A. J. Tomeu Hardasmal and A. G. Salguero, "Teaching Parallelism With Gamification in Cellular Automaton Environments" in IEEE-RITA, vol. 15, no. 1, pp. 34 - 42, FEB. 2020. doi: 10.1109/RITA.2020.2979067

Enseñando Paralelismo con Ludificación Ambientada en Autómatas Celulares

Antonio J. Tomeu y Alberto G. Salguero

***Title*— Teaching Parallel Programming with Gamification in a Cellular Automaton Environment.**

***Abstract*— Teaching of parallel programming to undergraduate computer science students can now be considered an inexcusable need. The current hardware platforms, which incorporate multiple cores and several execution threads, demand it. Notwithstanding the above, the teaching of parallelism through examples and classical algorithms, makes our students find it hard and complex. On the other hand, to teach through the techniques of gamification, has already demonstrated in a reliable way a positive reinforcement of the student in front of the learning of complex concepts. This work shows an experience developed by us to convey the teaching of parallelism to undergraduate students by gamification in microworlds. Results obtained by the students who followed this model, compared to a control group that followed the standard model, show a statistically significant advantage in favor of the teaching of parallelism, using a gamification with microworlds model.**

***Index Terms*— Cellular automaton, E-learning, gamification, microworlds, multicore programming, parallel programming.**

I. Introducción

La enseñanza de la programación concurrente y paralela se ha convertido, de unos años a esta parte, en una de las piedras angulares de la formación de los profesionales en tecnologías de la información y las comunicaciones (TICs). Según estimaciones de la Comisión Europea [10], desde ahora al año 2020 serán necesarios más de medio millón de especialistas en estas áreas, para satisfacer la demanda prevista por la industria. En consecuencia, garantizar que la formación de estos profesionales cumple con los plazos previstos por los respectivos títulos de grado, optimizando por tanto la tasa de graduación se antoja una necesidad. Sin embargo, esto está lejos de ser así, y el diferencial entre el número de años previstos para obtener el título y el número de años empleados para lograrlo es, desgraciadamente, muy amplio. Actualmente, en muchos países europeos, incluida España, el nivel de conocimientos de ciencias, y especialmente de matemáticas, ha disminuido drásticamente, si los comparamos con países asiáticos como Corea del Sur. Ello conduce a recibir a estudiantes de pregrado con pocas capacidades de abstracción, de dominio del lenguaje matemático, y de sistematización algorítmica de un problema. Con este estado de cosas, no es extraño que el aprendizaje de la programación en general [33] y de la programación paralela en particular, se conviertan para nuestros estudiantes en un auténtico *tour de force* [21], y que las tasas de éxito y rendimiento de las materias donde se enseñan sean relativamente bajas. Los ejemplos y casos prácticos que los profesores empleamos para vehicular nuestra materia tampoco ayudan; aunque son formativos y asequibles, son generalmente muy abstractos, y ni estimulan a los estudiantes a programarlos, ni una vez programados les dejan sensaciones de recompensa para abordar nuevos desarrollos.

Con el propósito de tratar de paliar la problemática expuesta, durante el curso 2017/2018, hemos desarrollado una experiencia docente que ha enseñado a los estudiantes a programar en paralelo mediante el uso de la ludificación, ambientada en micromundos. En lugar de los casos de estudio clásicos, hemos escogido otros basados en micromundos que por un lado estimulaban a los estudiantes a programarlos y por otro, una vez programados, les proporcionaban una sensación de recompensa lo suficientemente eficaz como para abordar con interés nuevos desarrollos, aumentado el compromiso de los estudiantes. Los objetivos perseguidos con la experiencia han sido:

- O1: medir el impacto de la ludificación en el rendimiento académico de los estudiantes.
- O2: medir el grado de esfuerzo de los estudiantes con el modelo propuesto en la experiencia, frente al modelo de uso habitual.
- O3: medir si la ludificación facilita el aprendizaje de la programación paralela multicore.

La experiencia se desarrolló con cuatro experimentos concretos representados a través de micromundos, y los estudiantes desarrollaron con ellos las prácticas de la asignatura durante el curso académico 2017-2018. Se empleó como grupo de control el grupo de estudiantes del año académico anterior, que desarrollaron las prácticas estándar, para medir el impacto del modelo que la experiencia propone. Posteriormente, se compararon los

Manuscrito recibido el día de mes de año; revisado día de mes de año; aceptado día de mes de año.

English version received Month, day-th, year. Revised Month, day-th, year. Accepted Month, day-th, year.

Antonio J. Tomeu y Alberto G. Salguero son miembros del Dtpo. de Ingeniería Infomática de la Universidad de Cádiz, Puerto Real, España.

E-mail: {antonio.tomeu, alberto.salguero}@uca.es

A. J. Tomeu Hardasmal and A. G. Salguero, "Teaching Parallelism With Gamification in Cellular Automaton Environments" in IEEE-RITA, vol. 15, no. 1, pp. 34 - 42, FEB. 2020. doi: 10.1109/RITA.2020.2979067

resultados obtenidos por los estudiantes de ambos grupos mediante medidas objetivas y subjetivas, a fin de contrastar si se habían alcanzado los objetivos propuestos. Los resultados obtenidos no dejan lugar a dudas, y muestran evidencia estadísticamente significativa a favor de enseñar programación paralela multicore mediante ludificación con micromundos.

II. Background

La ludificación se origina en la extinta Unión Soviética como incentivo alternativo a la retribución para hacer un trabajo, y desde hace algunos años se ha configurado como una estrategia novedosa en la práctica académica [2, 3, 7, 8, 14, 15, 16, 20, 26, 30,]. Su implementación práctica admite dos variantes: la primera es la escenificación, que asigna a los estudiantes roles (normalmente relacionados con su futura práctica profesional) y los hace interactuar entre sí; la segunda variante utiliza juegos donde el estudiante interactúa con ellos y aprende algo, o donde los estudiantes implementan el juego a partir de unas reglas dadas, siendo el proceso de implementación el que les lleva a aprender ese algo. Es esta última alternativa, de la que existen algunos precedentes recientes en la literatura para enseñar a programar [5, 11, 12, 13, 17, 19, 22, 24, 25, 27, 28, 29, 34], la que hemos escogido, ya que se adecúa singularmente bien a la enseñanza de la programación paralela. El uso de la ludificación en la enseñanza de la programación siempre ha perseguido como objetivos reforzar determinados aspectos del aprendizaje de la misma, o promover la participación activa de los estudiantes. Siendo un tópico relativamente novedoso, existen no obstante estudios [7] que aportan evidencia de cambios en el comportamiento de los estudiantes.

El concepto de micromundo se desarrolla en el ámbito de la Inteligencia Artificial y han sido definidos como "ambientes de aprendizaje interactivo sobre la base de computadores, donde los prerrequisitos están incorporados al sistema y donde los estudiantes se convierten en arquitectos activos de su propio aprendizaje". El uso de computadores facilita el análisis de ambientes simulados, la implementación de estrategias, y además introducen un componente de inmediatez y de interactividad que se muestran singularmente útiles para el estímulo a los estudiantes [8, 19].

