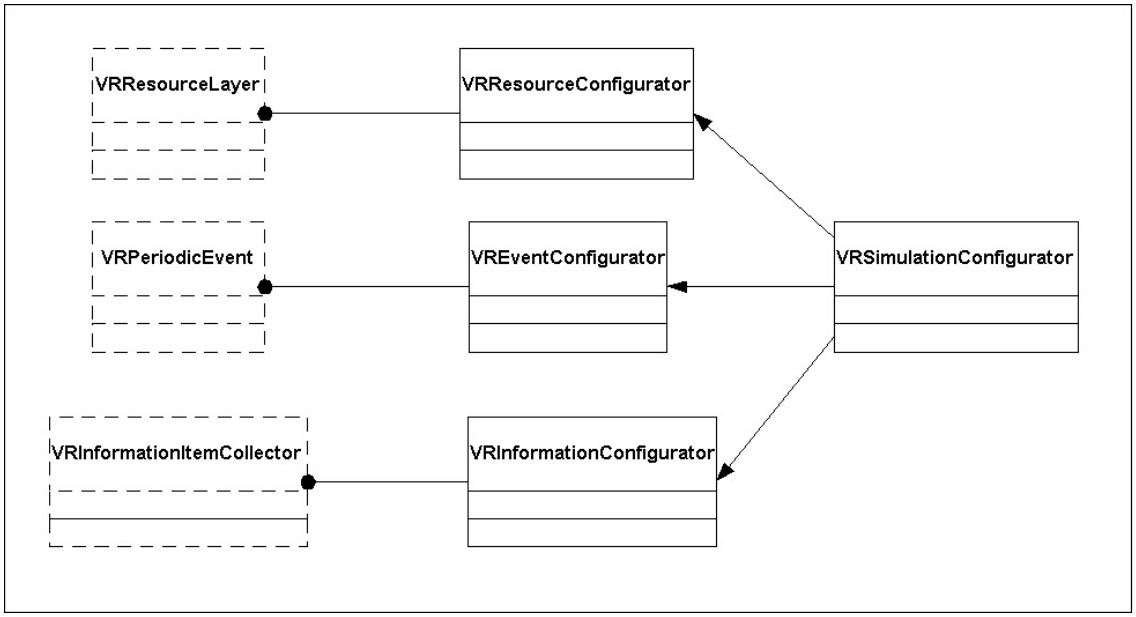

Diagrama 7 – Diseño del configurador de la simulación

6.4. Administrador de eventos y sucesos

Mantiene un conjunto de eventos y sucesos y a su vez controla la activación de los mismos según hayan sido configurados previamente. Pueden identificarse dos tipos de eventos:

- ✓ Eventos hacia el **Coordinador de la Simulación**: aquellos sucesos que inciden directamente sobre el estado de cualquier objeto dentro de la simulación (hábitat, agente, etc). Por ejemplo: activación de lluvia, cambio de temperatura, control del ciclo día-noche.

- ✓ Eventos hacia el **Recolector de Información**: para cada ítem de información que se haya configurado mantiene el intervalo o período en que se disparará la captura.

