Ingeniare. Revista Chilena de Ingeniería
ISSN: 0718-3291
facing@uta.cl
Universidad de Tarapacá
Chile

Castillo O., Luis F.; González B., Manuel; Isaza E., Gustavo; Vélez, Jairo I.
Hacia las arquitecturas cognitivas conscientes: aplicación en el dominio de los videojuegos
Ingeniare. Revista Chilena de Ingeniería, vol. 23, núm. 4, octubre, 2015, pp. 514-525
Universidad de Tarapacá
Arica, Chile

Hacia las arquitecturas cognitivas conscientes: aplicación en el dominio de los videojuegos

Towards conscious cognitive architectures: application in the video games domain

Luis F. Castillo O.[1,4] Manuel González B.[2] Gustavo Isaza E.[1] Jairo I. Vélez[3]

Recibido 17 de abril de 2014, aceptado 23 de diciembre de 2014
Received: April 17, 2014 Accepted: December 23, 2014

RESUMEN

Las arquitecturas cognitivas, desde su origen, han buscado modelar el comportamiento y estructura de agentes inteligentes considerados como modelos integrados, es decir, constituyen enfoques donde se articulan procesos de razonamiento, planificación, aprendizaje, etcétera, de un modo unificado.
El área de las arquitecturas cognitivas y su aplicación en videojuegos se ha desarrollado intensamente en los últimos años. Puede constatarse que la mayor parte de los trabajos se han realizado *ad hoc* sin un procedimiento sistemático para su desarrollo concreto, ni con vistas a plantear futuras evoluciones.
Excepcionalmente, algunos autores han propuesto criterios y escalas de complejidad para estructurar de un modo más formal y sistemático líneas de avance.
En este artículo se tomaron como referencia dos de las escalas más relevantes en la caracterización de arquitecturas cognitivas: donde se evalúan los grados de conciencia artificial de arquitecturas cognitivas. Utilizando como base la arquitectura CERA-CRANIUM y se ha propuesto implementar los mecanismos necesarios para desarrollar las extensiones que le permitan alcanzar niveles más altos en las dos escalas consideradas.
Entre las nuevas implementaciones realizadas se destacan la integración de un sistema de toma de decisiones para controlar el funcionamiento del *bot* basado en SOAR que reproduce mecanismos de cambio de estrategia ante el reconocimiento de modificaciones en el entorno, aprendiendo de su propia experiencia mediante "aprendizaje por refuerzo" así como una mejor simulación del comportamiento humano gracias a la implementación de algunos comportamientos de inspiración emocional. Se finaliza el trabajo proponiendo un método para aumentar progresivamente las capacidades asociadas a arquitecturas cognitivas.

Palabras clave: Arquitecturas cognitivas, CERA-CRANIUM, SOAR, comportamiento artificial, agentes inteligentes

ABSTRACT

From the beginning, cognitive architectures, have sought to model the behavior and structure of intelligent agents considered integrated models, i.e., they are approaches where processes of reasoning , planning, learning, etc. are articulated in a unified way .
The area of cognitive architectures and its application in video games has been x intensively from the beginning in recent years. It is evident that most of the work has been done ad hoc without a systematic

[1] Departamento de Sistemas e Informática. Grupo Investigación GITIR/Ci²Dt². Universidad de Caldas. Sede Principal Calle 65 # 26-10. Manizales, Colombia. E-mail: gustavo.isaza@ucaldas.edu.co; luis.castillo@ucaldas.edu.co
[2] Departamento de Ciencias de la Computación. Universidad de Zaragoza, Calle Pedro Cerbuna, 12, 50009. Zaragoza, España. E-mail: mgbedia@unizar.es
[3] Grupo de Investigación Ingeniería del Software. Universidad Autónoma de Manizales. Antigua Estación del Ferrocarril. Manizales, Colombia. E-mail: jvelez@autonoma.edu.co
[4] Departamento de Ingeniería Industrial. Universidad Nacional de Colombia, Sede La Nubia. Manizales, Colombia. E-mail: lfcastilloos@unal.edu.co

process for specific development or in order to raise future developments. Exceptionally, some authors have proposed criteria and scales of complexity to structure a more formal and systematic way forward lines. In this article, were taken as reference two of the most important scales in the characterization of cognitive architectures that assesses the degree of artificial consciousness cognitive architectures.
Using as a basis the CERA- CRANIUM architecture and it is proposed to implement the necessary mechanisms to develop extensions that allow reaching higher levels in the two considered scales.
Among the new implementations performed highlights the integration of a decision-making system for controlling the operation of the bot based on SOAR who played strategy change mechanisms to recognize changes in the environment, learning from their own experience by "reinforcement learning" as well as a better simulation of human behavior through the implementation of some emotional behaviors inspiration. Work by proposing a method to progressively increase skills related to cognitive architectures is finished.

Keywords: Cognitive architectures, CERA-CRANIUM, SOAR, artificial behavior, intelligent agents.

INTRODUCCIÓN

Se puede definir una arquitectura cognitiva [1] como el esquema o patrón para estructurar los elementos funcionales que configuran a un agente inteligente.

Las arquitecturas cognitivas proponen procesos computacionales que actúan como ciertos sistemas cognitivos (como podrían ser un "ser humano") o, más a menudo, actos inteligentes bajo determinada definición. El término "arquitectura" implica que se intenta modelar no solo el comportamiento, sino también las propiedades estructurales del sistema analizado.

Las arquitecturas cognitivas existentes pueden clasificarse en dos grupos principales. En primer lugar, la aproximación cognitivista, basada en información simbólica procesada por sistemas representacionales [4]. Las aproximaciones cognitivistas se corresponden con la clásica visión de que "la cognición es un tipo de computación simbólica, racional, encapsulada, estructurada y algorítmica" y de que un sistema cognitivo "instancia estas representaciones físicas como códigos cognitivos y sus comportamientos son una consecuencia causal de las operaciones que acarrean estos códigos" [5].