No obstante lo anterior, hay otros elementos a tener en cuenta, cuya interacción produce sinergias que facilitan el proceso de aprendizaje, y que son los siguientes:

- Alumno: es el sujeto que aprende, en nuestro caso, programación paralela.
- Contexto: que define el conjunto de actividades definidas por el profesor para un micromundo dado, y que obligan a los estudiantes a planificar su implementación.
- Técnico: que define el conjunto de herramientas del lenguaje de programación que el estudiante debe utilizar para dar soporte a su implementación del micromundo, en nuestro caso orientado a la programación paralela *multicore*.

El uso de micromundos simulados se ha generalizado en multitud de entornos educativos, incluida la educación superior [16, 17], e incluso en los planes de formación y capacitación de empresa. Mediante la reproducción de situaciones del mundo real, se pretende una interacción usuario-plataforma donde la toma de decisiones guía a finales exitosos o de fracaso, que ofrecen al estudiante información de *feedback* sobre su interacción [2] de la que extraer conclusiones útiles a futuro. De hecho, existen en el mercado una gran variedad de plataformas de propósito específico que permiten aprender programación mediante ludificación a adolescentes (*Scracth*, desarrollado en el MIT), neurociencias a estudiantes de medicina (*Future-Learning-PFL*) o geometría dinámica a estudiantes de física y matemáticas (*Cabri-Géométre*). Por otra parte, y desde una perspectiva más lúdica, es conocido que cualquier adolescente o adulto joven pasa actualmente gran parte de su tiempo interactuando con videojuegos [35]. La inmediatez de la recompensa, junto con la interacción con los –a veces complejísimos- mundos virtuales que estos recrean, los hacen enormemente atractivos para ellos, puesto que obtienen un *feedback* positivo casi instantáneo.

III. Micromundos con Autómatas Celulares

Existen múltiples definiciones del concepto de autómata celular en la literatura. En muchísimos campos han sido empleados como micromundos para modelar realidades físicas de alta complejidad; la propagación de incendios forestales, la percolación de sustancias, la combinación de solutos de una reacción química o la simulación del tráfico urbano son solo algunos ejemplos. Nosotros escogemos la definición establecida con carácter general en [32], y aplicada a la simulación de micromundos. Definimos pues un autómata celular (AC) como una 4-upla $M = (\zeta, \varepsilon, N^I, \rho)$ donde:

- ζ es una red regular discreta de células (también denominados nodos o celdas) junto con algún conjunto de condiciones de frontera en el caso finito, que definen la vecindad de las células situadas en la periferia de la red; esta retícula conforma el micromundo.
- ε es un conjunto finito (habitualmente con estructura de anillo abeliano) de estados que las células de la red pueden adoptar.
- N^I es un conjunto finito de células que definen la vecindad con las que interactúa una célula dada, que habita en el micromundo.
- ρ es una función de transición que especifica cómo una célula de la red cambia de estado en función del tiempo y del estado de la vecindad N^I.

Dado lo anterior, un micromundo puede definirse como una red ζ incluida en el espacio real R^d que cubre de forma homogénea una porción del espacio euclídeo *d*-dimensional. Cada célula está etiquetada por su posición $r \in \zeta$. La disposición espacial de las células está especificada por las conexiones con sus vecinos más cercanos, las cuáles se obtienen uniendo pares de células en alguna disposición regular. Para cualquier coordenada espacial r, el retículo de vecindad $N_b(r)$ es una lista de células vecinas definidas por

$$N_b(r) = \{r + c_i : c_i \in N_b, i = 1, \cdots, b\} \quad (1)$$

Donde b es el número de coordinación o, dicho de otra forma, el número de vecinos próximos en el retículo que interactúan con la célula ubicada en la coordenada$r$. Con N_b denotamos a la plantilla de vecinos próximos con elementos $c_i \in R^d$, para $i = 1, \cdots, b$. En el caso que nos ocupa, y para $d = 2$, los únicos polígonos regulares que forman una teselación regular del plano son triángulos $(b = 3)$, rectángulos $(b = 4)$ y hexágonos $(b = 6)$, y nosotros escogeremos para nuestro modelo el segundo caso, de manera que

$$\zeta = \{\mathrm{r} : \mathrm{r} = (\mathrm{r}_1, \mathrm{r}_2) \in Z^2\} \quad (2)$$

El número total de células disponibles lo notaremos habitualmente por $|\zeta|$. En simulaciones de computador, los AC utilizan retículos que han de ser necesariamente finitos $(|\zeta| < \infty)$, y deben imponerse condiciones de frontera que establezcan cuáles son los vecinos de aquellas células situadas en las fronteras declaradas. En nuestro caso, utilizaremos la condición de frontera nula o cilíndrica, según el micromundo que se pretende simular. El conjunto de células vecinas cuyo estado influye en una dada, que viene definido mediante la vecindad de interacción $N_b^I(r)$ para una célula r dada, de acuerdo a la siguiente expresión:

$$\mathrm{N_b^I(r)} = \{\mathrm{r} + \mathrm{c_i} : \mathrm{c_i} \in \mathrm{N_b^I}\} \quad (3)$$

Esta vecindad o plantilla de interacción puede escogerse de varias formas, aunque nosotros nos decantaremos en nuestra simulación por la conocida como vecindad de *Moore.*

Por otra parte, cada célula $r \in \zeta$ tiene asignado un estado $s(r) \in \varepsilon$. Los elementos del conjunto ε pueden ser números, letras o símbolos. Nosotros escogeremos ε en función del ambiente que se pretenda modelar con el micromundo. Una configuración global del autómata $s \in \varepsilon^{|\zeta|}$ queda determinada por el estado de todas las células del retículo y nuestro modelo ofrece una instantánea del estado del tumor en un instante dado del tiempo, y cambia dinámicamente ese estado utilizando una secuencia temporal discreta de acuerdo a la regla que la función de transición impone.

Finalmente, esa dinámica de evolución temporal del modelo viene determinada por la función de transición ρ, que especifica cómo una célula cambia de estado en función de su estado anterior, y de la interacción de la misma con su vecindad de células, de acuerdo a la ecuación 4.

$$\rho : \varepsilon^{\mu} \rightarrow \varepsilon \quad (4)$$

donde $\mu = |N_b^I|$. La regla es espacialmente homogénea, y no depende por tanto en forma explícita de la posición r de una célula dada.

IV. Casos de Estudio

Describimos en esta sección el conjunto de proyectos de programación paralela basados en micromundos que hemos propuesto a nuestros estudiantes como parte de las prácticas de la asignatura en la que hemos desarrollado la experiencia. La descripción se hará con un nivel de detalle lo suficientemente amplio como para que la experiencia pueda ser reproducida a partir del texto. En todos los casos, el esquema de trabajo seguido por los estudiantes es el que se ilustra en la Fig. 1.

