Artículo de Investigación. Revista Ciencia e Ingeniería. Vol. 42-3, No. 3 pp. 285-296, agosto-noviembre, 2021.
ISSN 1316-7081. ISSN Elect. 2244-8780 Universidad de los Andes (ULA)

Sistema adaptativo de estrategia para juegos serios emergentes utilizando un sistema de clasificador difuso

Adaptive strategy system for serious emerging games using a diffuse classifier system

Díaz, Francisco[1]; Aguilar, Jose[1,2,3,*]; Altamiranda, Junior[1]
[1]CEMISID, Escuela de Sistemas, Facultad de Ingeniería, Universidad de los Andes, Mérida, Venezuela.
[2]GIDITIC, Universidad EAFIT, Medellín, Colombia.
[3]Dpto. de Automática, Universidad de Alcalá, España.
[*]aguilar@ula.ve,

Resumen

En este trabajo se propone un sistema adaptativo de estrategias (SAE) para juegos serios emergentes (JSE), el cual maneja la emergencia de estrategias dentro del sistema adaptativo de videojuego (SAV), que es el subsistema dentro de un motor de JSE que permite su adecuación al entorno. SAE debe gestionar la emergencia de nuevos personajes reglas, entre otras cosas, en el JSE, para adaptarlo a su entorno. En este artículo, el SAE se usa en un salón de clases inteligente (SaCI), de tal manera que permita la adaptación del JSE a los estudiantes que en algún momento están en SaCI, y lo están usando en sus procesos de enseñanza-aprendizaje Este mecanismo de emergencia forma parte de otros mecanismos del JSE, los cuales permiten la emergencia de secuencias o de los parámetros del JSE, según el contexto actual en el JSE.

Palabras clave: Motor de Juegos Serios Emergentes, Sistema Adaptativo de Videojuego, Salón de Clases Inteligentes, Sistema de Adaptativo de Estrategias, Sistema Clasificador Difuso

Abstract

In this work, an adaptive system of strategies (SAE) for emergent serious games (JSE) is proposed, which allows the emergency of strategies in the adaptive system of video game (SAV) that is the subsystem within a JSE engine that allows its adaptation to the environment. SAE must manage the emergence of new rules, personages, among other things, in the JSE, to adapt it to its environment. In this article, the SAE is used in a smart classroom (SaCI), in such a way that the JSE is adapted to the students who are in a given moment in SaCI, and they are using in their teaching-learning processes. This emergency mechanism is part of other mechanisms of JSE, which allow the emergence of sequences or parameters, according to the current context in the JSE.

Keywords: Serious Emerging Games Engine, Video Game Adaptation System, Smart Classroom, Strategies Adaptive System, Fuzzy Classifier System.

Introducción

El diseño, desarrollo y modo de operar de un JSE está sustentado por la fusión de dos teorías: La teoría de juego serio (JS), de propósito específico al aprendizaje (Barajas y col., 2016); y la teoría de juegos emergentes (JE), cuyo comportamiento (emergente) surge de la interacción (espontánea y sin leyes explícitas) de sus elementos: jugador, personaje, etc. (Steven, 2004, Aguilar y col., 2013, Aguilar, 2014, Aguilar y col, 2014). Así, un JSE hace referencia a juegos de objetivo diferente al de sólo jugar. Es decir, un JSE opera a partir de diversos objetivos (educativos, entrenamientos, rehabilitaciones, etc.), y su dinámica acontece en función del contexto en que se realiza el JSE (Aguilar y col., 2018).

Según (Aguilar, 2014), se habla de emergencia cuando hay nuevas propiedades, comportamientos, estructuras y patrones coherentes a nivel macro, que surgen dinámicamente desde las interacciones entre las partes a nivel micro. Estas propiedades, comportamientos, estructuras, patrones, etc., son nuevas, con respecto a las partes individuales del sistema. Un clásico indicador de emergencia son los patrones observables en un nivel superior, con características temporales y espaciales específicas.

El motor de JSE (MJSE) es un conjunto de programas destinados a la creación, representación, ejecución y adaptación de un JSE (Aguilar y col., 2018). El MJSE funciona con tramas de diferentes videojuegos, adaptándose a requerimientos específicos de su contexto. En (Aguilar y col., 2018) se describe un MJSE, estructurado en capas, que sigue el objetivo específico, de aprendizaje o comprensión de un tema complejo, del JS, el cual permite la emergencia de comportamientos en el contexto, según los requerimientos en el mismo.

El SAV forma parte de la arquitectura del MJSE (Aguilar y col., 2018). El objetivo del SAV es adaptar el JSE al contexto, basado en comportamientos emergentes. El SAV consta de tres componentes (Aguilar, 2014, Aguilar y col., 2018), para permitir tres clases de comportamiento emergente: estrategias, secuencias y propiedades.

En este trabajo, para la emergencia de estrategias se propone el SAE, el cual está basado en un modelo genérico de reglas que define las estrategias y, por la otra, en un sistema clasificador difuso (SCD) que las gestiona y adapta a los objetivos del contexto (Aguilar y col., 2001, Terán y col., 2017).

En este trabajo, para la emergencia de estrategias se propone el SAE, el cual está basado en un modelo genérico de reglas que define las estrategias y, por la otra, en un sistema clasificador difuso (SCD) que las gestiona y adapta a los objetivos del contexto (Aguilar y col., 2001, Terán y col., 2017).

A continuación, comentamos algunos trabajos de interés para nuestra propuesta: en (Rasim y col., 2016) se enfrentan al problema en el que los jugadores tienen diferentes habilidades para jugar los juegos. Esto hace que los jugadores se frustren si el juego es demasiado difícil, o se aburren si es demasiado fácil. Ellos definen un personaje no jugable (NPC) para los JSs, el cual se adapta a los jugadores, de tal manera que puedan sentirse cómodos. En el ámbito del uso de técnicas inteligentes en videojuegos, en (Shafi y Abbass, 2017) presentan un análisis de las técnicas de inteligencia artificial (IA) más utilizadas en los videojuegos, y particularmente, en juegos del tipo de simuladores. Entre las técnicas que señalan en dicho trabajo están los sistemas de clasificación de aprendizaje (LSC, por sus siglas en inglés), las redes neuronales, y los algoritmos genéticos, entre otros.

En cuanto al uso de la lógica difusa en juegos, en (Saeed y col., 2013) se presenta un juego con carreteras montañosas, donde existe deslizamientos de tierra, problemas de congestión, alta tasa de accidentes, etc. El propósito de ese trabajo es minimizar la congestión del tráfico y el tiempo de espera de los vehículos. Ese juego está basado en la teoría de juegos y la lógica difusa. En (Camci y Kayacan, 2016) se propone un simulador de cuadricóptero, usando un modelo del tipo Takagi-Sugeno-Kang como controlador basado en lógica difusa (TSK-FLC) del mismo. En (Esfahlani y col., 2017) se utiliza el videojuego ReHabGame para pacientes con trastornos neuromusculares. El videojuego utiliza una interfaz basada en lógica difusa, para permitir al jugador controlar un avatar a través de un Kinect Xbox, el brazalete Myo y el pedal del timón.