Por otro lado están los sistemas conexionistas, dinámicos y activos (todos ellos basados en una mayor o menor extensión de los principios de autoorganización [5]) agrupados bajo el nombre general de sistemas emergentes. Estos sistemas discuten contra la visión del procesado simbólico y se posicionan a favor de tratar la cognición como emergente, autoorganizada y dinámica [14]. Existe además un tercer grupo de arquitecturas denominadas híbridas que combinan las características de ambos paradigmas.

Tomemos algunas de las más importantes arquitecturas cognitivas. Se puede obtener un vistazo general de sus propiedades si se hace una comparación en función de 12 (doce) características distintas: operatividad computacional, infraestructura de la representación, grounding semántico, restricciones temporales, epistemología global, corporeidad, percepción, acciones, anticipación, adaptación, motivación y relevancia de la autonomía. En la Tabla 1 se han omitido las cinco primeras características porque se pueden inferir directamente desde el propio paradigma en el que se basa el sistema: cognitivista (C), emergente (E) o híbrido (H). En la Tabla 1 se señala mediante una "×" que esta característica es clave en la arquitectura, "+" indica que existe esa característica y un espacio en blanco indica que esta característica no está presente en la arquitectura. Se asigna una "×" en la columna de adaptación si y solo si el sistema es capaz de desarrollar, para el caso, crear nuevos marcos o modelos de representación) y no simplemente aprender, que permita estimar parámetros de un modelo. Estos atributos son los que los autores Vernon, Meta y Sandini en [11] consideran que deben cumplirse por una arquitectura cognitiva.

De acuerdo con lo anterior, los sistemas cognitivos artificiales estarían constituidos por una red de subsistemas multifuncionales distribuidos que cooperan y compiten, cada uno de ellos posee su propia infraestructura de codificación o

Tabla 1. Comparativa de arquitecturas.

Arquitectura	Paradigma	Embodiment	Percepción	Acción	Anticipación	Adaptación	Motivación	Autonomía
Soar	C				+	+		
Epic	C		+	+	+			
ACT-R	C		+	+	+	+		
ICARUS	C		+	+	+	+		
Adapt	C	X	X	X	+	+		
AAR	E	X	X	X			+	X
Global Workspace	E	+	+	+	X		X	X
I-C SDAL	E	+	+	+	+	+	X	X
SASE	E	X	X	X	+	X	X	X
Darwin	E	X	X	+		X	X	X
Humanoid	H	X	X	X	X	+	+	
Cerebus	H	X	X	X	+	+		
Cog	H	X	X	X	+			
Kismet	H	X	X	X			X	

Fuente: Extraída de [15].

representación, que alcanzan juntos un objetivo común de comportamiento efectivo controlados por algún tipo de mecanismo de autosincronización o circuito de modulación. Esta red formaría la configuración filogenética y sus habilidades innatas.

Como se observa en la Tabla 1, que recoge información de las arquitecturas cognitivas más importantes, no existe ninguna que cumpla todos los requisitos estudiados.

Vernon, Meta y Sandini en [10-11] proponen que el siguiente paso en el desarrollo del área de las arquitecturas artificiales sea el de completar los atributos que faltan en las arquitecturas existentes para que cumplan con más rigor un criterio estándar de arquitectura cognitiva.

En este artículo se desarrollará esta tarea, tomando como punto de partida la arquitectura CERA-CRANIUM, una arquitectura cognitiva inspirada en la Teoría del Espacio de Trabajo Global (Global Workspace) [12], ver Figura 1, donde un tipo de espacio de memoria compartida donde diferentes agentes –en este caso representados por procesadores especializados– pueden colaborar y competir con

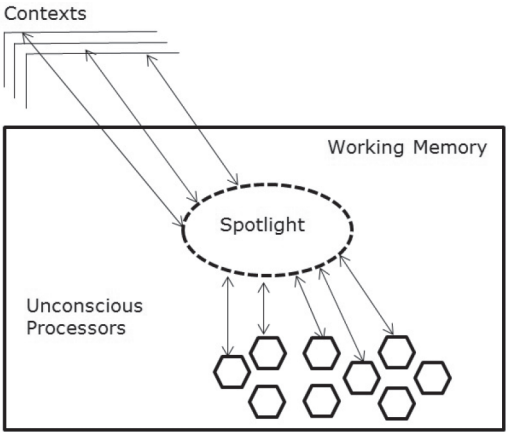

Figura 1. Modelo del Espacio de Trabajo Global. Fuente: Extraído de [12].

otros dinámicamente. Explicaremos con más detalle en la siguiente sección su funcionamiento y estructura.

MODELO DE INSPIRACIÓN

La *arquitectura CERA-CRANIUM* está compuesta por dos componentes principales (ver Figura 2):

CERA: Arquitectura de control estructurada en capas.

CRANIUM: herramienta para la creación y gestión de grandes cantidades de procesos en espacios de trabajo compartidos.

Como se explica más abajo, CERA utiliza los servicios de CRANIUM con el objetivo de generar procesos de percepción dinámicos y adaptables orquestados por un modelo computacional de la consciencia [8]. En términos de control del bot, CRANIUM proporciona un mecanismo para sincronizar un número de procesadores especializados diferentes que funcionan concurrentemente. Estos

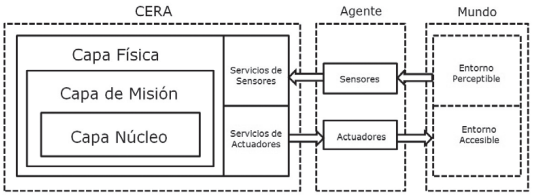

Figura 2. Diseño estructurado en capas de la arquitectura cognitiva CERA.
Fuente: Extraído de [12].

procesadores pueden ser de muchos tipos, aunque normalmente son detectores de información sensorial o generadores de comportamiento.