Tabla 1
Objetivos de Aprendizaje Para Cada Micromundo

Objetivo	Life	Wa-Tor	Tumor	Caves
Clase `Thread`	Sí	No	No	No
Tareas `Runnable`	No	Sí	Sí	Sí
Clase `Random`	Sí	Sí	Sí	Sí
Partición de Datos[1]	Sí	No	No	No
Partición de Datos[2]	No	Sí	Sí	Sí
Mutex	No	No	Sí	No
Barrera	Sí	Sí	Sí	Sí
Pool de hebras	No	Sí	Sí	Sí
GUI	Sí	Sí	Sí	Sí
Medidas de tiempo	No	No	Sí	Sí
Speedup	No	No	Sí	Sí

El profesor desempeña una labor previa donde expone a los estudiantes la entidad a modelar, para posteriormente extraer un modelo matemático continuo, y finaliza proponiendo a los estudiantes un modelo matemático discreto utilizando un autómata celular bidimensional. Los estudiantes, utilizando el lenguaje Java, implementan el autómata celular mediante un código secuencial, y luego obtienen una versión paralela, aplicando el esquema división de datos que describe la Fig. 2, junto con diferentes elementos adicionales de control del paralelismo que se muestran en la Tabla 1.

Esta tabla recoge en columna, para cada uno de los micromundos, el conjunto de tópicos de programación paralela que se utilizarán en su implementación. Así, por ejemplo, para implementar *Life*, los estudiantes utilizan tareas paralelas mediante el uso de la clase `Thread`, pero no mediante la interfaz `Runnable`, generaran datos aleatorios usando la clase `Random`, efectuán una división del espacio de datos manual y no automatizada, no emplean control de exclusión mutua, ni tampoco un ejecutor de *pool* de hebras para procesar las tareas, aunque sí las sincronizan mediante la clase `Barrier`; finalmente implementan una interfaz gráfica, y no toman tiempos de ejecución ni calculan el speedup.

Las simulaciones de los micromundos no son interactivas, excepto la simulación del planeta *Wa-Tor*, donde el estudiante puede modificar los parámetros de la simulación durante el transcurso de la misma. Cada uno de los micromundos propuestos en la experiencia aumenta la dificultad de implementación respecto al precedente, añadiendo nuevas complejidades, utilizando nuevas clase del API de control de la concurrencia y, en general, incrementado de forma gradual el aprendizaje de conceptos

[1]División manual del dominio de datos vía constructor de clase.
[2]División automática del dominio de datos con el método `availableProcessors()`.

más complejos, y reforzando por repetición aquellos que se consideran más importantes.

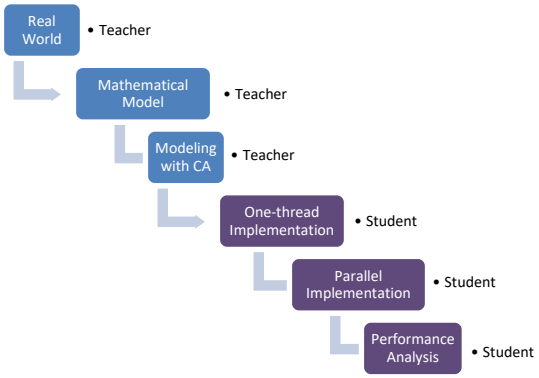

Fig. 1. Ciclo de trabajo para cada micromundo, con los roles desarrollados por el profesor y por los estudiantes.

Los conceptos de programación paralela *multicore* que hemos pretendido enseñar a los estudiantes han sido: técnicas de creación de tareas paralelas mediante herencia de la clase `Thread` e implementación de la interfaz `Runnable`, partición del dominio de datos (Fig. 2) manual y automática, control de exclusión mutua sobre los datos cuando es necesario, uso de ejecutores, uso de barreras, medidas de tiempo de ejecución, cálculo del *speedup* y representación mediante una interfaz gráfica del micromundo.

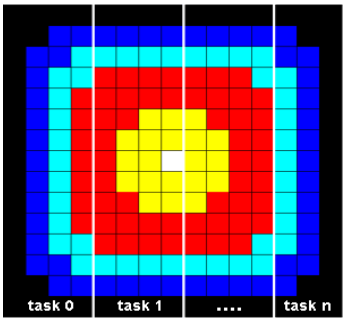

Fig. 2. Esquema de división del espacio de datos del micromundo entre un número *n* de tareas paralelas. Los estudiantes realizaron la división de forma manual y automática.

A. *El Juego de la Vida (Life)*

Propuesto en [6] por el matemático *J.H. Conway*, se dispone de un micromundo que simula la interacción entre una población de células que pueden estar un estado $a_{i,j} \in \{0,1\}$. La función de transición que los estudiantes deben aplicar a cada célula del micromundo puede ser descrita mediante la ecuación (5). Se pidió a los estudiantes que efectuaran una simulación paralela de *Life*, cumpliendo los objetivos planteados en la Tabla 1.

En esta fase de la experiencia, los estudiantes desarrollaron código paralelo con partición manual del dominio de datos. Se puso especial énfasis también en el diseño de una interfaz de usuario que permitiese obtener una visualización gráfica de la simulación, tal y como muestra la Fig. 3, donde cada generación mostrada debía haber sido calculada mediante hebras paralelas.

TABLA 2
PARÁMETROS EN LAS ECUACIONES DE LOTKA-VOLTERRA

Parámetro	Significado
α	Tasa de crecimiento de los peces
β	Éxito de los tiburones cazando
γ	Tasa de decrecimiento de los tiburones
δ	Alimento obtenido por los tiburones

$$\rho\left(a_{i,j}^{(t+1)}\right) = \begin{cases} 1, if\ a_{i,j}^{(t)} = 0\ y\ \sum_{i=-1,j=-1}^{1,1} a_{i,j}^{(t)} = 3 \\ 1, if\ a_{i,j}^{(t)} = 1\ y\ \sum_{i=-1,j=-1}^{1,1} a_{i,j}^{(t)} < 3 \\ 0, if\ a_{i,j}^{(t)} = 1\ y\ \sum_{i=-1,j=-1}^{1,1} a_{i,j}^{(t)} > 3 \end{cases} \quad (5)$$

B. *Tiburones y Peces en el Plantea Wa-Tor*

Propuesto en [9] por *A.K. Dewdeney*, *Wa-Tor* es un mundo acuático reticulado de estructura toroidal, donde habitan dos especies: tiburones y peces. Cada punto de la retícula de *Wa-Tor* $a_{i,j} \in \{0,1,-1\}$. Los peces (-1) y tiburones (1) siguen la dinámica clásica predador-presa descrita en la referencia [9] por las clásicas ecuaciones de *Lotka-Volterra* (ecuaciones 6 y 7).

$$\frac{dx}{dt} = \alpha x - \beta xy \quad (6)$$

$$\frac{dx}{dt} = -\gamma y + \delta yx \quad (7)$$

Aquí, x es es el número de peces, y es el número de tiburones, y los coeficientes tienen el significado que se muestra en la Tabla 2. Una vez interpretadas en forma discreta [23], describen un micromundo en el que inicialmente el 50% de los puntos de la retícula contienen peces, (-1) un 25% contienen tiburones (1) y el resto están vacías (0). Los estudiantes deben implementar una simulación del planeta *Wa-Tor* (ver Fig. 4), donde los objetivos de aprendizaje son los que se describen en la Tabla 1.

