ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

SISTEMA PARA SIMULACIÓN DE UNA SOCIEDAD ARTIFICIAL UTILIZANDO MÉTODOS DE VIDA ARTIFICIAL

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

SANTIAGO DAVID LAZO BEDOYA
slazo@uio.satnet.net

DIRECTOR: PhD. HUGO A. BANDA GAMBOA
hbanda@ieee.org

Quito, Abril 2008

DECLARACIÓN

Yo, Santiago David Lazo Bedoya, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y que, he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Santiago David Lazo Bedoya

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Santiago David Lazo Bedoya, bajo mi supervisión.

PhD. Hugo Banda Gamboa
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Ante todo agradezco a ese Principio Infinito de Compasión, Amor, Poder y Fuerza que trasciende la dualidad de este Universo, para ti mi agradecimiento por inspirarme valor y fuerza en todo momento, a ti mi agradecimiento por sostenerme y bendecirme.

Un profundo agradecimiento para mis padres Heidi Bedoya y Pedro Lazo, quienes han estado a mi lado durante todos estos años, los que con su ayuda, paciencia y amor me han permitido cumplir esta etapa de mi vida.

Para mi tía amada, Mercedes Bedoya de Cervantes, quien me ayudo personalmente para terminar mis estudios, un profundo reconocimiento y eterna gratitud.

Y por supuesto a mi director de Tesis, el Doctor Hugo Banda, por su soporte y guía.

DEDICATORIA

"Aquello que hicimos
nunca está del todo perdido
madurará en su momento
dando su fruto"

Divyavadana

A la Gloriosa *Wildes Heer* de la Segunda Guerra Mundial… *In memoriam*

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE DIAGRAMAS

RESUMEN

Vida Artificial (ALife) es un nuevo campo del conocimiento humano que se apoya fuertemente en las ciencias de la computación, para poder sintetizar y recrear vida, es así como se abren un sinnúmero de nuevas posibilidades y técnicas para distintas áreas de la informática, entre ellas se encuentran la simulación social o de colectividades basada en “seres independientes virtuales” o agentes que generan lo que se conoce como Sociedades Artificiales.

El presente trabajo consta de una investigación sobre la Vida Artificial, las Sociedades Artificiales cuyos conceptos y relaciones se encuentran en el Marco Teórico, en base a estos conceptos e información procedemos al desarrollo de una aplicación que permita manipular una Sociedad Artificial con algunas técnicas de ALife. Finalmente se presentan las respectivas Conclusiones sobre la investigación y el proyecto práctico realizado.

PRESENTACIÓN

Con el incremento del conocimiento humano, y la exploración de nuevas posibilidades en el campo de la ciencia apoyada por la herramienta informática, se ha formado recientemente un nuevo paradigma paralelo y análogo a la Inteligencia Artificial, este es la Vida Artificial, el primero trata de sintetizar, abstraer y representar la Inteligencia, el segundo trata igualmente de representar sintéticamente la Vida.

La Vida Artificial o más conocida en la comunidad internacional por ALife, ha permitido el desarrollo y evolución de muchas técnicas y aplicaciones de lo más variadas que han servido en varias áreas del conocimiento como por ejemplo la biología, la realidad virtual, las gráficas por computador, el entretenimiento, el arte, las ciencias sociales, la robótica, etc.

Para las ciencias sociales y biológicas, la simulación social se ha visto influida por el paradigma de ALife donde la interacción de agentes independientes que pueden ser considerados como “criaturas virtuales” permite la formación de una Sociedad Artificial. En este proyecto nos hemos enfocado principalmente en la sociedad humana, que igualmente es una sociedad animal, pero con diferencias substanciales como la cultura, sin embargo la simulación de la cultura humana es conocida como Cultura Artificial, que trata de sintetizar el surgimiento de la cultura en las sociedades humanas; pero por la amplitud de simular una sociedad humana, hemos decidido enfocarnos solamente en las técnicas de ALife que permiten representar una sociedad humana, libre de los factores culturales que la hacen tan variable y compleja en el tiempo y el espacio.

En base al conocimiento teórico procedemos a la práctica, mediante la ingeniería e implementación de un software que permita la simulación de una Sociedad Artificial pero enfocada en ALife.