Siguiendo la arquitectura de orientación a servicios de MRDS (Microsoft Robotics Developer Studio), los componentes de CERA y CRANIUM se integran como servicios ligeros distribuidos. MRDS incorpora un sistema de ejecución de tiempo real llamado CCR (Concurrency and Coordination Runtime) que está basado en .NET Framework. Partes de CERA y de CRANIUM se basan en el CCR para soportar patrones de coordinación complejos y asíncronos. También se usan las colas de hilos de alto rendimiento para ejecutar un gran número de procesadores inconscientes y maximizar la concurrencia. El protocolo DSSP (Decentralized Software Services Protocol) hace posible la composición de estos servicios. El modelo de servicios DSS permite la interacción de los servicios en un entorno descentralizado, por tanto, el problema de la capacidad limitada de CPU en el robot desaparece.

CERA (Conscious and Emotional Reasoning Architecture) es una arquitectura cognitiva distribuida en capas, diseñada para implementar un sistema de control para agentes autónomos [10]. La actual definición de CERA está estructurada en cuatro capas (Figura 2): la capa de servicios sensomotores [13], capa física, capa específica de misión y capa núcleo. Aunque las capas superiores tienen asignados significados más abstractos, la definición de las capas en CERA no está directamente asociada a comportamientos específicos. En su lugar, gestiona cualquier procesador especializado que opere en el tipo de representación utilizado en un nivel particular, como por ejemplo la capa física con representaciones cercanas a los datos sensoriales o la capa misión con representaciones de más alto nivel y orientadas a tareas concretas.

CRANIUM proporciona un subsistema en el que CERA puede ejecutar varios procesos concurrentes coordinados pero asíncronos que compiten entre ellos. Cada uno de estos procesadores especializados está diseñado para desarrollar una función específica con tipos de información concretos. En cualquier momento, el nivel de activación de un procesador concreto se calcula basado en una estimación heurística de cuánto puede contribuir a la solución global que actualmente se encuentra en el ETC. Los parámetros concretos usados para esta estimación se establecen en la capa de núcleo CERA. Por norma general, los ETC de CRANIUM operan constantemente ajustados por comandos enviados desde la capa núcleo de CERA. Este mecanismo cierra los bucles de realimentación entre la capa núcleo y el resto de la arquitectura: la entrada de la capa núcleo (percepción) se regula en función de su propia salida (modulación del ETC), que al volver determina qué se percibe.

CC-Bot2 es una implementación Java de la arquitectura CERA-CRANIUM especialmente desarrollada para la competición 2K BotPrize. Esta implementación tiene las siguientes características:

- La capa CERA de servicios sensoriomotores es básicamente una capa de adaptación a Pogamut 3.
- La representación usada para los datos sensoriales y los comandos en la capa física, como podrían ser "aparece un jugador en mi campo de visión" o "estoy siendo atacado", es Pogamut 3.

Evaluación de CERA-CRANIUM como arquitectura cognitiva

Puesto que la arquitectura CERA-CRANIUM está basada en la Teoría del Espacio Global de Trabajo de Baars y no presenta diferencias significativas de funcionamiento respecto de la arquitectura Global Workspace de Shanaman [12], se asume que las características de CERA-CRANIUM se corresponden con las de Global Workspace a efectos del análisis de la tabla anterior. Como se puede observar, la única característica que no aparece referenciada en CERA-CRANIUM es la de adaptación, por lo que se requiere de manera casi obligatoria la integración de algún mecanismo adaptativo. Es importante que esta mejora se realice en el conjunto global de la arquitectura y no en una implementación concreta para que la posterior evolución de estas implementaciones no se vea sujeta a las limitaciones de dicha implementación y la explotación de CERA-CRANIUM aplicada a distintos ámbitos se pueda realizar de manera transparente.

Evaluación de *conscale* como arquitectura consciente

En la Figura 3 se presentan los resultados obtenidos al aplicar el Proceso de Evaluación Simplificado

(PSE) sobre la implementación CC-Bot2. El cuadro contiene en su primera columna la lista de habilidades que abarca este modelo y más abajo se indica el valor del CQS, en la segunda columna se representan los perfiles cognitivos del modelo y finalmente, en la tercera columna, se muestra el nivel conceptual de ConsScale alcanzado.

Es conveniente resaltar que el PSE simplemente proporciona una aproximación de lo que podría ser el nivel real ConsScale de una implementación. CC-Bot2 cumple con algunas características de los niveles 3, 4 y 5. Sin embargo, se clasifica como un agente de nivel 2 porque ConScale requiere el cumplimiento completo de los niveles inferiores para poder ser calificado como perteneciente a un determinado nivel i. El índice CQS de un agente reactivo puro es 0.18. Sin embargo, la puntuación de CC-Bot (0.51) indica que el agente presenta capacidades cognitivas adicionales. Asimismo, CC-Bot está lejos de alcanzar el nivel 4, para el cual el CQS asociado sería de 12.21 o superior.

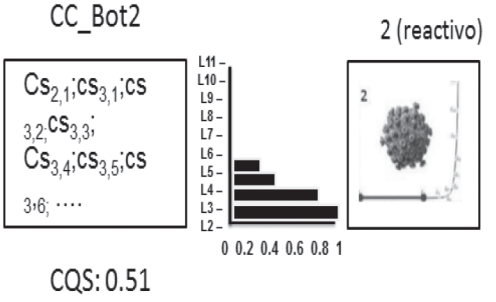

Figura 3. Evaluación PSE del CCBot2.