Los estudiantes implementaron la simulación paralela añadiendo varias mejoras respecto al micromundo anterior, siendo la más destacable el procesamiento de las tareas paralelas mediante la delegación de su ciclo de vida a un ejecutor de *pool* de hebras.

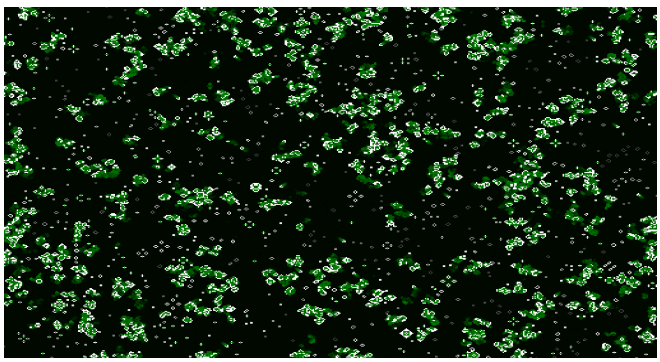

Fig. 3: *Life*. Se ilustra el estado del micromundo después de algunas generaciones de evolución a partir de una configuración inicial aleatoria. Se observa la formación de colonias de células vivas.

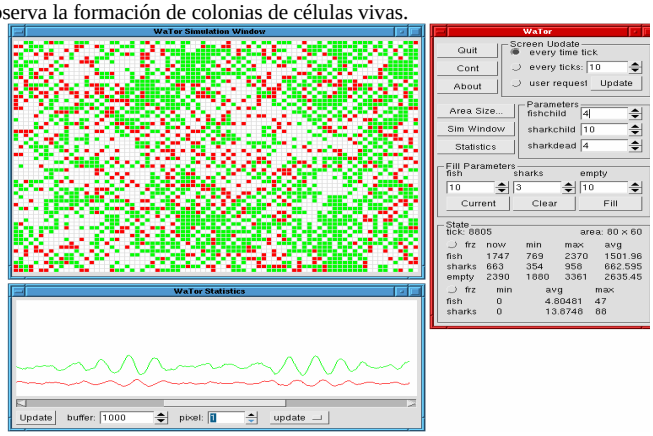

Fig. 4: El micromundo de *Wa-Tor*. Se ilustra el estado del micromundo (cuadrícula superior izquierda), las curvas de población que siguen la dinámica descrita por las ecuaciones de *Lotka-Volterra*, y la ventana (situada en la zona derecha de la imagen) que permite ajustar los parámetros de la simulación.

De esta forma, se centraron más en la codificación de la razón de ser de las tareas, y menos en la gestión de su ciclo de vida, que fue desarrollado de forma automatizada por el *pool* de hebras.

C. Simulación de Crecimiento Tumoral

Una célula tumoral en este micromundo es una entidad individual que ocupa un punto de una red bidimensional finita ζ, y que puede (ver [32]) realizar las siguientes acciones: migrar a otro punto de la red, proliferar mediante mitosis, morir, o permanecer estable en la fase G_0 del ciclo celular, donde las células están quiescentes. Las células tumorales habitualmente son de tipo *stem* (con capacidad de proliferación ilimitada) y *no-stem* (pueden efectuar una mitosis en la fase M (mitosis) del ciclo celular un número limitado de veces). Nosotros no supondremos células *stem*, pero en cambio modelaremos si una célula vive o no mediante la distribución de probabilidad de la ecuación 8.

$$\rho_l(s_{N(r)}) = \begin{cases} 1 \ \ con\ probabilidad\ W(s_{N(r)} \to 1) \\ 0 \ \ con\ probabilidad\ W(s_{N(r)} \to 0) \end{cases} \quad (8)$$

Los estudiantes tuvieron que modelar esta distribución y la descrita por la ecuación 9 utilizando como herramienta de generación de números aleatorios la clase `Random` de Java, que permite, a través de múltiples instancias, generarlos de forma paralela.

El ajuste de esta distribución permite tener, si se desea, también células *stem*, y lo que es más importante, permite modelar la supervivencia de las células ocasionada por factores intrínsecos al tumor (angiogénesis, microambiente tumoral, etc.) o a factores extrínsecos de tipo terapéutico (respuesta a citostáticos o citotóxicos, uso de fármacos antiangiogénicos, radiación, interferones, etc.) ajustando la distribución en función de la respuesta tumoral conocida ya sea *in vivo* o *in vitro*, y contrastada en la práctica clínica o de laboratorio.

Una célula que está viva puede proliferar generando una célula hija mediante mitosis siempre que haya espacio disponible para ello en su vecindad, fenómeno que el modelo contempla mediante una segunda distribución de probabilidad, condicionada a la anterior, como describe la ecuación 9.

$$\rho_p(s_{N(v)}) = \begin{cases} 1, \ with\ probability\ W'(s_{N(v)} \to 1) \\ 0, \ with\ probability\ W'(s_{N(v)} \to 0) \end{cases} \quad (9)$$

Ahora, la posición de la red $v \in s_{N(r)}$ albergará a la célula resultante de la mitosis de la célula r, si hay espacio para albergarla. Finalmente, una célula viva puede migrar, cambiando de posición dentro del tumor, siempre que haya espacio disponible para ello. Esto se ha contemplado mediante una tercera distribución de probabilidad, también condicionada a ρ_l.

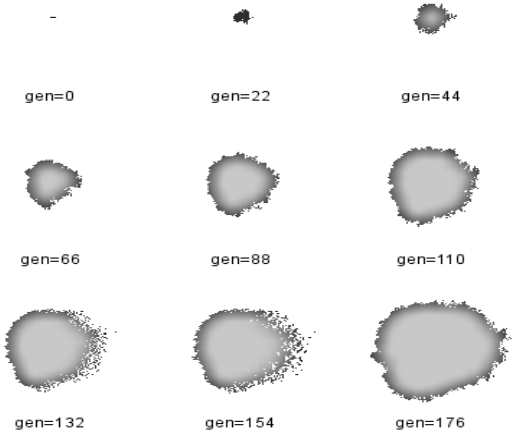

Fig. 5. Evolución del modelo de simulación tumoral para un dominio tisular de 128×128 células y 200 generaciones. Se muestran instantáneas del estado de la tumoración a partir de una semilla inicial de cuatro células en diferentes instantes del tiempo, que aparecen indicados mediantes el número de generación en que se tomó cada una de las instantáneas.

$$\rho_m(s_{N(v)}) = \begin{cases} 1, \ with\ probability\ W''(s_{N(v)} \to 1) \\ 0, \ with\ probability\ W''(s_{N(v)} \to 0) \end{cases} \quad (10)$$

El modelo está parametrizado por las distribuciones de probabilidad descritas en el apartado anterior, que permiten mediante simulación estocástica de *Monte-Carlo* decidir si una célula muere, permanece en la fase G_0 del ciclo, o bien entra en la fase M del mismo y se reproduce por mitosis, siempre que tenga espacio suficiente alrededor para hacerlo. La dirección de propagación de la mitosis se decide mediante una distribución de probabilidad que escoge una de cuatro direcciones posibles.