Este trabajo es un aporte para el desarrollo de nuevos proyectos de investigación y titulación en la Escuela Politécnica Nacional del Ecuador, ya que las

posibilidades son muy grandes en el campo de ALife, en la simulación social, y en posibles estudios sobre Sociedades Artificiales y Cultura Artificial, al momento de escribir estas líneas son todos estos temas novedosos y objetos de una profunda investigación alrededor del mundo.

Se ha estructurado este proyecto en tres secciones temáticas, el primer capítulo es el marco teórico y conceptual sobre ALife y Sociedades Artificiales, imprescindible pues para avanzar al segundo capítulo el cuál es propiamente el Desarrollo de software, el análisis y diseño, la ingeniería del software y su implementación y pruebas, finalmente el tercer y último capítulo son las conclusiones y recomendaciones resultantes de la elaboración del proyecto.

CAPÍTULO I: MARCO TEÓRICO

1.1 VIDA ARTIFICIAL O ALIFE

El avance de las técnicas y algoritmos utilizados en inteligencia artificial, han permitido desarrollar programas informáticos que simulen procesos vitales y orgánicos, tanto de organismos simples como pueden ser criaturas unicelulares, así como también de entidades más complejas desde el punto de vista biológico.

Ingresamos así a una nueva rama del conocimiento, mundialmente conocida como ALife (Artificial Life) o Vida Artificial, término que fue acuñado por el científico norteamericano Christopher Langton al celebrarse la "Primera Conferencia Internacional de la Síntesis y Simulación de Sistemas Vivientes" organizado en el año de 1987 en el laboratorio Nacional de los Álamos en Estados Unidos.

Esta nueva técnica es causa de un sinnúmero de investigaciones en universidades de todo el mundo con la finalidad de ampliar el rango de conocimientos y aplicaciones prácticas que se puedan derivar de ALife. Es importante mencionar que los estudios relacionados con ALife abarcan ciencias de todo tipo como por ejemplo la neurobiología, la filosofía, ciencias de la computación, sociología, biología, robótica, entre otras.

De lo dicho anteriormente podemos definir a la Vida Artificial o ALife como la rama del conocimiento que pretende sintetizar o abstraer las propiedades y características de los seres vivos en un medio artificial, como pueden ser programas de software, robots, circuitos electrónicos u otros medios.

ALife no solamente se encarga del estudio y simulación de procesos orgánicos, también considera la posibilidad de interacción con el entorno o ambiente y la relación con otros seres. Alife es una poderosa herramienta que mejora y amplía

las capacidades de la biología empírica al permitir recrear condiciones y aspectos que son muy difíciles de obtener en un medio natural.

1.1.1 RESEÑA HISTÓRICA DE ALIFE

Los orígenes de la Vida Artificial como disciplina pueden remontarse desde los albores de las investigaciones hechas en los campos de la Inteligencia Artificial, la Teoría General de Sistemas, la Informática Teórica y la Cibernética, es en estas ciencias donde la Vida Artificial encontraría sus cimientos

A finales de los años 30, el biólogo Ludwing Von Bertalanffy empieza a formular su Teoría General de Sistemas, en la que se conceptúa un Sistema como un conjunto de elementos interrelacionados entre ellos y con el entorno, este sería un modelo amplio que podría ser particularizado en cualquier ámbito específico, como por ejemplo los sistemas biológicos, sistemas dinámicos, sistemas sociales, etc.

Otro precedente de vital importancia fue la publicación del libro “Cybernetics” del matemático norteamericano Norbert Wiener en el año de 1948, donde se hace un estudio de los mecanismos de control en diversos ámbitos, tanto para las máquinas, organizaciones, seres humanos, entre otros, es en la ciencia de la Cibernética donde encontramos uno de los conceptos fundamentales en ALIFE, este es el de retroalimentación o *feedback*, este concepto implica que futuras acciones de control están determinadas por los sucesos precedentes, es un proceso dinámico en el que una cantidad variable depende parcialmente del valor que tenía en el momento anterior .

A finales de la década de 1940 John Von Neumann desarrolla el modelo computacional del autómata autoreplicante, en el cuál se intenta sintetizar una característica fundamental de los sistemas biológicos unicelulares y asexuales, como es la autoreproducción, el pensamiento de Von Neumann se resume en los siguientes lineamientos:

- La autoreproducción se lleva a cabo por una máquina bioquímica.
- El funcionamiento de la máquina bioquímica se puede describir como una secuencia lógica de etapas y acciones, es decir un algoritmo.
- Si el algoritmo puede ser llevado a cabo por alguna máquina, existe entonces una máquina de Turing que puede hacer lo mismo.
- Entonces se podría modelar una máquina de Turing capaz de auto reproducirse.