En la Tabla anterior (Tabla 2) se enumera la lista de habilidades cognitivas ya implementadas en el modelo CCBot-2. El objetivo es, de manera casi inmediata, implementar la habilidad cognitiva restante para poder completar el nivel 3 y, una vez superada esta barrera, analizar los diferentes requisitos de los niveles 4, 5 y (en menor medida) 6 para implementar algunos de ellos y conseguir el mayor avance posible en la escala.

Es esencial que la arquitectura CERA-CRANIUM posea la característica de adaptación. La opción más viable para implementar esta opción es crear una arquitectura híbrida basada en CERA-CRANIUM e integrar un módulo de comportamiento cognitivista. De esta manera se mantienen todas las propiedades de la arquitectura base y la propia estructura de CERA-CRANIUM nos permite realizar una integración casi modular de esta nueva característica.

Tabla 2. Lista de habilidades cognitivas implementadas en el modelo CCBot-2.

CS	Requisito
2,1	Capacidad de producir respuestas motoras fijas o reactivas (reflejos).
3,1	Adquisición autónoma y adaptativa de nuevas respuestas reactivas.
3,2	Uso de los sensores propioceptivos para generar respuestas adaptativas.
3,3	Selección de información sensorial relevante.
3,4	Selección de información motora relevante.
3,5	Selección de información relevante en memoria.
3,6	Evaluación (positiva o negativa) de objetos o sucesos seleccionados.
4,1	Aprendizaje por prueba y error. Reevaluación de los sucesos seleccionados.
4,5	Representación espacial relativa (depictive) de los objetos percibidos.
5,2	Persecución de múltiples metas.
5,4	Aprendizaje por refuerzo autónomo.

La integración de esta característica en el sistema facilitará además el desarrollo de algunos de los requisitos de los niveles 4, 5 y 6 de ConScale para la nueva implementación.

MODELO DE TRABAJO

Arquitectura CCBOT-SOAR

En esta sección se detallan las mejoras realizadas en la arquitectura para desarrollar la característica de adaptación (el Sistema de Toma de Decisiones) y las nuevas implementaciones complementarias como son la Memoria de Objetos y la simulación de emoción “Riesgo”.

De entre las arquitecturas cognitivas analizadas que presentan la característica de adaptación, SOAR [2-3] ha sido la seleccionada, por ser además la más sencilla de integrar en un sistema híbrido. Como se va a utilizar Pogamut para la implementación del bot es necesario que la arquitectura seleccionada disponga de una versión en Java para facilitar así la conexión entre Pogamut y la arquitectura (en este caso, jSoar1). Como el bot está centrado en la

capacidad de aprender de sus propias experiencias SOAR es la arquitectura seleccionada para el desarrollo del sistema de toma de decisiones del bot.

Implementación de memoria con SOAR

Durante el transcurso de una partida de Unreal Tournament 2004 (y en la de casi cualquier otro juego FPS) la posición de los objetos (munición, salud y otros objetos especiales) será siempre la misma. Los jugadores habitualmente recuerdan la posición de estos objetos ya sea porque estos se encuentran en puntos estratégicos muy fáciles de identificar (y más cuando la mayoría de los mapas tienen cierta simetría) o porque por el propio desarrollo de la partida consiguen acordarse. Implementar este comportamiento en el bot mejorado es tan sencillo como crear un nuevo procesador de perceptos.

Este nuevo procesador (bautizado como RememberItems en la actual implementación) capturar los perceptos de "objeto recogido" y almacena sus posiciones en dos listas diferentes: una para todos los objetos que incrementan o recuperan salud, así como los distintos tipos de escudos de energía y otra para el resto de objetos (principalmente municiones y armas, aunque también se incluyen potenciadores de disparo).

El procesador gestiona y actualiza estas listas con cada nuevo objeto recogido almacenando la posición del objeto aunque no su tipo, pues en algunos casos el juego crea puntos en los que los objetos que se pueden recoger, mutan a lo largo del tiempo y tampoco sería realista que un jugador recordase exactamente el tipo de objeto concreto que aparece en un punto.

El mismo procesador puede capturar perceptos sobre el nivel de salud y munición, de manera que cuando estos niveles se encuentran por debajo de un mínimo (definidos globalmente y utilizados por el procesador que ayuda a huir de los enemigos) propondrá al sistema una acción de movimiento dirigida hacia el objeto más cercano que se corresponda con su necesidad (salud o munición).

En la Figura 4 se representa gráficamente el funcionamiento interno de este procesador.

Sistema de toma de decisiones

Como extensión a la nueva característica de adaptación y como parte de la creación de una

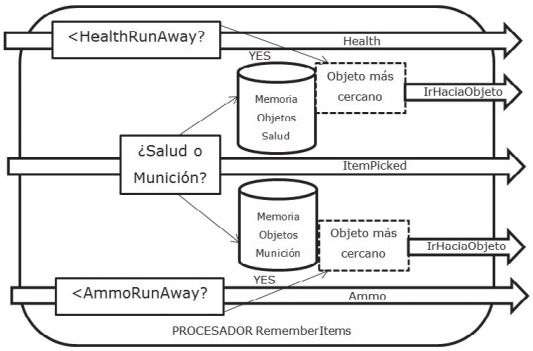

Figura 4. Procesador RememberItems.

memoria a largo plazo se ha implementado un sistema de toma de decisiones centrado en la capacidad de aprender de sus propias experiencias e implementando "emociones" en su comportamiento. En particular, el sistema ha sido desarrollado con base en los conceptos de motivación, drive, emoción y aprendizaje por refuerzo [6]. Los conceptos de motivación y drive forman parte de la representación del estado bot, más concretamente del estado interno del bot.

Aprendizaje por refuerzo

La valoración de los comportamientos cuando el bot está en un cierto estado, se realiza utilizando algoritmos de aprendizaje por refuerzo. El bot aprenderá qué acción escoger cuando se encuentra en un estado determinado.