En esta ocasión, Los estudiantes recibieron una implementación en pseudocódigo que simulaba el crecimiento tumoral (ver la Fig. 5) mediante un código secuencial [4] y un patrón de tipo condicional (pre, post) tal y como se describe en la referencia [4] y debían implementar una simulación paralela, donde los objetivos de

aprendizaje fueron los indicados en la Tabla 1 para este micromundo.

Fig. 6. Micromundo Cave en forma de laberinto resultante de aplicar la simulación paralela del autómata celular descrito en el texto.

D. Generación de Cavernas (Cave)

En muchos videojuegos, es habitual el uso de un entorno con estructura de laberinto (Fig. 6), por el cual se mueven diversos actores (el paradigma de este tipo de videojuegos es el venerable *Pac Man*). Existen propuestas en la literatura [18] que utilizan autómatas celulares bidimensionales para generar un laberinto que cambia en cada nueva partida. La retícula que modela a este micromundo dispone de células que toman valores de la forma $a_{i,j} \in \{floor,\ wall,\ rock\}$. Inicialmente la retícula se llena de células con estado $a_{i,j} = floor$, y a continuación se utiliza un generador de números aleatorios uniformemente distribuidos para convertir el estado del $r\%$ de las células a $rock$. Posteriormente, la retícula es procesada en paralelo. En cada generación, se aplica a todas las células de la retícula la función de transición de la ecuación (11).

$$\rho\left(a_{i,j}^{(t+1)}\right)\begin{cases} rock\ if\ a_{i,j}^{(t)} = floor\ and\ \sum_{i,j=-1}^{1} a_{i,j}^{(t)} \geq T \\ a_{i,j}^{(t)}\ if\ a_{i,j}^{(t)} = floor\ and\ \sum_{i,j=-1}^{1} a_{i,j}^{(t)} < T \end{cases} \quad (11)$$

$$\rho\left(a_{i,j}^{(t+1)}\right) = \begin{cases} wall\ if\ a_{i,j}^{(t)} = rock\ and\ N_f \geq 1 \\ a_{i,j}^{(t)}\ if\ N_f < 1 \end{cases} \quad (12)$$

Los parámetros T y r son fijados por el usuario en función de la complejidad final deseada para el laberinto, que en la aplicación práctica de este modelo, en el ámbito de los videojuegos, depende del nivel en que está el jugador. En general, a mayor nivel, la complejidad aumenta. Una vez que el laberinto ha sido construido tras n iteraciones, llega el momento de construir las paredes, para lo cual se aplica la función de transición de la ecuación 12 únicamente sobre aquellas células que son de clase $rock$. En la ecuación 12, N_f es el número de células vecinas que son de clase $floor$. Naturalmente, el método propuesto aqui admitiría –y de hecho exigiría- añadir el procesamiento necesario para garantizar que el laberinto generado es viable. Esto es, sería necesario añadir una rutina que busque caminos válidos en el laberinto generado, y lo descarte si no los hay. Tal procedimiento está perfectamente caracterizado en la literatura [14] y es de fácil implementación, aunque no fue planteado a nuestros estudiantes.

TABLA 3
CONTENIDOS DEL CURSO TEÓRICO (SEGÚN GUÍAS ACM/IEEE)

Tópico	Tiempo (horas)
Computaciones simultáneas multiples	8
Paralelismo versus concurrencia	4
Comunicación y coordinación	10
Condiciones de Carrera	4
Vivacidad y progresividad	4

V. CONTEXTO EXPERIMENTAL

Para desarrollar el experimento se escogió como contexto la asignatura “Programación Paralela y Distribuida”, cursada por los estudiantes del grado en Ingeniería Informática de la *Universidad de Cádiz* (España), en el quinto semestre del grado, con una carga académica de 6 créditos ECTS, 3 créditos teóricos y 3 créditos prácticos. Los estudiantes accedieron a la asignatura habiendo cursado previamente “Programación Concurrente y de Tiempo Real”, y poseían conocimientos elementales acerca de la creación, gestión y control de hebras concurrentes con el lenguaje Java. Sin embargo, no poseían conocimientos de programación paralela multicore. Tampoco tenían experiencia previa con el aprendizaje de la programación con micromundos. Las 30 horas de prácticas se dividieron en cuatro bloques de 7.5 horas, durante cada uno de los cuáles los estudiantes desarrollaron un miniproyecto de programación, con cada uno de los micromundos descritos, siguiendo el diagrama de trabajo propuesto en la Figura 1. También tuvieron que dedicar tiempo de trabajo personal en casa a cada uno de ellos. Las 30 horas de teoría se configuraron según las recomendaciones de la guía curricular de ACM/IEEE [1], de acuerdo a lo descrito para el curso *PD/Parallelism Fundamental* según se ilustra en la Tabla 3.

Se inscribieron en el curso 52 estudiantes (42 hombres y 10 mujeres), con una edad media de 21.84 ± 0.63 años, que siguieron las prácticas mediante los microproyectos de programación descritos en la sección IV. Abandonaron las prácticas de la asignatura 4 estudiantes y no la superaron (incluyendo los abandonos) 8 estudiantes. Como grupo de control, se utilizó a los estudiantes matriculados (46 estudiantes, 37 hombres y 9 mujeres), con una edad media de 22.35 ± 0.46 años, en la asignatura en el anterior curso académico, que siguieron las prácticas mediante el modelo estándar implementado algoritmos clásicos de paralelismo (producto paralelo de matrices, modelo *worker-slave*, etc.) Abandonaron 8 estudiantes. Al inicio del semestre los estudiantes completaron un cuestionario donde se medían algunas variables demográficas, como edad y sexo. Concluido el semestre se obtuvieron las calificaciones de prácticas de los estudiantes del grupo experimental. Los principales descriptores de la experiencia, una vez excluidos los abandonos se muestran en la Tabla 4. Se tenían disponibles esas calificaciones para los estudiantes del año académico previo, que actuaron como grupo de control.