6.4.1. Diseño del componente

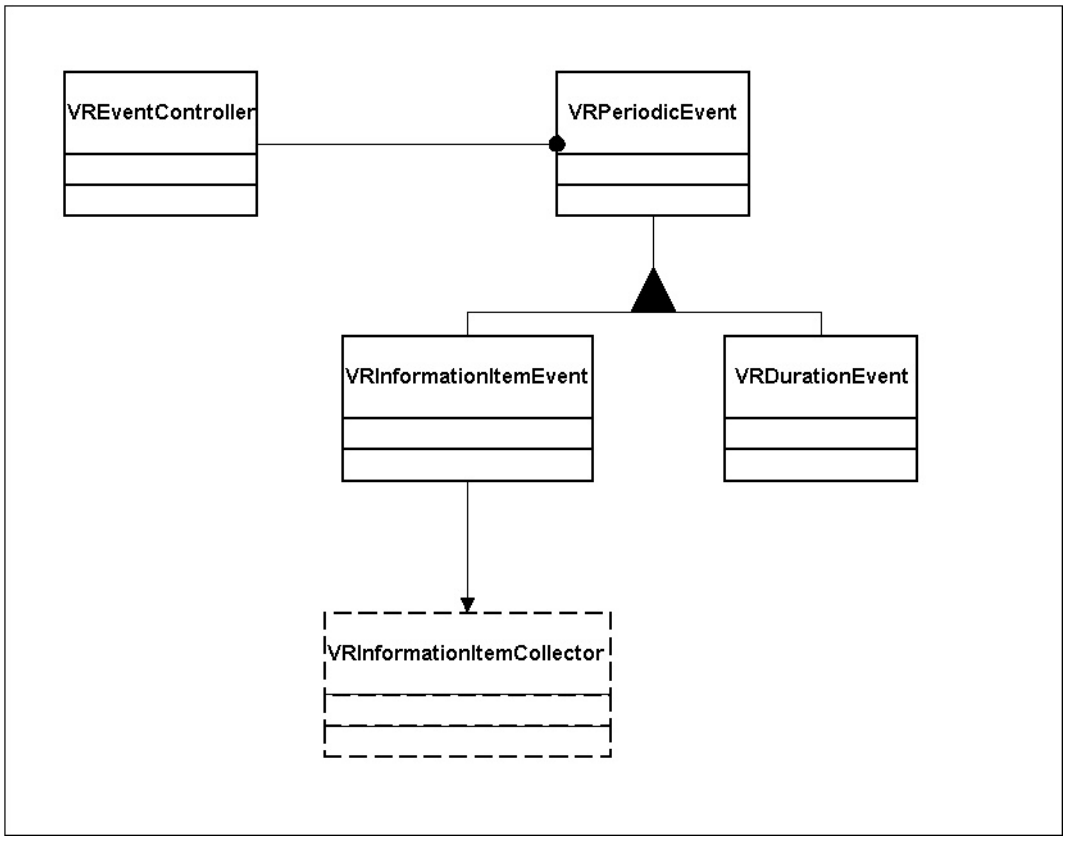

Diagrama 8 – Diseño del administrador de eventos y sucesos

VRPeriodicEvent: Responsable de ejecutar cierto evento en forma periódica a través del tiempo.

- ✓ VRDurationEvent: Responsable de ejecutar cierto evento en forma periódica y con una duración determinada a través del tiempo.

- ✓ VRInformationItemEvent: Es un evento relacionado con la activación de un Ítem de Información. Inicia la captura de información de un ítem específico.

VREventController: Esta clase administra, coordina y sincroniza los eventos necesarios para la simulación.

6.5. Scheduling de Recursos

Este componente es el responsable de administrar los eventos relacionados con la **administración de recursos**.

En un ambiente de agentes biológicos los recursos tienden a degradarse o cambiar su estado a través del tiempo. En consecuencia se hace necesario que la infraestructura de la simulación aporte el mecanismo adecuado para actualizar el estado de los recursos. Por esto último se definen eventos especiales tendientes a actualizar el estado de los recursos para reflejar el paso del tiempo.

Se complementa con la funcionalidad modelada en el *capítulo 5*.