A pesar de haberse convertido en uno de los algoritmos de aprendizaje más usados, Q-learning no sirve en nuestro entorno, ya que el videojuego Unreal Tournament es un entorno muy caótico que cambia muy rápido, entonces en el momento de realizar la política de Q-learning la mejor opción del estado futuro puede tener un Q-valor muy elevado, pero en realidad ese estado no llega a producirse porque ha cambiado el entorno muy rápido. En su lugar se utiliza el algoritmo denominado SARSA, que modifica la expresión del aprendizaje Q-learning del siguiente modo:

$$\delta t = \alpha\left[r_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}\right) - Q(s_t, a_t)\right]$$

Donde:

- $Q(s_t, a_t)$ es el Q-valor de la pareja estado-acción en el ciclo t.

- $Q(s_{t+1}, a_{t+1})$ es el Q-valor de la pareja estado-acción elegidos en el siguiente ciclo de decisión.
- r_{t+1} es la recompensa total obtenida durante el siguiente ciclo de ejecución.
- α es la tasa de aprendizaje, controla cuanta importancia se le da a la recompensa más reciente.
- γ es el factor de descuento, define cuánto afectan las recompensas futuras.

Gestión de emociones

Las emociones son una entidad producto de dos aspectos: alteración personal y valoración cognitiva.

Los fenómenos emocionales están formados por una respuesta genérica del sistema autónomo ("arraisal") y por la evaluación cognitiva de tal alteración ("appraisal"). Los *drives* o necesidades del bot varían su valor en cada paso de ejecución siguiendo cada drive una dinámica determinada. Dicho conjunto de dinámicas intenta dotar de "personalidad" al bot y, la variación del conjunto generará bots con diferentes personalidades. Estos valores del *drive*, junto con los valores de los estímulos externos sirven para calcular la intensidad de las motivaciones relacionadas con cada *drive*. La motivación de mayor intensidad es la motivación dominante y va a ser la que determine el estado interno del bot.

Este estado interno junto con el estado externo, determina el estado global del bot.

Estado interno del bot

Es necesario conocer el estado del bot para el proceso de toma de decisiones. En este sistema el estado del agente es la combinación de su estado interno y su estado externo.

$$S = Sinterno * Sexterno$$

El estado interno del agente depende de las motivaciones, que están ligadas a las necesidades, es decir, a los drives del bot y está compuesto por la motivación dominante y los niveles de salud y munición. Los posibles valores del estado interno están reflejados en la Tabla 3. Se considera que la salud es alta cuando es superior a 100, baja cuando es inferior a 40 y media en el resto de casos. La munición se considera alta cuando es superior a 100, baja cuando es inferior a 50 y media en el resto de casos.

Tabla 3. Valores del estado interno.

Estado interno	Valores
Motivación dominante	Agresividad, enfermedad, incapacidad, Ok
Salud	Alta, media, baja
Munición	Alta, media, baja

Los *drives* del bot están relacionados con necesidades fisiológicas y con necesidades de interacción social y seguridad. A medida que esta necesidad aumenta, lo hace la intensidad del *drive*. El valor inicial e ideal de todos los *drives* es cero. En este caso se considera que un *drive* está satisfecho, pues no existe necesidad.

La intensidad del *drive* interior se ve modificada por la fuerza del estímulo externo. Si el valor del *drive* es bajo, se necesita un estímulo fuerte para activar el comportamiento motivado. Por otro lado, si el *drive* es alto, un estímulo medio será suficiente [7]. Siguiendo esta idea, la intensidad de las motivaciones en la arquitectura (Mi) es la suma de la intensidad del *drive* relacionado (Di) y el valor del estímulo externo relacionado (wi), tal y como se expresa en la ecuación (1):

$$Mi = Di + wi \tag{1}$$

Donde

$si\ \ Di < Ld \rightarrow Mi = 0$

$si\ \ Di > Ld \rightarrow se\ aplica\ la\ ecuación$

Mientras todos los *drives* tengan un valor inferior a Ld se considera que el bot no tiene ninguna necesidad, o no hay motivación dominante, y el bot se encuentra en un estado ideal en el que todas sus necesidades están satisfechas.

Estado externo del bot

El estado externo es el estado del bot en relación con algunos elementos, pasivos y activos, del entorno Unreal Tournament que puedan interactuar con el bot. Se ha simplificado el estado externo porque las posibilidades son inmensas y de esta manera se evita que haya un número muy elevado de estados del bot en relación con todos los objetos.

La Tabla 4 contiene los elementos que componen el estado externo.

Tabla 4. Valores del estado externo.

Estado externo	Valores
Veo Salud	Sí, No
Veo Munición	Sí, No
Me Atacan	Sí, No
Veo Enemigo	Sí, No
Peligro	Nulo, bajo, medio, alto,

Función de refuerzo

Los drives del bot están relacionados con sus necesidades. Se define el *mood* del bot como el grado de satisfacción de sus necesidades, de manera que, cuando todos los *drives* del bot están satisfechos el *mood* es máximo. El *mood* del bot es una función de los valores de sus *drives* y factores de personalidad αi que valoran la importancia de cada *drive* respecto del *mood* del bot, siendo

$$Wb = Wbideal - P\alpha i * Di \quad (2)$$

Wbideal ha sido denotado como el valor ideal de *mood* del bot. A medida que aumentan los valores de los *drives*, el *mood* del agente disminuye. Dependiendo de los valores de los factores de personalidad, el aumento de los *drives* puede afectar en mayor o menor medida al *mood*. El valor del *mood* se calcula en cada paso de simulación, además de su variación (ΔWb). Este incremento se puede calcular como:

$$\Delta Wbk+1 = Wbk+1 - Wbk \quad (3)$$

Rolls, en [9] propone que las emociones son estados provocados por refuerzos (recompensa o castigo), de manera que nuestras acciones estarán dirigidas a obtener recompensas y evitar castigos. Siguiendo este punto de vista, en el sistema de toma de decisiones se va a utilizar el *appraisal* como refuerzo positivo y negativo dentro del sistema de aprendizaje por refuerzo. De esta forma el refuerzo también está relacionado con la reducción de los *drives*, ya que las variaciones positivas del *mood* implican una reducción de los *drives* y las negativas un aumento.