Adicionalmente, utilizando como soporte la plataforma digital de la asignatura sobre *Moodle*, los estudiantes

TABLA 4

VARIABLES PRINCIPALES DEL EXPERIMENTO

Variable	Experimental	Control
N (excluidos abandonos)	48	38
Sexo (M-F)	39-9	30-8
Abandono (%)	7.69%	13.6%
Calificación media	7.03±1.88	6.31±1.62
Suspensos (%)	18.18%	21.05%

completaron un cuestionario cuyas variables recoge la Tabla 5. Para medir las bondades del modelo de enseñanza del paralelismo y el grado de consecución de los objetivos de la experiencia, se han utilizado tres escenarios prospectivos, que se describen a continuación:

- Medida de la calificación media obtenida por el grupo de estudiantes en las prácticas de la asignatura desarrollada con micromundos, representada por un valor numérico en el intervalo [0-10], comparada con el mismo valor obtenida por los estudiantes en el grupo de control.
- Medida del grado de esfuerzo de los estudiantes con el modelo de prácticas propuesto, versus el modelo estándar, medido en horas de dedicación a las prácticas de la asignatura (en clase y en casa).
- Medida de la impresión subjetiva de los estudiantes sobre su percepción personal del aprendizaje de la programación paralela, medida a través de un instrumento con forma de encuesta de múltiples ítems, y valoración en el intervalo [0-6].

VI. ANÁLISIS Y DISCUSIÓN

Los resultados obtenidos en el primero de los escenarios de medida (veáse la Fig. 7) mostraron cómo los estudiantes que siguieron el modelo de prácticas basado en la ludificación mediante micromundos obtuvieron una calificación media de prácticas de 7.02 ± 1.88 puntos, frente a los estudiantes del grupo de control, que obtuvieron una calificación media de 6.31 ± 1.62, siendo el suspenso una calificación inferior a 5.0 puntos en ambos casos. El porcentaje de estudiantes que no superaron las prácticas de la asignatura fue del 18.8% en el grupo experimental, frente al 21.05% del grupo de control. Se aplicó el test de *Shapiro-Wilk* a las muestras de datos de ambos grupos para la variable de calificación media.

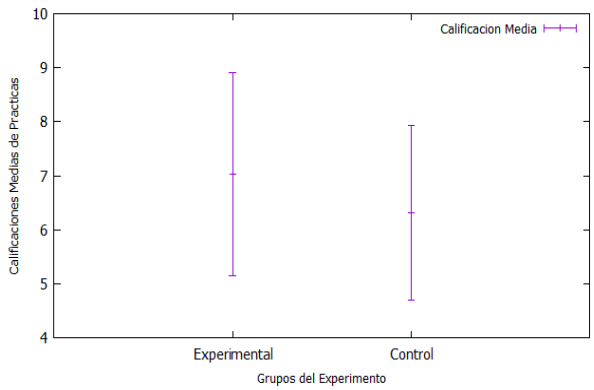

Fig. 7: Calificaciones medias junto con la desviación estándar en los grupos experimental y de control.

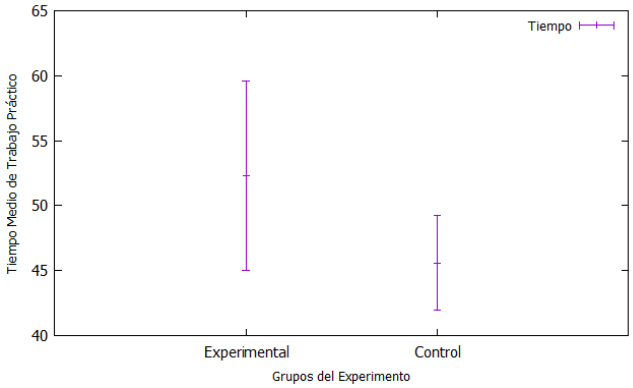

Fig. 8: Tiempo medio y desviación estándar de horas de trabajo de prácticas en grupos experimental y de control.

Ni el grupo experimental ($W = 0.8545,\ p < 0.001$) ni el de control ($W = 0.8880,\ p < 0.001$) eran normales para esta variable, con $\alpha = 0.05$. Atribuyendo al azar la diferencia de calificaciones entre ambos grupos de estudiantes (H_0), el estadístico U de *Mann-Withney* ($U = 434.500, p = 0.033$) mostró que esa diferencia no se debe al azar, y es atribuible al método propuesto de aprendizaje del paralelismo. Se observa pues una mejora estadísticamente significativa de los resultados medios de los estudiantes en el desarrollo de las prácticas de la asignatura y que muestra que la experiencia cumplió el objetivo O1. Para el segundo escenario y objetivo O2, se pidió a los estudiantes que llevaran una bitácora del total de tiempo dedicado al desarrollo de las prácticas de la asignatura, en el centro académico y en casa.

Los resultados medios obtenidos se ilustran en la Fig. 8. Se aprecia que el modelo propuesto necesita para ser desarrollado un número total de horas (52.3 ± 7.32) de dedicación discretamente superior al tiempo necesario cuando se utiliza el modelo estándar (45.6 ± 3.66). Se aplicó el test de normalidad *Shapiro-Wilk* a la distribución en horas de trabajo de los estudiantes de los grupos experimental y de control. Ni el grupo experimental ($W = 0.7963,\ p < 0.001$) ni el de control ($W = 0.8125,\ p < 0.001$) eran normales para esta variable, con $\alpha = 0.05$. Atribuyendo al azar la diferencia en horas de trabajo de los estudiantes(H_0), el estadístico U de *Mann-Withney* ($U = 346.272, p = 0.029$) mostró que esa diferencia no se debe al azar, y que el esfuerzo adicional requerido es atribuible al método propuesto de aprendizaje del paralelismo. Para el tercer escenario, se utilizó una encuesta ($n = 48$) de múltiples variables (Tabla 5) con la que se midió la impresión subjetiva de los estudiantes para estudiar el objetivo O3, que fue realizada por estos al finalizar el semestre de forma anónima, y se midieron, en una escala discreta de valores entre 1 (para nada de acuerdo) y 5 (para muy de acuerdo); el valor de 0 se utilizó para representar no sabe/no contesta. Los resultados de la encuesta se representan gráficamente en la Figura 9 que muestra, para cada una de las variables que miden el aprendizaje de aspectos relacionados con la programación paralela (q1 a q7), cuántos estudiantes la puntuaron en cada una de las seis posibles categorías de puntuación (intervalo 0-5). Se aprecia

como en la práctica totalidad de las variables citadas, los alumnos los puntúan principalmente con valores situados en

TABLA 5
VARIABLES SUBJETIVAS MEDIDAS

Variable	Significado
q1	Sé crear hebras por herencia de la clase *Thread*
q2	Sé crear tareas implementado la interfaz *Runnable*
q3	Se dividir el dominio de datos manualmente
q4	Sé dividir el dominio de datos automáticamente
q5	Sé sincronizar con un protocolo de barrera cíclica
q6	Sé controlar el acceso a datos bajo exclusión mutua
q7	Sé utilizar un ejecutor de pool de threads
q8	Los micromundos me estimular a hacer las prácticas
q9	Prefiero programar micromundos en las prácticas

la parte alta de la escala de valoración (valores 4 y 5), lo cual aporta evidencia de que la impresión subjetiva sobre su aprendizaje de tópicos clásicos de la programación paralela es razonablemente bueno.