Por otro lado, en (Pan y col., 2017) proponen un juego gráfico que se centra en los objetivos de accesibilidad, denominado juegos de accesibilidad difusa (FRG). En un FRG, el objetivo del jugador es maximizar su valor de verdad al alcanzar un conjunto de objetivos dados, mientras que un otro jugador apunta a lo contrario. A su vez, en (Vasudeva y col., 2017) se utiliza la teoría de juegos y la lógica difusa, para analizar el despliegue denso de redes heterogéneas (HetNets), para hacer frente a las demandas de capacidad en las futuras redes inalámbricas 5G.

Este artículo se organiza de la siguiente manera, en la sección 2 se describen los aspectos teóricos bases de este trabajo, tales como la emergencia de estrategias en JSE, y la arquitectura general de un MJSE y de los SCD. La sección 3 presenta el diseño del SAE para el MJSE. A continuación, se presenta un caso de estudio, y se compara el MJSE con otros trabajos similares, para, finalmente, presentar las conclusiones.

.

2 Aspectos Teóricos

2.1 Emergencia de Estrategia

Los tipos de emergencia clásicos que se pueden dar en un JSE son los siguientes componentes (Aguilar 2014, Aguilar y col., 2018):

- Estrategias: se generan nuevas logísticas y/o tácticas, siguiendo las normas, leyes y reglas del videojuego. Estas logísticas y/o tácticas no han sido diseñadas, creadas, ni predefinidas por el diseñador del juego.
- Secuencia: se crean nuevas tramas (orden cronológico de diversos acontecimientos presentados a un jugador) en los juegos.
- Propiedad: cambia las características de los objetos, lo que puede conllevar a nuevos escenarios, personajes, etc.
- Final: determina cuando debe terminar el videojuego, por ejemplo: hacer emerger vidas infinitas, o finalizar el juego cuando se alcance un objetivo, entre otras cosas.
- Utilidad: hace emerger como se va a utilizar el JSE, en función del contexto o la narrativa del ambiente donde se esté usando.

Particularmente, la emergencia de estrategias consiste en permitir que surjan procedimientos o métodos (tácticas), así como acciones (logísticas), encaminadas hacia un fin determinado, siguiendo las normas, leyes y reglas del videojuego. Los procedimientos, métodos o acciones no han sido predefinidos por el diseñador del juego. Por ejemplo, la emergencia de estrategias de golpes, tácticas de combos de ataque, etc., en videojuegos de combate como Street Fighter, Mortal Kombat, etc., son ejemplos de lo anterior.

La emergencia de estrategias permite cambiar el comportamiento de los personajes en un videojuego, en función de los acontecimientos en el juego y del ámbito donde se desarrolla el juego.

De esta manera, un MJSE debe posibilitar este tipo de emergencia, para lo cual requiere de un conjunto de submotores, y de un sistema que permita la gestión de las diferentes estrategias, que en nuestro caso estará basado en un SCD

2.2 Arquitectura de un MJSE

La arquitectura del MJSE está basada en el trabajo (Aguilar y col. 2018), la cual fue desarrollará para soportar un JSE en un SaCI (la cual se inspira en la teoría de agentes para definir un SaCI, revisar (Aguilar y col., 2005, Aguilar y col., 2007) para más detalles de las bases teóricas). El MJSE está basado en capas jerárquicas (ver Fig. 1).

Fig. 1. Arquitectura del Motor de Juegos Serios Emergentes

La arquitectura está conformada por tres capas, una que corresponde a los componentes de un clásico motor de videojuego, otra que corresponde a la emergencia inicial del JSE adecuado al contexto donde será usado subsistema de emergencia del videojuego (SEV) y otra que va adaptando el JSE mientras se usa SAV, a través de diferentes tipos de emergencia en el juego: de secuencia, de estrategias, etc. La descripción detallada de esas capas se encuentra en (Aguilar y col., 2016, Aguilar y col., 2018), a continuación solo se dará una descripción general de ellas.

i. Núcleo de Motor de Videojuego (NMV): es el elemento central del MJSE, en él se encuentran los seis submotores de base para cualquier videojuego, los cuales son: gráficos, físico, de sonido, de interacción, de video y de renderización. Más detalles de esos submotores en (Aguilar y col., 2018).

ii. Subsistemas: están compuesta por el SEV y de SAV, que son las capas de MJSE que permiten hacer emerger un JSE. Ambos subsistemas usan los siguientes submotores:

a) *Submotor de IA (SIA):* se encarga de introducir comportamientos inteligentes en los diferentes componentes del JSE. Para ello, en este componente se despliegan las diferentes técnicas usadas de la IA para permitir la emergencia en un JSE. Aquí estaría el SCD que gestiona la emergencia de estrategias.

b) *Submotor de Trama Emergente (STE):* es el responsable de hacer emerger en el JSE las narrativas y las secuencias de las tramas adaptadas al contexto (ver Fig. 2).

- ***SEV:*** en el caso del STE, permite hacer emerger la primera versión del JSE que será ejecutada, según los objetivos que se debe cumplir con el JSE. El STE, cuando es invocado por el SEV, está compuesto por los siguientes componentes (ver Fig. 2, y (Aguilar y col., 2016, Aguilar y col., 2018) para más detalles):

1. *Gestor de Materia:* determina la temática que se está tratando en el contexto para, a, y partir de allí, establecer el objetivo que debe cubrir el JSE. Así, crea una emergencia de utilidad

definiendo para qué va a ser utilizado el videojuego.

2. *Gestor de Videojuegos:* busca en repositorios de *videojuegos* (por ejemplo, edugame, advergame, etc.), subtramas o videojuegos para esa temática. Dichas subtramas/videojuegos son definidos como recursos de aprendizaje (RA). Para la búsqueda, compara los metadatos de los RAs en los repositorios con el definido por el *Gestor de Materia*. Si no consigue al menos un videojuego parecido a lo buscado, llama al módulo siguiente.
3. *Módulo de Generación de JSE (MGJSE):* es el responsable del ensamblaje de un nuevo JSE usando las subtramas provistas por el *Gestor de Videojuegos*. En un trabajo previo, se ha definido el MGJSE basado en ACO (Aguilar y col., 2018). El MGJSE tiene imbricadas las funciones de los siguientes tres componentes del SEV, para generar inicialmente un JSE.
4. *Storyboard:* se encarga de generar los guiones narrativos o subtramas del JSE.
5. *Gestor de Escenas:* genera el ambiente o entorno, requerido por las tramas del JSE.
6. *Sistemas de Eventos:* se encarga de generar eventos especializados requeridos por el *Storyboard*, para generar los comportamientos deseados en el JSE.