Como se ha comentado, el *appraisal* es la evaluación cognitiva de la alteración personal que produce una emoción. En nuestro sistema se define en función de la variación que experimenta el *mood*:

$$si\ \Delta Wb > Lh \rightarrow Appraisal\ \ positivo$$
$$si\ \Delta Wb < Ls \rightarrow Appraisal\ \ negativo$$

Donde ΔWb es la variación del *mood* y $Lh \geq 0$ y $Ls \leq 0$ son las variaciones mínimas del *mood* del agente que producen *appraisal* negativos o positivos.

Mapa de peligro

Habitualmente, los jugadores conocen casi a la perfección los llamados "puntos calientes" de los mapas multijugador, es decir, las zonas del escenario más peligrosas (donde es más probable matar y ser matado). Este peligro puede ser consecuencia de que la zona se encuentre en la línea de fuego de una posición oculta al jugador desde la que el enemigo pueda disparar con facilidad, por ser una zona de tránsito casi obligatorio del mapa y por tanto se congreguen muchos enemigos, o simplemente porque sea fácil acabar siendo víctima de algún elemento del escenario (lava, arenas movedizas, el vacío, etcétera).

Por lo tanto, es interesante contar con un "mapa mental de peligro", análogamente al que "poseen" los jugadores fruto de su experiencia de juego o del propio conocimiento del terreno.

Cada vez que el bot sea aniquilado o vea como otro jugador muere, el valor correspondiente al punto de navegación más cercano al lugar donde sucedió la muerte se incrementará en un valor η ya definido, mayor en el caso de la muerte propia, pues se supone que la manera de jugar es habitualmente más segura frente a la de otros jugadores. Además de incrementarse el valor del punto más cercano, también aumentará el valor de todos los puntos de navegación que se encuentren en un radio inferior o igual a un valor τ en un cuarto del incremento (η/4) del punto central.

En el caso de los mapas pequeños, la propia función de expansión de valor terminará tarde o temprano asignando el valor máximo de Peligro a todos los puntos de navegación del mapa, cumpliendo así la premisa de que en esta clase de mapas, *a priori*, no existen zonas más peligrosas que otras, como se evidencia en la Figura 5.

Modelo de riesgo

La simulación de emociones puede ser algo muy complejo si lo que se busca es imitar el fondo y no así la forma. En un juego de mecánica tan sencilla como el Unreal Tournament 2004, en el que es prácticamente imposible que el comportamiento de

Figura 5. Ejemplo visual de mapa de peligro.

un jugador refleje tristeza, alegría o melancolía, es, sin embargo, muy sencillo emular los sentimientos de un jugador humano de cara al resto de competidores.

Además de la simulación de emociones propuesta en el sistema de toma de decisiones, se propone la característica "riesgo" como un añadido para enriquecerlo.

Según los jugadores expertos, la propia experiencia como jugador enseña que en este tipo de juegos competitivos, la tendencia es que la concentración se vaya perdiendo en función de los resultados en la partida. Esta condición puede simularse simplemente modificando el radio de selección de acciones en el *workspace* de cada capa. El parámetro utilizado como criterio de selección es el nivel de activación de cada acción respecto del contexto.

Debido a que ambos extremos de la emoción provocan el mismo resultado, lo único que se debe hacer es crear un modificador (θ) a este radio de selección en función de la posición del bot en la partida. Calculando la diferencia relativa (en función de la puntuación necesaria para ganar la partida) respecto de estos cuatro valores y asignando un peso a cada una de estas diferencias, se obtiene el modificador para el radio de elección de acciones.

En el bot, este modificador se calcula con la siguiente fórmula matemática basada en la clasificación de la partida:

$$\theta = 1 + \alpha \max\{Sown - Smax / Stop\} + \alpha \min\{Sown - Smin / Stop\} + \alpha next\{Sown - Snext / Stop\} + \alpha prev\{Sown - Sprev / Stop\}$$

Donde:

- *Sown* es la puntuación de nuestro bot
- *Stop* es la puntuación objetivo de la partida
- *Smax* es la puntuación del jugador que va ganando la partida
- *Snext* es la del jugador que está solo un puesto por encima de nuestro bot
- *Sprev* es la puntuación del siguiente jugador en la clasificación de la partida
- *Smin* es la puntuación del último clasificado en la partida
- αmax, αmin, αnext y αprev son los diferentes pesos de estas diferencias (la suma de los pesos debe ser 1) en función de la importancia que se le quiera dar a cada una de ellas.

El modificador θ se envía a la capa física y esta se encarga de almacenar la variable para uso de su *workspace* y de propagar el valor actualizado a las capas superiores.

Integración del sistema de toma de decisiones en CERA-CRANIUM

En el funcionamiento original de CERA-CRANIUM cada ciclo de ejecución (alrededor de cuatro ciclos por segundo), la arquitectura seleccionaba una acción de cada tipo de entre las propuestas por el *workspace* en la capa física en función de su nivel de activación y ejecutaba todas.

Con la integración del sistema de toma de decisiones (y por ende de SOAR) se modifica este funcionamiento de manera que cada vez que se ejecuta un ciclo, de entre todas las acciones propuestas se asigna a cada una un operador de SOAR y se selecciona únicamente una haciendo uso del motor de reglas.

En SOAR se encuentra el estado interno del bot y la capa física introduce en el vector de entrada toda la información que obtiene de los sensores del mundo externo, además de las acciones propuestas

y el módulo de jSoar en su fase de Input del ciclo de ejecución lee la información de este vector de entrada y la introduce en su representación del estado.