6.5.1. Diseño del componente

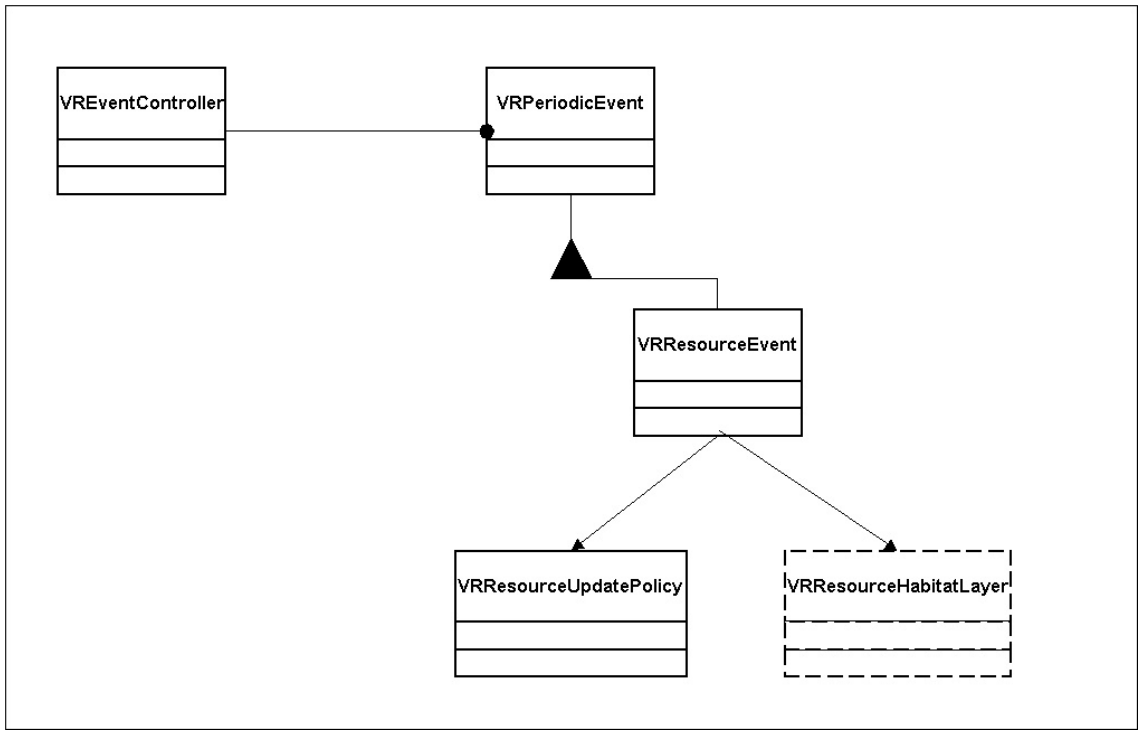

Diagrama 9 – Diseño de Sheduling de recursos

VRResourceEvent: Subclase de VRPeriodicEvent. Encargado de disparar la actualización del estado de los recursos contenidos dentro de un VRResourceHabitatLayer determinado. Cada VRResourceHabitatLayer representa un tipo de recurso determinado. Además colabora con un VRResourceUpdatePolicy.

VRResourceUpdatePolicy: Representa la política de actualización del estado de un recurso. Su protocolo es ***updateState(VRResource)*.** Este mecanismo necesita ser extendido por el usuario del Framework, quien deberá definir la lógica de actualización.

6.6. Recolector de información

Hay que tener en cuenta que la simulación es un proceso continuo y no determinístico, por lo tanto este componente debe ser capaz de capturar información sin detener la misma.

El punto que referíamos en el **coordinador de la simulación** acerca de tener una visión global de la simulación es también necesario en este componente, ya que necesitaremos tomar o recolectar información generada por la simulación para su posterior análisis.

Es aquí donde se registrarán los distintos **elementos de información** (VRInformationItemCollector) a ser recolectados mientras la simulación corre. El problema en particular que se esté simulando, sumado al interés del usuario configurarán los distintos elementos de información a recolectar.

La objetivo es que el usuario pueda configurar que es lo que quiere recopilar de la información que genera la simulación. Algunos estarán interesados en algún aspecto, otros en otros. Según el objetivo para el cual se corre la simulación dependerá de que datos quieren capturarse.

Cada elemento de información tendrá su propia lógica para capturar la información requerida. Se puede ver esto como un conjunto de reglas que establecen cómo capturar la información.

Se puede dividir la estructura de los elementos de información en las siguientes partes:

Estrategias de inicio de la captura

- ✓ Por interrupción, disparadas por eventos.
- ✓ Por muestreo o Polling: cada determinados períodos de tiempo.

Lógica o cálculo de la Información

- ✓ Funciones que se ejecutan para calcular esa pieza de información. El Framework provee las primitivas que el usuario puede utilizar.

- ✓ Entrada: simulación

Almacenar la información para su posterior análisis

6.6.1. Diseño del componente

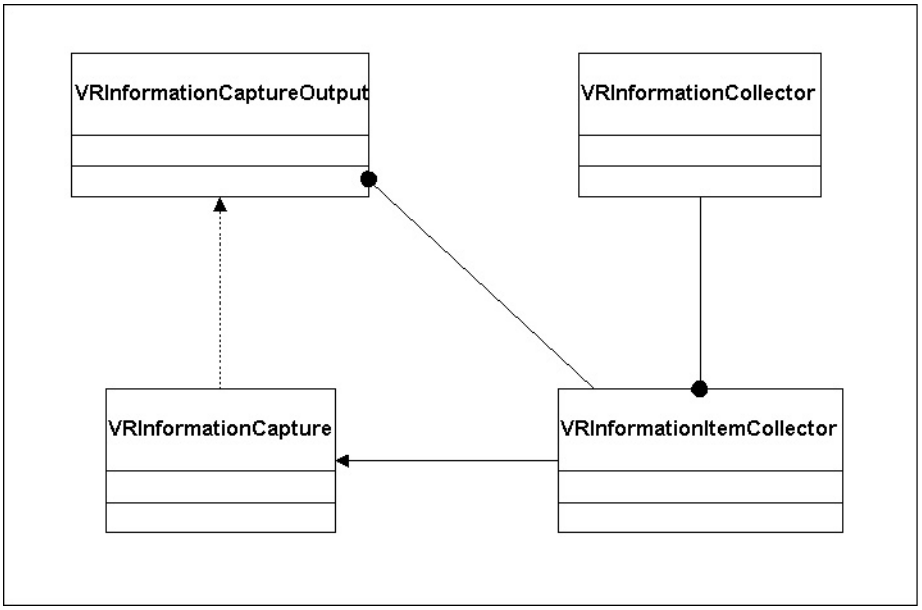

Diagrama 10 – Diseño del Recolector de Información

VRInformationItemCollector: Representa un Ítem de captura de información configurable por el usuario. La información aquí almacenada puede verse como una estructura de diccionario, donde la clave represente el momento en el tiempo en que se disparó la captura y su valor el elemento resultante de dicha captura.