- **SAV:** en el caso del STE, permite ir adaptando al JSE durante el desarrollo del mismo. La capa SAV permite que en un JSE se generen comportamientos emergentes durante el juego, actuando sobre sus características de base. En particular, en (Aguilar y col., 2018) se proponen 6 niveles de emergencia en un JSE, que se dividen en dos módulos, a saber:

1. *Módulo de Emergencia Fuerte:* este módulo está compuesto de tres subcapas, que permiten las siguientes emergencias fuertes en el videojuego:
   - *Estrategias:* se generan nuevas tácticas y logística en el juego.
   - *Secuencia:* se crean nuevos escenarios o tramas en los juegos, cambiando el ambiente y el contexto del juego.
   - *Propiedad:* cambia las características en los objetos.
2. *Módulo de Emergencia Débil:* este módulo está compuesto de tres subcapas, que permiten las siguientes emergencias débiles:
   - *Final:* patrones que determinan la culminación de los juegos, o continuar el juego de forma infinita.
   - *Modelo de Negocio:* surgimiento de mercados y servicios alrededor de los JSEs.
   - *Utilidad:* uso del JSE

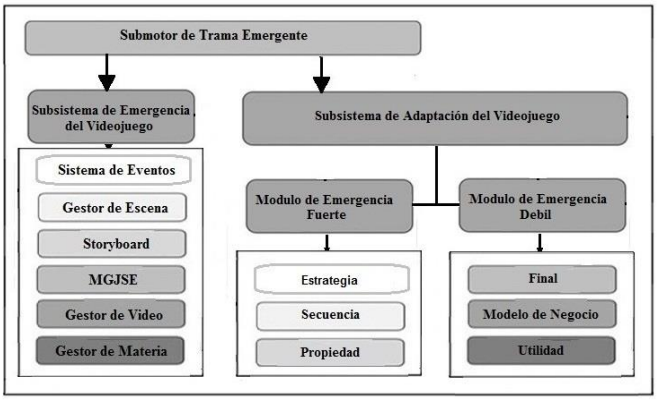

Fig. 2. Submotor de Trama Emergente.

La Fig. 2 describe de manera general al STE, que como se comentó antes, es invocado tanto por el SEV como por el SAV, para permitir la emergencia en un JSE. En el caso del SEV, realiza la emergencia de la primera versión del JSE adaptado al contexto donde será usado, para lo cual usa los componentes de base para la construcción de un JSE (gestor de escenas, etc.). En el caso del SAV, adecua el JSE durante su uso, bajo diferentes mecanismos de emergencia: estrategias, secuencias, etc. Todo ello ocurre, sin modificar su núcleo NMV. En este artículo solo se especifica la emergencia de estrategia.

2.3 Sistemas Clasificadores Difusos (SCD)

Un *Sistema Clasificador* es un tipo de máquina de aprendizaje basada en reglas (Aguilar y Rivas, 2001, Terán y col., 2017). Particularmente, un SCD está basado en un conjunto de reglas o clasificadores, los cuales son de la forma *Si* <*condición*> *Entonces* <*acción*>, tal que las condiciones y acciones son definidas por variables difusas.

De esta manera, la activación de una regla se logra cuando se cumplen las instancias de la <condición> de dicha regla. El peso de cada regla está basado en el grado de activación de la misma. Específicamente, la importancia de una regla viene definida por la ecuación (1):

$$S_i(t+1) = S_i(t) + Acti_i(\mathrm{t}) - PS_i + R_i(t) \qquad (1)$$

Donde $Acti_i$(t) es el grado de activación de la regla *i*; $PS_i(t)$ es el pago dado por la activación de la regla; $R_i(t)$ es el pago recibido, definido como $\sum_{j=D} Pr * Act_j(t)$, tal que *D* es el conjunto de reglas que activó la regla *i* en el instante t; y Pr es la rata de pago, dada como un parámetro del SC. La función de ajuste para la función de pertenencia de un conjunto difuso F, viene dada por (Menolascina y col., 2005):

$$S_F(t+1) = S_F(t) + Acti_i * \mu_{F[x_k]}, \qquad (2)$$

Si el operador de la condición es "O".

$$S_F(t+1) = S_F(t) + Acti_i * \frac{1}{\mu_{F[x_k]}}, \qquad (3)$$

Si el operador de la condición es "Y".

Dónde $S_F(t)$ es el valor de crédito de la función de pertenencia del conjunto difuso F en el tiempo t; $\mu_{F[x_k]}$ es el grado de pertenencia del elemento x_k en el conjunto difuso F presente en la <condición>.

La definición de las ecuaciones (2) y (3) permiten asignar más crédito al conjunto difuso que influyo más en el nivel de disparo de una regla (Aguilar and Rivas, 2001). El macroalgoritmo (ver Fig. 3) de un SCD para ajustar las reglas es:

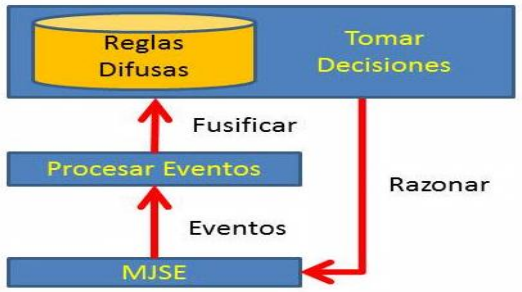

Fig. 3. Modelo del algoritmo de SCD

Inicializar SCD

Repita Mientras *(existan eventos)*

Fusificar (evento)

Activar (SCD, evento)

Actualizar (SCD)

3 Sistema de Adaptación de Estrategia

La capa llamada sistema de adaptación de estrategias **(**SAE), permite la emergencia de estrategias en un JSE. Para realizar esas tareas, se apoya en el SIA y STE. La emergencia de estrategias genera nuevas tácticas y logísticas en el juego, sin dejar de seguir las normas, leyes y reglas del mismo. El punto relevante, en este caso, es definir como se modelarán las estrategias. En nuestro caso, ellas serán definidas por reglas, en las cuales en el antecedente de la regla se establece qué debe suceder (eventos que deben ocurrir), y en el consecuente las acciones que deben ocurrir en el juego dado esos eventos. También, es fundamental definir como se adaptarán las reglas, que en nuestro caso será usando un SCD. En la Fig. 4 se presenta el modelo de SCD a usar.