En la Figura 6 se resume de manera gráfica la integración del Sistema de Toma de Decisiones (STD) en los flujos *top-down* y *bottom-up* de la arquitectura CERA-CRANIUM.

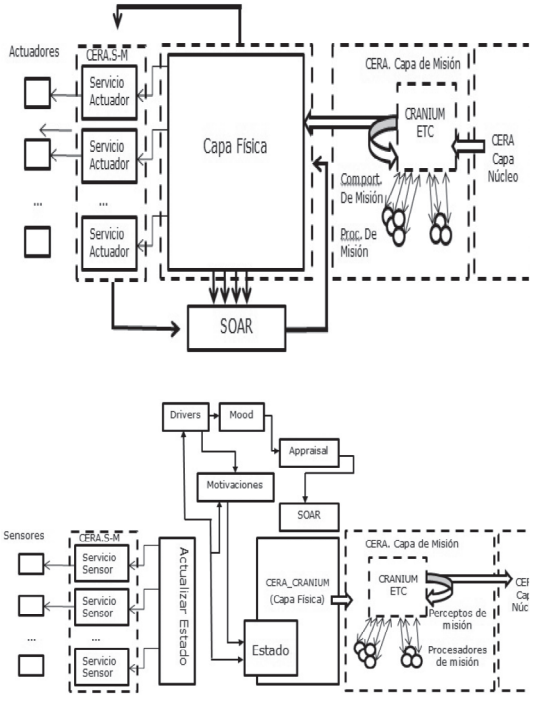

Figura 6. Integración del STD en flujo bottom-up.

Todas estas nuevas implementaciones desarrolladas se han realizado de la manera más genérica posible para respetar el espíritu original de la arquitectura CERA-CRANIUM.

El sistema de toma de decisiones y más en concreto el módulo encargado de calcular el estado (tanto interno como externo) del bot y sus motivaciones pueden modificarse de manera totalmente independiente no solo a la arquitectura en general, sino también al módulo de comunicación con SOAR, posibilitando si se desea en un futuro añadir o cambiar la información utilizada o su funcionamiento.

Aprendizaje por refuerzo

Los valores de la tasa de aprendizaje ($\alpha = 0,3$) y del factor de descuento ($\gamma = 0,8$) de Sarsa son los valores por defecto de Sarsa.

Sistema de toma de decisiones

1. Aunque en un principio se pensó en realizar el control del estado como un procesador CERA-CRANIUM, el hecho de que los procesadores compitan entre ellos complicaba su funcionamiento, ya que era posible que otro procesador "cogiera" un percepto (por ejemplo de presencia de enemigos) y el procesador de estado no pudiera procesarlo en el momento necesario. Esto provocaba un peligroso desfase entre el estado que el sistema enviaba al sistema de toma de decisiones y el verdadero estado del bot.
2. Los valores de los parámetros estímulo motivacional (wi = 1), valor del límite de activación de la motivación (Ld = 2) y *mood* ideal (Wbideal = 100) han sido fijados basándose en los resultados experimentales obtenidos en [39]. Se consigue de esta manera que el bot no se emocione con las novedades, pero tampoco las ignore.

Modelo de Riesgo

1. Los valores utilizados para los pesos en el cálculo de la fórmula son $\alpha max = 0,15$, $\alpha min = 0,15$, $\alpha next = 0,35$ y $\alpha prev = 0,35$. La razón por la que se han establecido mayores pesos a las puntuaciones "siguiente" y "anterior" que a las correspondientes a las puntuaciones "máxima" y "mínima" de la partida, es porque en un comportamiento humano el jugador intentará quedar siempre en la mejor posición posible aunque no pueda ganar, de manera que sentirá más interés por cuánto le queda para subir o bajar un puesto en la clasificación de la partida.
2. Para evitar sobrecargar la ejecución del bot, el valor de riesgo se actualizará únicamente cuando se reciba el evento de que ha muerto otro jugador o cuando sea él mismo el que ha muerto (por otra parte, son estos los únicos casos en los que se actualiza la clasificación en las partidas DeathMatch).
3. El valor riesgo tendrá siempre un valor entre {1,2}. Se ha establecido el radio de selección de acciones inicial a la mitad de la activación máxima para evitar problemas al aplicar el modificador.

Evaluación conscale de ccbotsoar

En la Tabla 5 se recogen las habilidades cognitivas (CS) de la escala ConScale que se cumplen gracias a estas nuevas implementaciones.

Tabla 5. Avances en ConScale de CCbotSOAR

CS	Implementación
3,7	Memoria de Objetos
4,2	Sistema de Toma de Decisiones (Aprendizaje por Refuerzo)
4,3	Sistema de Toma de Decisiones (Aprendizaje por Refuerzo)
5,1-5,3	Sistema de Toma de Decisiones (Motivaciones-Aprendizaje por Refuerzo)
5,6	Mapa de Peligro / Modelo de Riesgo
6,1-6,2-6,3	Sistema de Toma de Decisiones (Motivaciones)

Aunque la implementación satisface la mayoría de los requisitos de los niveles 4 y 5 (todos excepto uno en cada nivel) y algunos de nivel 6 (la mitad, para ser exactos) se clasifica como nivel 3. ConsScale requiere el completo cumplimiento de todas las habilidades cognitivas de un nivel *i* y también de todos los niveles inferiores para poder clasificar un agente como perteneciente al nivel *i*. El índice CQS de un agente puro de nivel 4 es 12.21. La puntuación de este agente (7.21) indica que existen características adicionales, pero el agente está lejos de un agente de nivel 4 que puntuaría 12.21 o más.