VRInformationCollector: Contiene la lista de VRInformationItemCollector. Sirve como punto de entrada del Recolector de Información para los módulos que así lo requieran (por ej: Módulo **Deducción de Información**)

VRInformationCapture: Encapsula la lógica de la captura de Información. El usuario del Framework extenderá esta clase para proveer su lógica particular. El

protocolo es: ***capture(VRSimulationEngine)***. Debe retornar un VRInformationCaptureOutput.

VRInformationCaptureOutput: Representa la salida del proceso de captura.

6.7. Deducción de información

Uno de los objetivos principales de toda simulación es aportar información que ayude a la toma de decisión en escenarios con cierto grado de incertidumbre y no determinísticos (no modelables a través de formulas o relaciones rígidas).

Este componente será responsable de calcular la información útil que surja de la simulación y ayude al usuario en el proceso de toma de decisión.

Entre los problemas que surgen al momento de modelar este componente se encuentran:

- ✓ ¿Que obtener como salida de la simulación?
- ✓ ¿Como obtenerlo?
- ✓ Identificar las propiedades a obtener.

Este componente proveerá los siguientes tipos de información:

6.7.1. Control de Cumplimiento de Objetivos

Este punto reflejará el seguimiento del porcentaje de cumplimiento de los distintos objetivos definidos por el usuario para los distintos tipos de agentes.

Se tendrá por cada objetivo definido por el usuario una línea de tiempo que refleje su evolución.

De esta manera se podrá identificar que objetivos se cumplieron con mayor grado durante la simulación o en determinados períodos de tiempo.

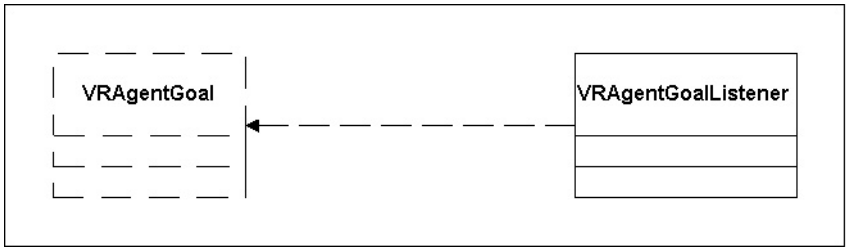

Diagrama 11 – Diseño del manejo de objetivos

VRAgentGoalListener: Encargada de registrar el cumplimiento de los objetivos a través del tiempo. Funciona como un listener o dependiente.

6.7.2. Deducción de patrones dinámicos de comportamiento

Patrones dinámicos de comportamiento hace referencia a relaciones entre agentes surgidas en la simulación, las cuales no fueron definidas en tiempo de diseño.

Son propiedades surgidas por la configuración que realizó el usuario más la ejecución de la simulación en esas condiciones.

Las técnicas que se utilizarán para detectar estas propiedades emergentes serán:

- Objetos intermedios (abstracción de propiedades comunes).
- Clasificación de tipos de interacciones entre agentes.

6.7.2.1. Objetos Intermedios (medium agents)

La técnica consiste básicamente en identificar aquellos agentes con estados o características similares y crear agentes de nivel intermedio que representen esta situación. Podríamos decir que cuando este hecho ocurre, estamos ante la presencia de un conjunto de agentes auto-organizados.
En nuestro caso, la manera de determinar esa similitud entre estados de agentes, consistirá en la comparación en el grado de cumplimiento de los objetivos similares.

El proceso de identificación de estados similares entre dos agentes es el siguiente:

- ✓ Dado el agente 1 con su lista de objetivos (P1, P2, ..., Pn) y el agente 2 con su lista de objetivos (Q1, Q2, ..., Qn).

- ✓ Identificar los objetivos similares del agente 1 y del agente 2. Dos objetivos son similares si: son del mismo tipo o si satisface la función de comparación dada por el usuario. Para esto, el Framework define la interface VRAgentGoalComparator para ser implementada por el usuario.

- ✓ Una vez definidos los pares (Pi, Qi) de objetivos similares se compara el porcentaje de cumplimiento de los mismos $| \%Pi - \%Qi | < \varepsilon i$. εi es el error permitido para el par de objetivos i.