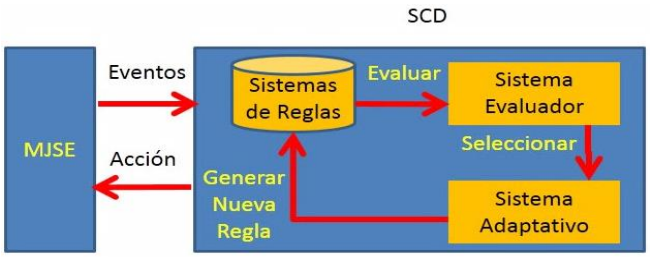

Fig. 4. Modelo de SCD

- MJSE*:* genera eventos cada uno de sus submotores
- Sistemas de Reglas: este módulo contiene las reglas del sistema, y además, recibe los eventos que van ocurriendo en el juego, para luego fusificarlos.
- Sistema Evaluador*:* según los eventos recibidos, se activan las respectivas reglas. En menos palabras, es el razonar difuso del SCD.
- Sistema Adaptativo: el SCD va adaptando las reglas, en función de sus comportamientos durante el juego. Aquellas más efectivas (más usadas, permiten alcanzar el objetivo del juego, etc.), perduran más en el tiempo, generando nuevas reglas.

A continuación, se definen todos los elementos que permiten caracterizar el SCD para la emergencia de estrategias.

3.1 Variables Difusas y Conjuntos Difusos

Tanto los eventos de cada uno de los submotores, como las acciones en el JSE, serán definidos por las siguientes variables y conjuntos difusas:

- Contexto JSE (CJ): representan variables que definen el tema del JSE (Menolascina y col., 2005) (CJx para [x=0,1,2,3…n]). Por ejemplo, en un JSE en el contexto educativo, suponiendo una clase de matemática, las variables difusas podrían ser: suma, resta, multiplicación o división de números reales.Todas tendrían los mismos conjuntos difusos, los cuales serían: Ninguna (N), Escasa (E), Mediana (M) y Alta (A).
- Evento Físico (EF): representa eventos que ocurren en el juego (EF_x para [x=0,1,2,3…m]). Por ejemplo, un

carro moviéndose, un personaje saltando, un futbolista jugando, el avatar tropieza una pelota, etc. Los conjuntos difusos son: Verdadero (V), Más o menos (MM) y Falso (F).

- Evento Acústico (EA): representa tipos de eventos auditivos, como: cantar, gritar, el sonido de un rayo, el ruido al partirse un vaso, etc. (EA_x donde [x=0,1,2,3…p]). Los conjuntos difusos son: V, MM y F.
- Evento Cámara (EC): eventos que ordenan el movimiento, tanto en posición como en rotación, de la cámara que muestra el juego al jugador (EC_x para [x=0,1,2,3…r]).Por ejemplo: mirar hacia arriba o hacia abajo. Los conjuntos difusos considerado para estas variables son: N, E, M y A.
- Evento de Video (EV): es cuando en el videojuego se utiliza una animación, efecto especial, o video. Por ejemplo: cuando aparece un efecto especial de larga duración, un videoclip, etc. (EV_x, donde [x=0,1,2,3…q] representa los tipos de eventos videos). Los conjuntos difusos son: V, MM y F.
- Acción de Movimiento (AM): se encarga de pedir al MJSE que aplique un movimiento sobre al avatar (AM_x para [x=0,1,2,3…s]). Sus conjuntos difusos son: N, E, M y A.
- Acción de Destreza (AD): son teclas especiales del videojuego que varían según el tipo de habilidad a permitir: salto, agachar, abrir, cerrar, patear, agarrar, soltar, etc. o cualquier otra actividad en el JSE (AD_x para [x=0,1,2,3…t]). Los conjuntos difusos son: N, E, M y A.
- Acción de Avanzada (AA): define una acción que es disparada por un algoritmo de IA después que obtiene una solución (AA_x [x=0,1,2,3…v]). Por ejemplo, después de utilizar los arboles de comportamiento (BT) en Halo 2, o min-max en ajedrez. Los conjuntos difusos son: N, E, M y A.

3.2 Funciones de Pertenencia

A continuación, se definen las funciones de pertenencia de cada uno de los conjuntos difusos asociados a cada variable difusa. Se propone, de manera general, el uso de funciones de pertinencia del tipo trapezoidal. Las variables difusas EF, EA y EV, son caracterizadas por las funciones de pertenencia mostradas en la Fig. 5.

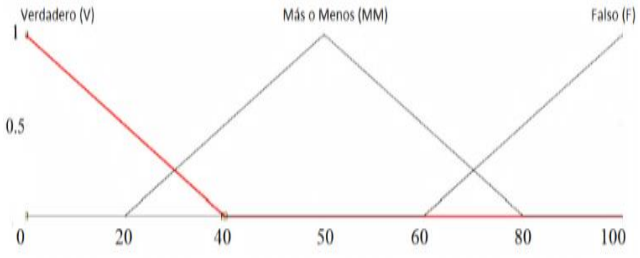

Fig. 5. Función de **pertene**ncia de EF, EA, EV

Cada una de esas variables difusas tiene un universo de discurso del [0%, 100%], y están compuestas por los conjuntos difusos V, MM y F. Cada conjunto difuso tiene una función de pertenencia triangular, variando entre 0-40%, 20-80% y 60-100%, respectivamente.

Las variables difusas CJ, AD, EC, AM, AD y AA se caracterizan por las funcione de pertenencia mostradas en la Fig. 6, y están compuestas por los conjuntos difusos N, E, M y A. Cada conjunto difuso tiene una función de pertenencia triangular, variando entre 0-20%, 10-50%, 40-80% y 70-100%, respectivamente.

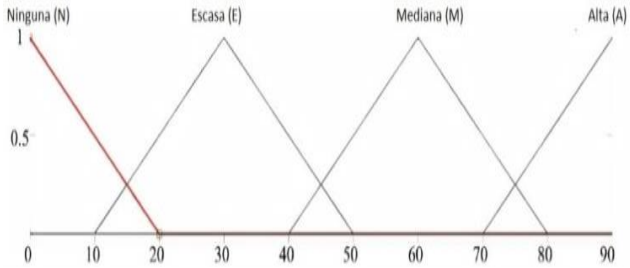

Fig. 6. Funciones de pertenencia de CJ, AD, EC, AM, AD y AA

3.3 Reglas de Control Genéricas

A continuación, se presenta un grupo de reglas asociadas a las variables difusas, las cuales se dividen según el tipo de estrategia de juego:

- JSE de agilidad: son juegos de saltos y poderes, como por ejemplo, Mario Bros, Donkey Kong, etc. Un ejemplo de sus posibles reglas serían:

  Si <EF=hueco> entonces

  <AM=saltar_adelante>

  Si <EF=enemigo> entonces

  <AD=disparar_enemigo>

- JSE de conjetura: son juegos de cálculo, como por

ejemplo: suma, resta, multiplicación, memoria, etc.