Adicionalmente, la Fig. 10 muestra las puntuaciones subjetivas de las variables q8 y q9, que miden si los estudiantes se han sentido estimulados a comprometerse con las prácticas de la asignatura, en el sentido propuesto en [20], y su satisfacción global con la experiencia. En ambos casos los resultados ofrecen valoraciones altas que indican que el estímulo ha existido (q8), y que su satisfacción global (q9) con la experiencia es alta.

VII. CONCLUSIONES Y TRABAJO FUTURO

El análisis de las medidas realizadas en el apartado anterior ofrecen una validez estadística lo suficientemente significativa como para considerar viable el modelo propuesto, en función de los objetivos previstos. Recordemos que estos se concretaban en lograr llevar a los estudiantes al aprendizaje de un cuerpo mínimo de conocimientos lo bastante amplio y profundo como para afrontar el desarrollo de soluciones paralelas en el ámbito de las plataformas *multicore*, utilizando el lenguaje de programación Java, con un grado de esfuerzo razonable, y aumentando el grado de compromiso del estudiante con su aprendizaje. La discusión de los resultados ha demostrado que todo ello es posible. Concretamente, se ha logrado medir un impacto positivo de la experiencia en las calificaciones medias que los estudiantes obtienen en las prácticas de la asignatura

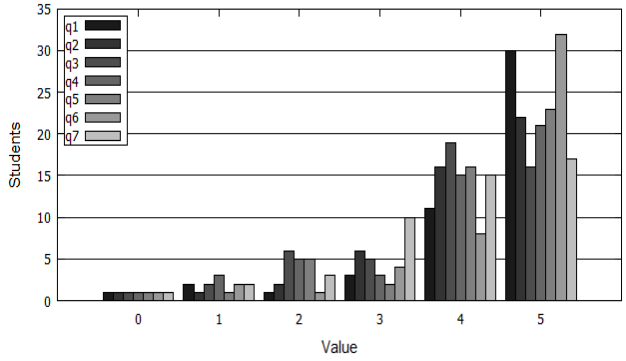

Fig. 9: Resultados de la evaluación del modelo en función de las variables subjetivas q1 a q7 listadas en la Tabla 5.

La encuesta subjetiva que ha medido la consecución del aprendizaje de diferente aspectos de la programación paralela *multicore* (Fig. 9) confirma que los estudiantes puntúan esa consecución de forma "alta" o "muy alta" mayoritariamente. Otro de los logros de la experiencia ha sido un mayor estímulo entre los estudiantes a la hora de comprometerse con el desarrollo de las prácticas de la asignatura, que se concreta como ya hemos en una calificación media superior (7.02 ± 1.88 vs. 6.31 ± 1.62) para los estudiantes que programaron micromundos (grupo experimental) frente a los que no lo hicieron (grupo de control), con una discreta mejora en la tasa de alumnos que califican con suspenso. El refuerzo que los estudiantes recibieron para comprometerse con el desarrollo de las prácticas, medido con el ítem q8 (Fig. 10) muestra de nuevo altas puntuaciones sobre ese compromiso.

Creemos que esta mejora global en los resultados obtenidos obedece principalmente al factor novedoso que la ludificación introduce por sí misma, y al atractivo que lograr implementar los micromundos y verlos en acción ofrece a los estudiantes. Por ambas vías, obtienen un estímulo en forma de recompensa visual que con otros enfoques de prácticas quizás no tienen, o tienen en menor medida; ese estímulo actúa como refuerzo positivo en el compromiso del estudiante con su aprendizaje, y mejora su actitud general hacia el trabajo que tiene que hacer. El principal aspecto negativo que identificamos es la mayor dedicación a las prácticas de la asignatura que los estudiantes necesitan (Fig. 8), aunque cabe señalar que esta siempre se ha mantenido dentro del número de créditos ECTS que la asignatura tiene asignados.

Como trabajo futuro nos planteamos la extensión del modelo a micromundos soportados por arquitecturas *manycore* de tipología GPU y la aplicación del modelo en el desarrollo de las prácticas de la asignatura "Programación Concurrente y de Tiempo Real".

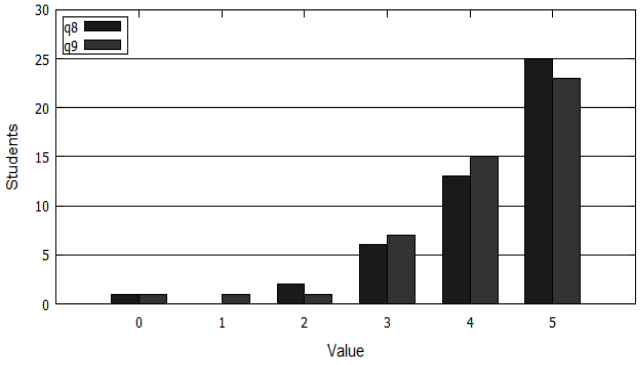

Fig. 10: Resultados de la evaluación del modelo en función de las variables subjetivas q8 y q9 listadas en la Tabla 5.

REFERENCIAS

[1] ACM/IEEE. "Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science". *ACM*, New York, USA. 2013.

[2] E. Buzek. "Web Platforms for Parallel Programming Tutorials". Digitalní Repositár UK. 2017.

[3] Y. Chou "Actionable gamification: Beyond points badges and leaderboards". Fremont, CA, USA, Octalysis Media, 2016.

[4] M.I. Capel, A.J. Tomeu and A.G. Salguero. "Teaching concurrent and parallel programming by patterns: An interactive ICT approcha". Journal of Parallel and Distributed Computing, Vol. 105, July 2017, pp. 42-52. 2017. DOI: 10.1016/j.jpdc.2017.01.010

[5] S. Combefis et al. "Learning Programming Through Games and Contest: Overview". Journal of Parallel and Distributed Computing, vol. 105, pp. 42-52, 2017. DOI: 10.1016/j.jpdc.2017.01.010

[6] J.H. Conway. "Mathematical Games: The fantastic combination of John Conway`s new solitaire game life". *Scientific American*, issue 223, pp. 120-123. 1970

[7] E.L. Deci, R. Koestner and R. M. Ryan."Extrincic rewards and intrinsic motivation in educations: Reconsiderd once again". *Rev. Edu. Res*, vol. 71, pp.1-27, 201

[8] P. Denny. "The effect of virtual achievements on student engagement". *Proc. SIGCHI Conf. Human Factors Comput. Syst*, pp. 763-772, 2013

[9] A.K. Dewdney. "Computers Recreations Sharks and Fish Wage an Ecological War on the Toroidal Planet Wa-Tor". *Scientific American*, pp. 14-22, December 1984.