Si bien puede parecer que este incremento no es significativo, se debe fijar en el valor CLS para comprender que, realmente, se está más cerca de alcanzar el nivel 4 de lo que se pensaba. El valor CLS (Cumulative Level Score) combina las puntuaciones de todos los niveles en una medida única que sigue una progresión logarítmica. El CLS de los niveles 3 y 4 es, respectivamente, 1.25 y 1.361. El valor CLS del bot alcanza el 1.33, lo que indica que se encuentra a muy poca distancia del nuevo objetivo, que sería el nivel 4.

CONCLUSIONES

Durante el transcurso del artículo se han ido cumpliendo los objetivos marcados inicialmente en la introducción:

Se ha realizado la evaluación de la arquitectura CERA-CRANIUM según la definición de Vernon, Meta y Sandini [11], en esta evaluación se evidenció que la arquitectura necesitaba algún mecanismo de adaptación.

Se ha evaluado y validado el bot CCBot2 en la escala ConScale de habilidades cognitivas. Los requisitos propuestos para alcanzar los niveles 4, 5 y 6 sirvieron como hoja de ruta para las nuevas implementaciones; se han implementado nuevas funcionalidades como la Memoria de Objetos, el Mapa Interno de Peligro, el Modelo de Riesgo y se ha desarrollado e integrado un Sistema de toma de Decisiones dentro de la arquitectura CERA-CRANIUM.

En concordancia, se ha realizado un análisis experimental del comportamiento del nuevo bot (CCBotSoar).

TRABAJO FUTURO

La integración del Sistema de Toma de Decisiones dentro de la CERA-CRANIUM se ha realizado de manera que resulte totalmente transparente al contexto en el que se quiera utilizar, respetando así el concepto de la propia arquitectura. Algunas de las posibles líneas de trabajo futuras para mejorar el bot implementado (CCBotSoar) y la propia arquitectura cognitiva CERA-CRANIUM son las siguientes:

Avances en la escala ConScale de habilidades cognitivas. A la vista de los posibles requisitos que se presentaron en 3.3.2 y que finalmente no fueron desarrollados, es muy posible que con la implementación correcta del requisito CS4,4 (“Capacidad básica de planificación: cálculo de las siguientes acciones secuenciales”) pueda cumplirse también el requisito CS5,5 (“Capacidad avanzada de planificación teniendo en cuenta todas las metas activas”) por lo que el nuevo bot alcanzaría un nivel 5.

Mejora y enriquecimiento de la información manejada en el Sistema de toma de decisiones para una representación más completa de los estados Interior y Exterior del agente así como de las motivaciones utilizadas por el sistema como por ejemplo el odio hacia ciertos jugadores que hayan matado a nuestro bot varias veces.

Aplicación de la arquitectura CERA-CRANIUM a otros contextos para intentar averiguar su verdadero potencial cognitivo. Ya que su estructura lo permite, podrían implementarse múltiples bots orientados a distintos objetivos y en distintos entornos (por

ejemplo otros tipos de partida u otros juegos no FPS) y comparar sus resultados con otras arquitecturas que también puedan ser aplicadas a diversos marcos experimentales.

REFERENCIAS

[1] P. Langley, J.E Laird and S. Rogers. "Cognitive architectures: Research issues and challenges". Cognitive Systems Research. Vol. 10, pp. 141-160. 2009.

[2] J.E. Laird, A. Newell and P. Rosenbloom. "Soar: an architecture for general intelligence. Artificial Intelligence". Vol. 33, pp. 1-64. 1987.

[3] D. Ray. jSoar: "A pure Java implementation of Soar". In The 29th Soar Workshop. Ann Arbor, MI. 2009.

[4] A.M. Turing. "Computing Machinery and Intelligence". Mind. Vol. 49, pp. 433-460. 1950.

[5] R. Arrabales, A. Ledezma and A. Sanchis. "Establishing a roadmap and metrics for conscious machines development". Proceedings of the 8th IEEE International Conference on Cognitive Informatics. 2009.

[6] R.S. Suttony and A.G Barto. "Reinforcement Learning: An Introduction". A Bradford Book. MIT Press. Cambridge, MA. USA. Sec. 4.3.1. 1998.

[7] E. Rolls. "Emotions in Humans and Artifacts, chapter Theory of emotion, its functions, and its adaptive value". MIT Press. 2003.

[8] R. Arrabales, A. Ledezma and A. Sanchis. "Assessing and characterizing the cognitive power of machine consciousness implementations". 2009.

[9] R. Arrabales Moreno and A. Sanchis de Miguel. "A Machine Consciousness Approach to Autonomous Mobile Robotics". In: 5th International Cognitive Robotics Workshop. AAAI-06. 2006.

[10] R. Arrabales, R. Ledezma Espino and A. Sanchis de Miguel. "Modelling Consciousness for Autonomous Robot Exploration". In 2nd International Work-Conference on the Interplay between Natural and Artificial Computation, IWINAC. Lecture Notes in Computer Science Series. Vol. 4527, pp. 51-60. 2007.

[11] D. Vernon, U. Etisalat, G. SharjahMetta and G. Sandin. "A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents". In Evolutionary Computation, IEEE Transactions. Vol. 11, pp. 87-95. 2007.

[12] B.J. Baars. "In the Theatre of Consciousness: Global Workspace Theory, A Rigorous Scientific Theory of Consciousness". Journal of Consciousness Studies. Vol. 4, pp. 292-309. 1997.

[13] C. von Hofsten. "An action perspective on motor development". Trends in Cognitive Science. Vol. 8, pp. 266-272. 2004.

[14] P. Langley. "An adaptive architecture for physical agents". In IEEE/WIC/ACM International Conference on Intelligent Agent Technology. Compiegne, France: IEEE Computer Society Press. Vol. 2, pp. 18-25. 2005.

[15] M.P. Shanahan and B. Baar. "Applying global workspace theory to the frame problem". Cognition. Vol. 98, pp. 157-176. 2005.