Si <CJ=sumar>entonces <AA=calcular>

Si <EO=valido>entonces <AM=mover>

Si <EO=vacío>entonces <AM=mover>

- JSE de velocidad: son juegos de manejo de algún vehículo, como por ejemplo: carro, moto, avión, barco, bicicleta, etc. Un ejemplo de sus posibles reglas serían:

Si <EF=chocar> y <EA=gritar>entonces

<AA=proteger>

Si <EF=chocar> o <EF=parar> entonces

<AM=parar>

Existen otros grupos de reglas de estrategias, vinculadas a JSE de lucha, rompecabezas, entre otros.

3.4 Comportamiento de las reglas difusas (instancias)

A continuación, se dan ejemplos de algunas de las instancias de las reglas de control genéricas, que podrían definir las estrategias en un JSE dado:

Si EF=hueco es V y EF=enemigo es V entonces

AM=saltar es A y AD=disparar es A

Esta regla indica que si un evento físico es un hueco y otro es un enemigo, entonces se realizan las acciones saltar y disparar al mismo tiempo.

Si (EF=chocar es V o EA=grita es MM)

Entonces AA=proteger es A.

En esta regla se establece que si ocurre un evento físico de chocar y más o menos un evento auditivo, entonces se invoca a una técnica de IA que permita establecer la estrategia de protegerse.

4 Caso de Estudio

4.1 Contexto de aplicación: SaCI

Se considera un SaCI como el propuesto en (Chamba-Eras y col., 2017, Sánchez y col., 2015), tal que en el aula inteligente todos sus componentes son modelados usando el paradigma de sistemas multiagentes (SMA), el cual caracteriza a sus dispositivos de hardware (*pizarra inteligente*, *laptop*, *tableta*, *smartphone*, etc.) y de software *(*entorno virtual de aprendizaje *(VLE*, por sus siglas en inglés*)*, sistema académico, etc.) como agentes.

El SaCI utiliza un middleware reflexivo autonómico para entornos de aprendizaje inteligente en la nube, llamado AmICL, propuesto en (Chamba-Eras y col., 2017, Sánchez y col., 2015, Sánchez y col., 2020). Este middleware reflexivo autonómico está basado también en SMA, y posibilita el despliegue de las diferentes comunidades de agentes de SaCI.

En particular, uno de esos agentes es el Agente Juego Serios Emergentes (AJSE), el cual gestiona los JSEs de forma autónoma. El AJSE, una vez recolectada la información del entorno de SaCI (perfil de los estudiantes, objetivos del actual proceso de aprendizaje, etc.), adapta el JSE a SaCI, llamando al *Gestor de Materias*. Para ello, el AJSE interactúa con los agentes de SaCI: gestor del repositorio de objeto de aprendizaje (ROA), sistema académico (SA), sistema recomendador (RS) de recursos educacionales, y el VLE (ver Fig. 7) (Díaz y col., 2019).

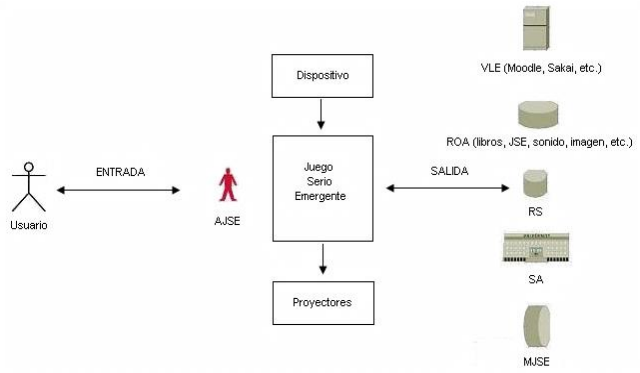

Fig. 7. Modelo conceptual de un AJSE en un SaCI.
Fuente: (Díaz y col., 2019)

A partir de esas interacciones, el AJSE usa el MJSE para hacer emerger el JSE adecuado para ese momento en SaCI el SEV, y después, lo va adaptando mientras se va usando el SAV. Para esto último, usa mecanismos como el presentado en este trabajo, que permite la emergencia de estrategias en un JSE.

4.2 Contexto dinámico pedagógico para analizar el comportamiento del SAV

Una vez en el SaCI, se parte de la hipótesis que se está en una clase de matemáticas y se requiere definir un JSE con el objetivo de explicar y resolver problemas con fracciones.

En (Aguilar y col., 2018) se muestra cómo se genera inicialmente el JSE usando los componentes del MJSE, para explicar la clase de fracciones. En particular, se usa el repositorio de objetos de aprendizaje *agrega* (ver, http://agrega.educacion.es), y se seleccionan tres videojuegos del dómino en fracciones compatibles con el

tema de aprendizaje (ver Fig. 8).

Fig. 8. Tres videojuegos de fracciones utilizando domino.
Fuente (Aguilar y col., 2018)

Se supone que inicialmente el SEV propone como JSE inicial, el videojuego dómino "Combinado" (ver Fig. 9).

Fig. 9. El videojuego de domino "Combinado"

El videojuego de dómino "Combinado" consiste en que cada uno de los dóminos tiene dos zonas opuestas entre sí, ya sea izquierda (arriba) o derecha (abajo). Estas zonas van a tener: una figura y una fracción, calculo o valor, donde el segundo da un resultado de fracciones matemáticas modelada en la figura del dómino. Para jugar un dómino, su figura o cálculo debe ser igual a un dómino disponible en el tablero (con alguno de sus zonas aun sin un domino adyacente), ocupando los espacios vacíos en el tablero. Por ejemplo, en la Fig. 9, el círculo negro señala la zona izquierda de un dómino aun no jugado, que tiene como valor 12/32. Este debe ser colocado en el tablero, adyacente a un dómino con una zona disponible, que tenga el mismo resultado, en uno de los espacios vacíos disponibles (en la Fig. 9 solo hay 3). En este caso es $\sqrt{9/64}$ = 0,375, que se encuentra en el espacio vacío abajo a la izquierda. A continuación, se explica cómo SaCI usa el SAV para la emergencia de estrategias.

4.3 Emergencia Fuerte de Estrategias

Esta sección describe el proceso de emergencia de estrategia gestionado por el AJSE. A partir de los datos del contexto de SaCI proporcionados por los otros agentes, y con el JSE generado inicialmente por el SEV, el SAV invoca al SCD para ir adecuando las reglas que definen las estrategias del JSE, las cuales van emergiendo a través del tiempo.

- Eventos y Acciones: mediante el SCD se establecen los diferentes tipos de eventos y acciones que pueden suceder en el juego (ver Tabla 1), con sus respectivos valores. Por ejemplo, en el caso de CJ, el contexto educativo es utilizar fracciones de las matemáticas. En el caso de EF, los movimientos de los números (en este caso, los dóminos). En el caso de las acciones, las posibles son realizar un cálculo AA o moverse AM.