[10] P. Díaz. "Faltan 900.000 profesionales TIC". Diario El Mundo, edición digital, 31 de Mayo de 2015

[11] P. Fotaris, T. Mastoras, R. Leinfellner and Y. Rosunally. "Who wants to be a phytonista: using gamification to teach computer programming". 2015. http://www.academia.edu/download/44361560/Who_wants_to_be_a_Pythonista_Using_Gamif.pdf.

[12] F. Gallego, C. Villagrá, F. Llorens and R. Carmona. "PLMan: a game-based learning activity for teaching logic thinking and programming". *International Journal of Engineering Education*, 33(2B), pp. 804-815, 2017.

[13] F. Gallego, C. Villagrá, F. Llorens and R. Carmona. "PLMan: a game-based learning activity for teaching logic thinking and programming". *International Journal of Engineering Education*, 33(2B), pp. 804-815, 2017.

[14] S. Gunuc. And A. Kuzu. "Student engagement scale: development, reliability and validity" *Assessment & Evaluation in Higher Education*, Vol. 40, No. 4, pp 587-610. 2015.

[15] Hamari, J. Koivisto and H. Sarsa. "Does Gamification Work? A literature review of empirical studies of gamification". *Proc. 47th Hawai Int. Conf. Syst. Scie.* Pp 1-10, 2011.

[16] W. Hsin et al. "A Practitioners Guide to Gamification of Education". *Research Report Series. Behavioural Economics in Action*. Rothman School of Management of Toronto, 2013.

[17] M. Ibáñez, A. Di-Serio and C. Delgado-Kloos. Gamification for engaging "Computer Science students in Learning Activities: A case study". *IEEE Transactions on Learning Technologies*, Vol. 7, Issue 3, pp. 291-pp. 291-301, 2014.

[18] L. Johnson, G.Y. Yannkakis and J. Togelius. "Cellular automata for real-time generation of infinite cave levels". *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. Article nº 10. 2010. DOI 10.1145/1814256.1814266

[19] A. Knuttas et al. " Increasing collaborative communications in programming course with gamification: a case study". *Proceedings of 15th International Conference on Computer Systems and Technologies.* pp. 370-377, 2014.

[20] M. Mallon. "Gaming and Gamification". *Public Services Quarterly*. Vol. 2, issue 3, pp. 210-221, 2013. DOI: 10.1080/15228959.2013.815502

[21] P.E. McKenney, M. Gupta, M.M. Michael, P. Howard, J. Triplett and J. Walpole. "Is Parallel Programming Hard, And If So, Why?". *Cornell University Library*, 2017. In https://arxiv.org/abs/1701.00854.

[22] P. Mozellius, T. Florica, O. Shabalina, C. Miller, C. Malliarakis, C. Balan and S. Satyadhyan. "Game-Based Technologies in Teaching Programming in Higher Education: Theory and Practices". *Recent Patents on Computer Science*, Vol. 9. Issue 2, pp. 105-113, 2016.

[23] S. Olek. "An Accurate Solution to the Multispecies Lotka-Volterra Equations". *Society for Industrial and Applied Mathematics (SIAM)*, Vol. 36, issue 3, pp.480-488, 1994.

[24] N. Paspallis.. "A Gamification Platform for Inspiring Young Students to Take an Interest in Coding." In V. Strahonja, N. Vrček., D. Plantak Vukovac, C. Barry, M. Lang, H. Linger, & C. Schneider (Eds.), *Information Systems Development: Transforming Organisations and Society through Information Systems (ISD2014 Proceedings). 2014* ISBN:978-953-6071-43-2.

[25] M. Ortiz-Rojas, K. Chiluiza and M. Valcke . "Gamification in Computer Programming: Effects on Learning, Engagement, Self-Efficacy and Intrinsic Motivation". *Proceeding of European Conference on Games Based Learning.* pp. 507-514. 2017.

[26] M. Ortiz, K. Chiluiza and M. Valcke. "Gamification in higher education and STEM: A systematic review of literature". *In 8th International Conference on Education and New Learning Technologies*, pp. 6548 – 6558, 2016.

[27] C.R. Prause and M. Jarke. "Gamification for enforcing coding conventions". *Proceedings of 10th Joint Meeting on Foundations of Software Ingineering".* pp. 649-660. DOI: 10.1145/2786805.2786806

[28] J. Rojas and G. Fraser. "Teaching mutation test using gamification". *Proceeding of the European Conference Software, 2016.*

[29] N. Sahari, T.S. Meriam, T. Wook and A. Ismail. "The study of gamification application architecture for programming language course". *Proceeding of the 9th International Conference on Ubiquitous Infomation Managment and Communication.* Article nª 17, 2015.

[30] M. Sanmugam and Z. Abdullah. "Gamification: Cognitive Impact and Creating a Meanningful Experience in Learning". *IEEE 6th International Conference on Engineering Education*. 2014

[31] J. Simoes, R. Díaz and A. Fernandez. "A social gamification framework for a K-6 learning platform". *Computers in Human Behavior*, Vol. 29, issue 2, pp. 345-353, 2013

[32] A.J. Tomeu, A.G, Salguero and. M.I. Capel."Speeding Up Tumor Growth Simulations Using Parallel Programming with Cellular Automata". *IEEE Latin America Trans*, vol. 14, issue 11, pp. 4611-4619, 2016. DOI: 10.1109/TLA. 2016.7795837

[33] Watson, C. and Li, F. W. (2014). "Failure rates in introductory pogramming revisited". In Proceedings of the 2014 conference on Innovation & technology in computer science education, pp 39-44. 2014

[34] Watson, C. and Li, F. W. (2014). "Failure rates in introductory pogramming revisited". In Proceedings of the 2014 conference on Innovation & technology in computer science education, pp 39-44. 2014

[35] G. Zichermann and C. Cunningham. "Gamification Design: Implementating Mechanics in Web Mobile Apps". Sebastopol, CA, USA. O'Reilly, 2011.

Antonio J. Tomeu received a BsC and MsC in Computer Science from de University of Granada, Spain in 1992, and a PhD in Mathematics from University of Cádiz, Cádiz, Spain, in 2002. He is Associate Professor of Computer Science in University of Cádiz, coordinator of TECDIS (Red Iberoamericana de Investigación en Tecnologías Concurrentes, Distribuidas y Paralelas), and editor in chief of *Annals of Multicore and GPU Programming*. His current research interests are applications of parallel programming techniques to simulate natural phenomena and physical modeling with cellular automaton.

Alberto G. Salguero received a BsC and MsC in Computer Science from de University of Granada, Spain in 2004, and a PhD in Computer Science from University of Cádiz, Cádiz, Spain, in 2013. He is Associate Professor of Computer Science in University of Cádiz, member of TECDIS (Red Iberoamericana de Investigación en Tecnologías Concurrentes, Distribuidas y Paralelas. His current research interests are applications of parallel programming techniques to simulate natural phenomena and ontologies applied to data integration.