- ✓ Si los agentes poseen una cantidad X de objetivos similares entonces se reconocen como similares y se crea un agente intermedio (medium agent) que refleja esta relación.

Una vez detectada la similitud entre un conjunto de agentes, se creará un VRMediumAgent, que tendrá las siguientes características:

- ✓ Es un VRAgent más dentro de la simulación.

- ✓ Poseerá los objetivos que fueron similares de los agentes agregados.

- ✓ Contendrá las tareas necesarias para cumplimentar los objetivos comunes.

- ✓ Cualquier otra característica que posee un VRAgent estará presente en un VRMediumAgent como promedio de las características de los agentes agregados.

- ✓ Conocerá los agentes que dieron su origen (agentes agregados)
- ✓ Mantendrá la lista de εi que le dieron origen (lista de restricciones que se tendrán que seguir cumpliendo para la existencia de este VRMediumAgent).

El objetivo de un VRMediumAgent es el de modelar niveles de agregación y registrar el comportamiento emergente surgido en la simulación.

Diseño del componente

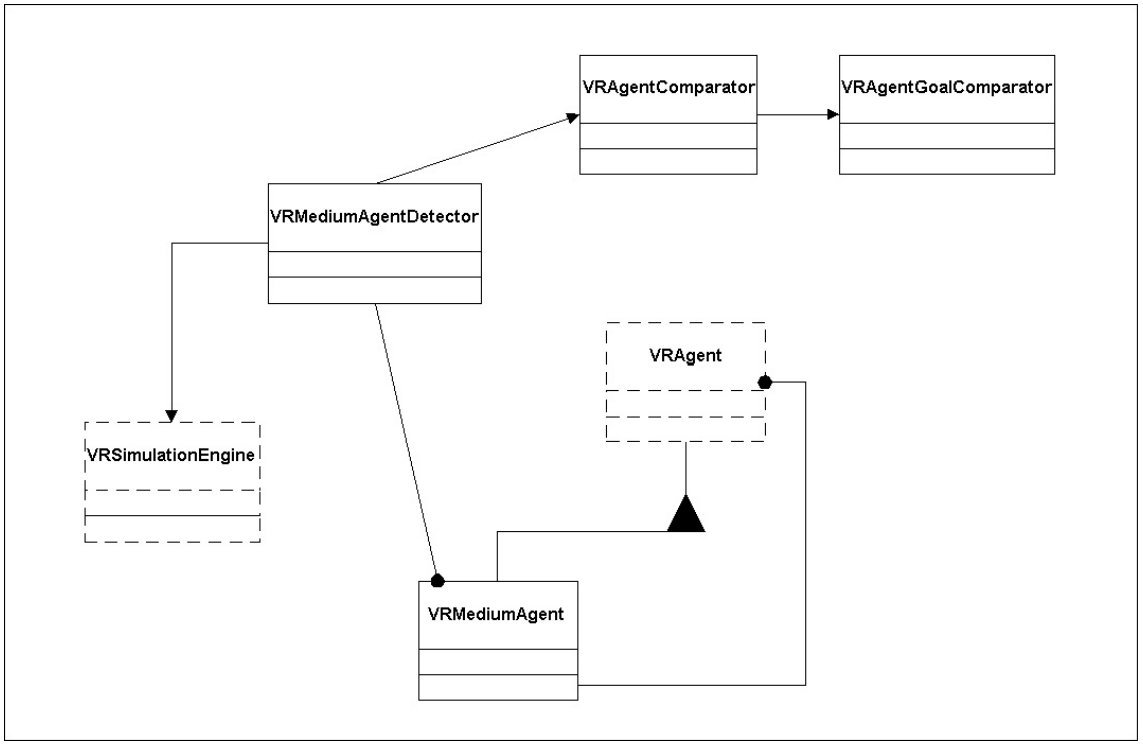

Diagrama 12 – Deducción de información a través de MediumAgents

VRAgentGoalComparator: Es una interface que provee el Framework que deberá ser implementada para comparar objetivos. Retornará verdadero en caso de similitud.

VRAgentComparator: Aplica el algoritmo de comparación de estados de Agentes. Conoce al VRAgentGoalComparator y contiene además otras variables de configuración del algoritmo, como por ejemplo: cantidad de objetivos similares que determinan la similitud entre agentes, lista de coeficientes de errores permitidos entre objetivos.

VRMediumAgentDetector: Responsable de detectar similitudes entre agentes de la simulación y crear los VRMediumAgent. Cumple con el objetivo del componente en el sentido de crear niveles de organización y detectar el comportamiento emergente.

VRMediumAgent: Resultado de la agregación de agentes similares. Representa la auto-organización de agentes. Se incluye como un agente mas dentro de la simulación, lo que permite la creación de diferentes niveles de organización.