Tabla 1. Eventos y acciones en el JSE

Estado	Variables	Valores posibles
Entradas		
Evento	CJ	fracciones
	EF	movimientos de los números (dominós)
Salida		
Acciones	AA	realizar un cálculo
	AM	moverse

- Estrategias: se establecen los tipos de reglas para el contexto del JSE. Como es resolver fracciones matemáticas, se deben establecer reglas genéricas alrededor de esa temática. La Tabla 2 muestra algunos ejemplos de dichas reglas. La regla 1 establece que por ser un juego de dómino, entonces se debe realizar una acción de calcular. La regla 2 indica que después de calcular, entonces ocurre el evento EF=dato. Finalmente, la regla 3 indica que al llegar el evento dato, entonces ocurre la acción AM=mover.

Tabla 2. Ejemplo de Reglas Genéricas

N°	Regla
R1	Si <CJ=matemáticas> entonces <AA=calcular>
R2	Si <AA=calcular> entonces <EF=dato>
R3	Si < EF =dato>entonces <AM=mover>

- Ejecución del JSE: a continuación, se muestra un ejemplo del desarrollo del JSE:

1er paso: se activa la regla R1, tal que la primera acción es AA=calcular, la cual puede ser: sumar, restar, dividir, etc.

2do paso: se activa la regla R2, que es AA=calcular el primer dómino de arriba hacia abajo, lo que da el valor del dato=12/32 | 3/8. A continuación, se compara con los datos de los 3 espacios vacíos que se observan en la Fig. 10.

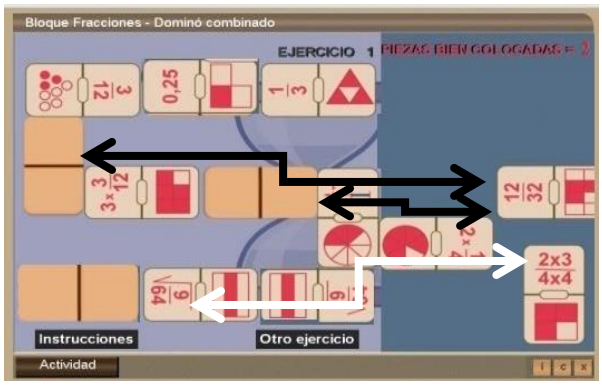

Fig. 10. Comienzo

En este caso, se generan los EFs = $EF_{a1}, EF_{a2}, EF_{b1}, EF_{b2}, EF_{c1}, EF_{c2}$ que dan los resultados de la comparación. Por ejemplo: la zona izquierda (12/32) se compara con la pieza (3/8) de la parte de arriba del espacio vacío que señala la flecha a, y la figura (3/8) de la zona derecha se compara con 3x(3/12) de la zona de abajo del espacio vacío b dando como resultado que es falso=F, como se muestra en la Tabla 3, y así se hace sucesivamente para cada espacio vacío (fila).

Tabla 3. Listados de pruebas difusa para 12/32 | Figura (3/8)

EF		**Dato=** 12/32 | Figura(3/8)	**AA**
A	1	(Figura(3/8)=12/32)=F y (Figura(3/8)=3x(3/12))=F	Falso
	2	(Figura**(3/8)=** Figura**(3/8))=V** y (12/32=3x(3/12))=F	Falso
B	1	(Figura(1/6)=12/32)=F y Figura(3/8)=1/2)=F	Falso
	2	(Figura(1/6)= Figura(3/8))=F y (12/32=1/2)=F	Falso
C	**1**	**(null=12/32)=MM y (Figura(3/8)=$\sqrt{9/64}$)=V**	**Verdadero**
	2	**(null= Figura(3/8))=MM y (12/32=$\sqrt{9/64}$)=V**	**Verdadero**

3er paso: en la Tabla 3, generada por R2, se observa que existen 6 posibles soluciones del primer domino con el dato =12/32 | Figura (3/8), pero solo 2 válidas (c1 o c2). Con las válidas, se dispara la regla R3, para mover el dato =12/32 | Figura (3/8) según lo permitido por EF_{c1} o EF_{c2}. Según eso, las opciones son colocar el dómino en 2 posiciones posibles 12/32 | Figura (3/8) o viceversa, Figura (3/8) | 12/32, que es el mismo resultado de la zona izquierda del dómino colocado en el tablero $\sqrt{9/64}$ = 12/32 = 3/8 = 0,375, el cual da como resultado V; y null (vacío) del otro lado, el cual da como resultado más o menos = MM. Al escoger al azar una de las dos, se mueve el dómino como lo indica la flecha blanca en la Fig. 11.

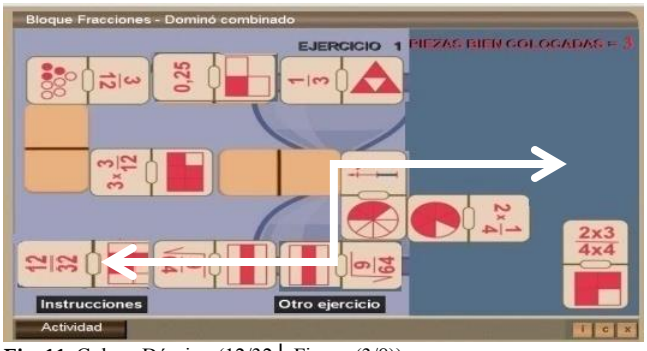

Fig. 11. Coloca Dómino (12/32 | Figura (3/8))

- Iteraciones: se evalúan nuevas iteraciones que aparecen en el juego Por ejemplo, después del último movimiento, la regla R2 genera la Tabla 4. Como se observa en la Tabla 4, existen 4 posibles soluciones ($EF_{d1}, EF_{d2}, EF_{e1}$ y EF_{e2}), con una sola valida EF_{d2}.

Tabla 4. Listados de pruebas difusa para Figura (1/6) | 2x1/4

EF		**Dato=** Figura(1/6) | 2x1/4	**AA**
D	**1**	(Figura(1/6)=2x1/4)=F y (Figura(1/6)=1/2)=F	Falso
	2	**(Figura 1/6)= Figura(1/6))=V y (2x1/4=1/2)=V**	**Verdadero**
E	**1**	(Figura(3/8)= Figura(1/6))=F y (2x1/4=3x3/12)=F	Falso
	2	(Figura(3/8)=2x1/4)=F y (Figura(1/6)=3x3/12)=F	Falso

Posteriormente, se dispara la regla R3, para realizar el siguiente movimiento según EF_{d2}, donde la flecha blanca indica hacia donde se movió la pieza, quedando como se observa en la Fig. 12.

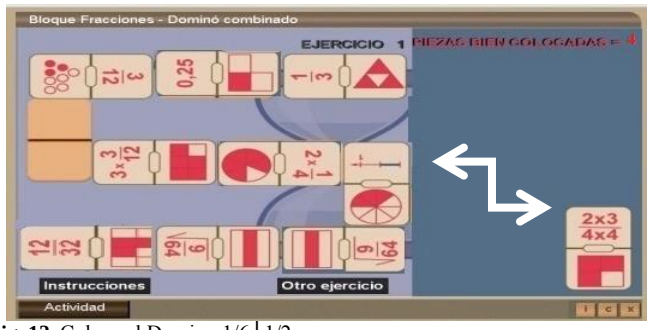

Fig. 12. Coloca el Domino 1/6 | 1/2

En la siguiente iteración, de nuevo se vuelve a activar R2, generando la Tabla 5. Como se observa en la Tabla 5, existen 2 posibles soluciones (EF_{j1}, EF_{j2}) con una sola valida EF_{j1}.

Tabla 5. Listados de pruebas para 2x3/4x4 | Figura (1/4)

EF		Dato= 2x3/4x4 | Figura(1/4)	AA
F	1	**(Figura(3/8)=2x3/4x4)=V y Figura(1/4)=3x(3/12))=V**	**Verdadero**
	2	(Figura(3/8)= Figura(1/4))=F y (2x3/4x4=3x(3/12))=F	Falso

Al final, se realiza el último movimiento según lo establecido por EF_{j1}, moviendo la última pieza del dómino según la flecha blanca de la Fig. 13.

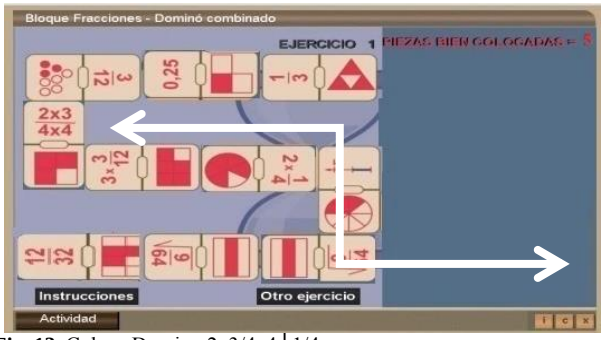

Fig. 13. Coloca Domino 2x3/4x4 | 1/4

- Actualización de la regla: basado en las reglas R1, R2 y R3, el SCD puede producir un proceso de generación de nuevas reglas (estrategias). Por ejemplo, la Tabla 6 muestra una posible nueva regla R4 generada con ellas.

Tabla 6. Nueva Regla Generada por el SCD

N°	Regla
R4	Si CJ=matemáticas es A y AA=calcular es A entonces EF=dato es V y AM=mover es A

Una vez actualizada las reglas, se inicia un nuevo ciclo (iteración) del SCD. Si se dejase el JSE durante un tiempo funcionando en SaCI para un curso y grupo de usuarios, ira estabilizando el grupo de reglas de estrategias adecuadas para ese contexto en el tiempo (como las mostradas en Tablas 2 y 4). En menos palabras, este mecanismo hace emerger el grupo de reglas estratégicas del JSE para un contexto dado de SaCI.

5 Comparación con otras Propuestas

La comparación que se propone realizar es basada en si permiten emergencias de estrategias en el JSE, si usan la lógica difusa, qué técnica difusa usan, y cómo las usan en el juego (ver Tabla 7).

El desarrollo de JS en (Rasim y col., 2016) involucra un motor adaptable, que mediante la aplicación de una máquina de aprendizaje puede adaptarse a diferentes jugadores. En (Saeed y col., 2013) aplican la teoría de juegos y control basado en lógica difusa para la simulación vehicular, que deben librarse de obstáculos de derrumbes o deslizamiento de tierra, roca o lodo, debido a diferentes razones: deforestación, terremotos, erupciones volcánicas, fuertes lluvias, entre otros. En (Camci y Kayacan, 2016) se utiliza el modelo difuso Takagi-Sugeno-Kang (TSK) para controlar cuadricopteros.

En (Esfahlani y col., 2017) proponen un JS para la rehabilitación física basado en un Kinect de Xbox One, que permite el seguimiento de articulaciones en un pedal del timón. El sistema usa reglas difusas del tipo Mamdani's que definen las estrategias de rehabilitación, las cuales se van adecuando de acuerdo al jugador, en función del rendimiento de la persona en la terapia y su nivel de habilidad física. En (Pan y col., 2017) se describen videojuegos con incertidumbre estocástica, donde se utiliza un sistema de transición difusa (FTS, por sus siglas en inglés) para hacer emerger estrategias. El objetivo de un jugador es maximizar su valor para alcanzar tareas mientras que el rival apunta a lo contrario.

En (Vasudeva y col., 2017) se diseñan reglas de inferencia difusas para las decisiones de transferencia y la selección de la estación base destino en redes HeTNets, teniendo en cuenta la eficiencia energética/espectral. El sistema de inferencia utiliza la teoría de juegos basado en lógica difusa para abordar este problema. En (Subbaraj y Savarimuthu, 2014) se propone un modelo de juego no cooperativo para el ruteo seguro y tolerante a fallos en redes móviles ad hoc (MANET) basado en ACO. También utiliza un enfoque integrado de confianza y certificado de autoridad basado en lógica difusa, para el intercambio seguro de datos, aislando los nodos dañinos de la comunicación.

En comparación con los anteriores trabajos, en el presente trabajo se propone un sistema para la gestión de la emergencia de estrategia, usando un SCD, para JSE. Es el único trabajo que propone incorporar la emergencia de estrategias en MJSE, como mecanismo adaptativo de los JSEs. La emergencia permitida en otros trabajos, es vinculada a tomas de decisiones durante el desarrollo del juego, lo cual no impacta la propia dinámica del sistema (en nuestro caso, el JSE).

Tabla 7. Comparación con otros trabajos recientes

Técnica Aplicada	Rasim y col., 2016	Saeed y col., 2013	Camci y Kayacan, 2016	Esfahlani y col., 2017	Pan y col., 2017	Vasudeva y col., 2017	Subbaraj y Savarimuthu, 2014	Presente Trabajo
Emergen estrategias				X	X			X
Capacidad Adaptativa del Juego	X			X	X			X
Usan lógica Difusa		X	X	X	X	X	X	X
Para qué	Motor Adaptativo	Simulación Vehicular	Control Cuadracopteros	Estrategias de Rehabilitación Física	Toma de Decisión	Toma de Decisiones	Certificado de Autoridad y Confianza	MJSE
Técnica difusa		Control Difuso	TSK	Mamdani's	FTS	Teoría de Juegos Difusa	Reglas	SCD

6 Conclusiones

En este artículo presentamos el componente SAE para un MJSE, el cual permite la emergencia fuerte de estrategias, de tal manera de adaptar las tácticas y lógicas de un JSE a los eventos que se van dando en un JSE. En particular, el SAE se probó en un SaCI, de tal manera de permitir que el JSE se adapte al proceso de enseñanza-aprendizaje que se esté dando en un momento dado en él.