Nota:

VRMediumAgent corresponde a la instanciación del *pattern* ***Composite,*** que representa la agregación de otros objetos más simples. Modela niveles de organización, comportamiento emergente.

6.7.2.2. Clasificación de tipos de interacciones entre agentes

Existen tres elementos principales en la interacción: los objetivos, el acceso a los recursos y las habilidades de los agentes (tareas que posee el agente para cumplir sus objetivos) pueden ser considerados para crear una tipología de las situaciones de interacción.

Compatibilidad de Objetivos

El objetivo de un agente A (O_A) **es incompatible** con el objetivo de un agente B (O_B) , si el cumplimiento de O_A origina el incumplimiento de O_B.

Habilidades del agente

Nos referimos a "habilidad del agente" con respecto a un objetivo X cuando posee las tareas necesarias para cumplir dicho objetivo.

La siguiente tabla resume nuestra clasificación de los tipos de interacciones entre agentes:

Objetivos	Recursos	Habilidades	Tipo de Interacción	Observaciones
Compatible	Suficientes	Suficientes	**Independencia**	No posee problemas desde el punto de vista multi-agentes
Compatible	Suficientes	Insuficientes	**Colaboración Simple**	Consiste en el agregado de habilidades. No requiere coordinación suplementaria.
Compatible	Insuficientes	Suficientes	**Obstrucción**	Situaciones en las que un agente impide la finalización de las tareas de otro.
Compatible	Insuficientes	Insuficientes	**Colaboración Coordinada**	Colaboración Compleja. Los agentes deben coordinar sus acciones.
Incompatible	Suficientes	Suficientes	**Competición individual pura**	Los agentes deben negociar para alcanzar sus objetivos.
Incompatible	Suficientes	Insuficientes	**Competición colectiva pura**	Ante habilidades insuficientes los agentes deben asociarse para cumplir sus objetivos.
Incompatible	Insuficientes	Suficientes	**Conflictos individuales sobre recursos**	Los recursos no pueden ser compartidos.
Incompatible	Insuficientes	Insuficientes	**Conflictos colectivos sobre recursos**	Se forman asociaciones para obtener monopolios de recursos.

Tabla 4 – Tipos de interacciones entre agentes

Diseño del componente

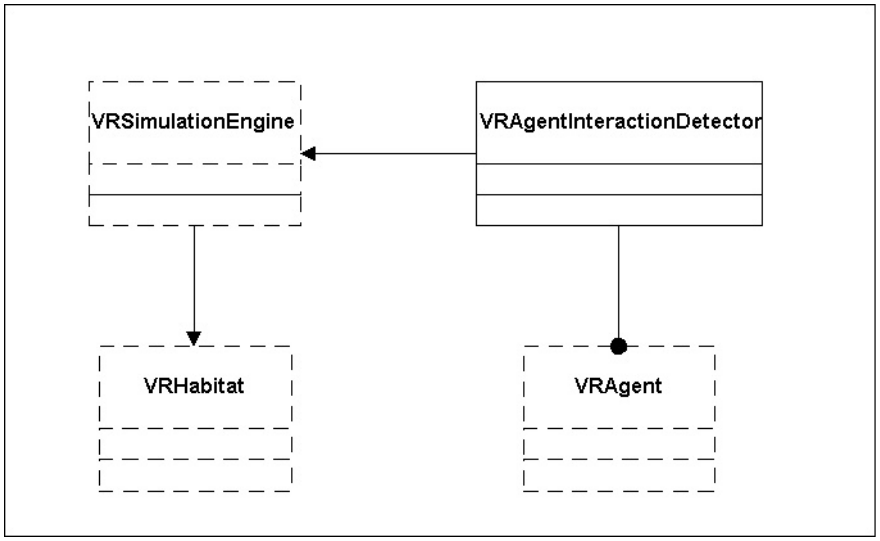

Diagrama 13 – Deducción de información a través de tipos de Interacciones entre Agentes

VRAgentInteractionDetector: Se encarga de detectar el tipo de interacción entre dos o más agentes. Evalúa la compatibilidad de objetivos, la suficiencia de recursos y las habilidades de los agentes intervinientes en la detección.

Conclusiones

Las simulaciones basadas en mundos virtuales son modelos extremadamente simplificados de la complejidad del mundo real. La intención de estos mundos artificiales es lograr una abstracción del comportamiento del mundo real en situaciones específicas a fin de volcar estos datos en una computadora.

Sin duda alguna, esta forma de afrontar el estudio de los sistemas complejos, está, en la actualidad, adquiriendo cierta relevancia. La fama que se ha ganado en los últimos años de experimento curioso, donde podía verse a unos bichitos moverse sin orden y concierto por la pantalla, parece que se está dejando a un lado. Sin duda, su origen fue ese, y de ahí su nombre y su uso inicial para poder reproducir en una computadora sistemas vivos que existían en la naturaleza. Y sin duda, se seguirá utilizando para ese fin. Sin embargo, cada vez más investigadores adaptan los mecanismos evolutivos de la Vida Artificial para el análisis y predicción de otro tipo de sistemas que no sean los tradicionalmente biológicos.

Si bien nuestro trabajo se enmarcó dentro del campo de los agentes biológicos, los límites en la aplicación de las técnicas de Vida Artificial no se quedan en la frontera de lo estrictamente biológico, sino que son aplicables a muchos otros campos.

Nos parece que la programación orientada a objetos es uno de los mejores soportes (plataformas) para desarrollar aplicaciones o sistemas dentro del campo de la Vida Artificial, pues el concebir todo como un objeto encaja perfecto con muchos de los conceptos de esta rama de la informática, es decir que un objeto se podría ver como un "ser" artificial que posee comportamiento y estado. Por lo tanto tener una plataforma con estas características hace más sencilla la tarea de definir estos sistemas, de manera rápida y con una base sólida.

Puntos resueltos de proyectos anteriores

En el prólogo del trabajo mencionamos nuestro proyecto **VRColonies** como motivador de nuestra tesis. En las conclusiones de dicho trabajo expresábamos los siguientes párrafos como trabajos futuros:

"Extender el sistema para detectar de alguna forma Patrones de Comportamiento existentes en este tipo de ambientes. No sólo encontrar estos Patterns sino buscar

la manera de que puedan formar parte de un Framework dedicado al dominio de Mundos Virtuales."

Creemos haber resuelto este punto al haber diseñado el Framework y proveer la herramienta que permite detectar las relaciones entre agentes y clasificarlas. *(ver Capítulo 6, Sección 6.7.2.2)*

Mencionábamos también:
"Capturar datos relacionados con el comportamiento emergente o del grupo de criaturas de la misma especie para dar una idea más precisa al usuario de cómo evolucionan las especies que definió."

Este punto lo resolvimos con la captura de propiedades comunes en un grupo de agentes. *(ver Capítulo 6, Sección 6.7.2.1)*

Nuevos patrones encontrados

Patrón	Descripción	Instanciación
Ejecución de eventos periódicos	Objetos que ejecutan su comportamiento en forma periódica.	VRPeriodicEvent
Modelización del comportamiento a través de unidades competitivas	Modelización del comportamiento como competición de unidades mas simples. Conjunto de tareas que compiten para definir quien se ejecuta en cada momento.	Modelo de Tareas competitivas / Objetivos. VRAgent, VRAgentGoal, VRGoalEvaluator, VRAgentTask, VRTaskEvaluator
Señales / Sensores	Transformación de objetos (señales) de un formato determinado a otro formato unificado y entendible dentro de un determinado sistema (VRAgent).	Señales Externas / Sensores / Señal interna. VRAgent, VRAgentSensor, VRAgentSensorHandler, VRSignal, VRUSignal.
Modelo de layers como definición de un objeto más complejo	Modelar Objetos como conjunto de unidades más simples. Cada unidad representa un aspecto del objeto general.	VRHabitat, VRHabitatLayer, VRResourceLayer, VRSignalLayer
Propagación de señales	Objetos que cambian su estado en forma continua a través del tiempo. Inciden en el estado de otros objetos a medida que cambian su estado.	VRAgentSensor, VRSignal, VRSignalLayer, VRResource, VRResourceLayer

Objetivos alcanzados

Expresamos a continuación nuestra resolución a los objetivos planteados en la propuesta del trabajo:

- Crear un Framework que soporte aquellas aplicaciones que se desarrollen dentro del dominio de los mundos virtuales.

A medida que avanzábamos en el desarrollo del trabajo, percibimos que este objetivo era muy general así planteado, debido a la amplitud del campo de los mundos virtuales. Decidimos entonces enfocar nuestra investigación a los mundos virtuales de agentes reactivos biológicos.

- Dotar al Framework con la infraestructura necesaria para soportar la ejecución de simulaciones que corran sobre el mismo. Esto incluye: manejo de eventos que ocurren en la simulación, manejo del factor tiempo, broadcast de los estímulos que ocurran dentro del ambiente hacia las criaturas, administración de procesos de ejecución continua, etc.

- Incluir en el Framework la capacidad de deducir información surgida de las simulaciones (Patrones dinámicos de comportamiento).