Particularmente, el SAE permite la emergencia de estrategias, en forma de reglas, las cuales impactan el comportamiento en el JSE. Para ello se usó un SCD, que gestiona las estrategias de acuerdo a sus usos durante el juego, con la finalidad de crear nuevas reglas, y hacer desaparecer aquellas que no son útiles.

Como proyecto futuro se realizaran pruebas del MJSE en diferentes contextos, además de SaCI, y en el caso concreto de SaCI, en diferentes procesos de enseñanza-aprendizaje. En todos los casos, se evaluara su impacto en los mismos usando indicadores propios a cada contexto, y en especial en SaCI, usando indicadores pedagógicos que permitan mostrar su efectividad en dichos procesos. También, se explotara la analítica de aprendizaje (Aguilar y col., 2019) y las propiedades de auto organización (Perozo y col., 2013) para permitir la autogestión.

Conflicto de intereses:

Los autores declaran que no hay conflicto de intereses.

Referencias

Aguilar J, Rivas F, 2001 Introducción a las técnicas de computación inteligente, Editors, MERITEC.

Aguilar J, Cerrada M, Mousalli G, Rivas F, Hidrobo F, 2005, A Multiagent Model for Intelligent Distributed Control Systems, Lecture Notes in Artificial Intelligence, Vol. 3681, pp. 191-197

Aguilar J, Hidrobo F., Cerrada M, 2007, A Methodology to Specify Multiagent Systems, Lecture Notes in Artificial Intelligence, Vol. 4496, pp. 92-101.

Aguilar J, 2014, Introducción a los sistemas emergentes, Talleres Gráficos, Universidad de Los Andes, Mérida.

Aguilar J, Altamiranda J, Chávez D, 2016, Metropolis: an Emerging Serious Game in a Smart City, Revista Venezolana de Computación, Vol. 3, No. 2, pp. 38-48.

Aguilar J, Buendia O, Pinto A, Gutiérrez J, 2019, Social learning analytics for determining learning styles in a smart classroom, Interactive Learning Environments

Aguilar J, Altamiranda J, Díaz F, 2018, Design of a Serious Emerging Games Engine Based on the optimization Algorithm of Ant Colony. DYNA. Vol. 85, No. 206, pp. 311-320.

Barajas A, Álvarez F, Muñoz J, Oviedo A, 2016, Process for modeling competencies for developing serious games, Revista Electrónica de Investigación Educativa, Vol. 18, No. 3, pp. 146-160.

Camci E, Kayacan E, 2016, Game of Drones: UAV Pursuit-Evasion Game with Type-2 Fuzzy Logic Controllers Tuned by Reinforcement Learning, Proceeding IEEE International Conference on Fuzzy Systems, pp. 618-625.

Chamba-Eras L, Aguilar J (2017) Augmented Reality in a Smart Classroom—Case Study: SaCI, *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, Vol. 12, No. 4, pp. 165-172.

Díaz F, Aguilar J, Altamiranda J, 2019, Specification of a Managing Agent of Emergent Serious Games for a Smart Classroom, IEEE Latin America Transactions, Vol. 18, No. 1, pp. 51-58.

Esfahlani H, Cirstea S, Sanaei A, Wilson G, 2017, An adaptive self-organizing fuzzy logic controller in serious game for motor impairment rehabilitation. Proceeding IEEE Int. Symp. Ind. Electron, pp. 1311-1318.

Pan H, Li Y, Cao Y, Li D, 2017, Reachability in Fuzzy Game Graphs, Proceedings IEEE Transactions on Fuzzy Systems. Vol. 25, No. 4, pp. 984-972.

Perozo N., Aguilar J., Terán O., Molina H., 2013, A Verification Method for MASOES, IEEE Transactions on Cybernetics, Vol. 43, No. 1, pp. 64-76.

Rasim T, Langi A, Munir S, Rosmansyah Y, 2016, A survey on adaptive engine technology for serious games. Proceedings of International Seminar on Mathematics, Science, and Computer Science Education 1708. Vol. 1708, No. 1, pp. 050003-1–050003-9.

Saeed Y, Aziz LS, Ahmed K, 2013, Obstacle Management in VANET using Game Theory and Fuzzy Logic Control.

Proceedings International Journal of Scientific & Engineering Research, Vol 4 No.1, pp. 9- 13.
Sánchez M, Aguilar J, Valdiviezo P, Cordero J, 2015, A Smart Learning Environment based on Cloud Learning, International Journal of Advanced Information Science and Technology, Vol. 39, No. 39, pp. 36-49.
Sanchez M, Exposito E, Aguilar J, 2020, Autonomic computing in manufacturing process coordination in industry 4.0 context, Journal of Industrial Information Integration, Vol19,
Shafi K, Abbass H, 2017, A Survey of Learning Classifier Systems in Games, IEEE Computational intelligence magazine, pp. 42-55.
Steven J, 2004, Sistemas Emergentes: O qué tienen en común hormigas, neuronas, ciudades y software, Ediciones Turner/Fondo de Cultura Económica, España.
Subbaraj S, Savarimuthu P, 2014, EigenTrust-based non-cooperative game model assisting ACO look-ahead secure routing against selfishness, Proceedings EURASIP Journal on Wireless Communications and Networking, pp. 1-20.
Terán J, Aguilar J, Cerrada M, 2017, Integration in industrial automation based on multi-agent systems using cultural algorithms for optimizing the coordination mechanisms, Computers in Industry, Vol. 91, pp. 11-23,
Vasudeva K, Dikmese S, Güvenç S, Mehbodniya A, Saad W, Adachi F, 2017, Fuzzy-Based Game Theoretic Mobility Management for Energy Efficient Operation in HetNets, Special Section on Future Networks: Architectures, Protocols and Applications, Vol. 5, pp. 7542-7552.

Recibido: *13 de febrero de 2020*

Aceptado: *12 de julio de 2020*

Díaz, Francisco: *graduado de Ingeniero de Computación en el año 2004 en la UVM, M. Sc. en Informática Aplicada IUPTJE en el 2010 y estudiante del Doctorado en Ciencias de la Computación en la ULA, Correo electrónico:* franjdv@gmail.com

Aguilar, José: *graduado en Ingeniería de Sistemas en 1987 en la ULA, M. Sc. en Ciencias de la Computación en 1991 (Universite Paul Sabatier-Francia), Doctorado en Ciencia de la Computación en 1995 (Universite Rene Descartes-Francia). Postdoctorado en Ciencias de Computación en la Universidad de Houston (2000).*

Altamiranda, Junior: *graduado en Ingenieria de Sistemas en 2002, M. Sc. en Ciencias de la Computación en 2006 y Doctorado 2012, todos en la ULA. Correo electrónico:* altamira@ula.ve