Se plantea una resolución a estos puntos en el *capítulo 6*.

- Capturar y desarrollar aquellos patrones que aprehendan los conceptos generales que se encuentran en estos tipos de dominios y plasmarlos en el Framework. De acuerdo a lo estudiado y visto en estos mundos virtuales, se intentará descubrir los problemas recurrentes que aparecen en los mismos y expresarlos en forma de patrones.

Extensibilidad del Framework

Las utilidades de este tipo de sistemas se relaciona directamente con el estudio del comportamiento de distintas "entidades" que conviven en un medio cambiante a través del tiempo. Se pueden mencionar:

- ✓ Urbanismo: modelización del tráfico.

- ✓ Arquitectura: Estudio del comportamiento de las personas en cuanto a elección de ambientes o lugares se refiere, ejemplo: observar el movimiento del grupo de personas ante las distintas condiciones impuestas por el medio ambiente, que sectores prefieren, etc...

- ✓ Medicina: Comportamiento de grupos de virus de una misma clase, epidemias, dispersión de una plaga, etc...

- ✓ Sociología: modelos de rebelión, de creación y dispersión de opiniones.

- ✓ Economía: modelos de intercambio de bolsa, de establecimiento y evolución de precios.

Trabajos Futuros

Implementación

Por la concepción del Framework de haber sido "pensado" en el paradigma de objetos, pensamos que la plataforma JAVA es la adecuada para la implementación de su funcionalidad.

Instanciación

Generar aplicaciones que utilicen el Framework y puedan concretarse y apreciarse los beneficios del mismo.

Extensión a otros campos

Análisis de la extensión del Framework para otras áreas: economía, sociología, medicina, arquitectura, urbanismo, etc.

Bibliografía

[1] - Capítulo 2: Los Reproductores – El Gen Egoísta de Richard Dawkins

[2] - GAIA – sitio Internet

[3] - Capítulo 3: Las espirales inmortales – El Gen Egoísta de Richard Dawkins

[4] - Pagina Vida Artificial

[5] - ROCH97

[6] - Vida Artificial por Miguel José Hernández y Javier Serrano_Trabajo de IA (Vida Artif)

[7] - Langton 89

[8] - Glosario de Carlos von der Becke

[9]- http://members.nbci.com/_XMCM/jeusamio/vida_artif.html

[10] – Multi-Agent Systems – An introduction to distributed Artificial Intelligence. Jacques Ferber.

[11] - Durfee, E.H., Lesser, V.R. and Corkill, D.D. Trends in Cooperative Distributed Problem Solving. In: IEEE Transactions on Knowledge and Data Engineering, March 1989, KDE-1(1), pages 63-83.

[12] - Jennings, N.R., Sycara, K. and Wooldridge, M. A Roadmap of Agent Research and Development. In: Autonomous Agents and Multi-Agent Systems Journal, N.R. Jennings, K. Sycara and M. Georgeff (Eds.), Kluwer Academic Publishers, Boston, 1998, Volume 1, Issue 1, pages 7-38

[13] - Jennings, N.R., Sycara, K. and Wooldridge, M. A Roadmap of Agent Research and Development. In: Autonomous Agents and Multi-Agent Systems Journal, N.R. Jennings, K. Sycara and M. Georgeff (Eds.), Kluwer Academic Publishers, Boston, 1998, Volume 1, Issue 1, pages 7-38.].

[14] - Decker, K., Sycara, K. and Williamson, M. Middle-Agents for the Internet. In: Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI-97), January, 1997

[15] - Nwana, H.S. Software Agents: An Overview. In: The Knowledge Engineering Review, October/November 1996, Volume 11, Number 3, pages 205-244.

[16] - Finin, T., Labrou, Y. and Mayfield, J. KQML as an Agent Communication Language. In: Software Agents, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 291-316.

[17] - Stuart A. Kauffman, Antichaos and daptation, Sci. Am., nov.1991, p. 64

[18] - Pensando en niveles.Un enfoque de los sistemas dinámicos para entender el mundo - Uri Wilensky -Mitchel Resnick

[19] - Minsky, M., The Society of Mind, Nueva York: Simon & Schuster, 1987.

[20] - Hofstadter, D., Godel, Escher, Bach: An Eternal Golden Braid, Nueva York: Basic Books, 1979.]

[21] - Gell-Mann M - Complex Adaptive Systems, en Morowitz H, Singer JL - The Mind, the Brain and Complex Adaptive Systems - Addison-Wesley, 1994

[22] - Chaitin, 1975; Rabin, 1977

[23] – Kauffman

[24] – HUBE94

[25] - Gleick 1989

[26] - Virtual Worlds

[27] - Gamma 94