En base de las ideas del autómata auto replicante, Von Neumann y Stanislaw Ulam crean el modelo computacional del autómata celular, un autómata lógico que no requería cuerpo físico y que podía ser considerado como una matriz con infinito número de puntos que cambiaban su estado a través del tiempo.

Las investigaciones sobre autómatas celulares continuaron durante los años siguientes, pero no sería hasta el año de 1984 cuando se funda el instituto Santa Fe de Estados Unidos dedicado a la investigación interdisciplinar de Sistemas Complejos, y con su auspicio, aparece en 1987 la Vida Artificial como línea de estudio dentro de los Sistemas Complejos. Teniendo como sede el Laboratorio Nacional los Álamos en Nuevo México, se lleva a cabo la “Primera Conferencia Internacional de la Síntesis y Simulación de Sistemas Vivientes”, esta conferencia conocida como ALIFE I fue planificada por un investigador de los autómatas celulares, Christopher Langton, y auspiciada también por el Centro de Estudios No Lineares y Apple Computers, en dicha conferencia se reunieron científicos de las ramas más diversas.

La definición de Christopher Langton en el congreso ALIFE I fue la siguiente: “La Vida Artificial es el campo de estudio dedicado a la comprensión de la vida, intentando abstraer los principios dinámicos fundamentales que subyacen a los fenómenos biológicos, y recreando esas dinámicas en otros medios físicos, haciéndolos accesibles a nuevos tipos de manipulación experimental y de pruebas."

Desde entonces cada dos años el congreso de ALIFE se reúne, con la participación de técnicos y especialistas de diversas disciplinas que fomentan la investigación de la Vida Artificial en el mundo entero.

1.1.2 AREAS CUBIERTAS POR ALIFE

En los años de vida de esta nueva rama del conocimiento, la Vida Artificial se ha enfocado en diversas áreas, entre ellas las más importantes investigaciones se han realizado en:

- Procesos a nivel prebiótico, es decir investigaciones encaminadas a los mecanismos bioquímicos que permiten la aparición de una célula orgánica.
- Procesos a nivel celular, especialmente se ha estudiado en este campo, la auto replicación o auto reproducción celular.
- Procesos a nivel de organismo, en este sentido se han desarrollado investigaciones sobre la evolución, la auto adaptación y el aprendizaje.
- Procesos a nivel colectivo, donde se estudia la interacción de un organismo con otros de la misma o de distintas especies, así como también con su entorno.

En todas estas investigaciones la Inteligencia Artificial aporta con técnicas y mecanismos que permiten el desarrollo de estos estudios.

1.1.3 AUTÓMATAS CELULARES (AC)

Los autómatas celulares son sistemas dinámicos, que son discretos en el tiempo y en el espacio, operan en una malla regular y uniforme, donde cada punto es llamado una célula o celda, estas celdas cambian de estado durante cada ciclo de reloj de acuerdo a reglas predefinidas, la actualización de las celdas se realiza de forma sincrónica.

Como se dijo anteriormente el matemático John Von Neumann fue quien originalmente plantea los Autómatas Celulares como modelos formales de organismos que se auto reproducen, por lo tanto su relación con la Vida Artificial es estrecha, ya que los Autómatas Celulares podrían simular el crecimiento

celular, los microorganismos, las bacterias, entre otras caracterizaciones del mundo biológico.

El modelo original fue considerado como un arreglo infinito de una o dos dimensiones, en el cuál se considera el avance del tiempo, el cuál es discreto, en cada punto en el tiempo, cada célula del arreglo se encontrará en un estado dado de un conjunto finito de estados posibles, con cada paso de avance en el tiempo o ciclo de reloj, las celdas cambian de estado de acuerdo con el estado propio y el de las celdas aledañas; por ejemplo, si el arreglo es de una dimensión, los estados de las celdas derecha e izquierda determinarán el futuro estado de la celda del medio.

La regla que determina como cambian los estados de todas las celdas en el transcurso del tiempo, es conocida como *regla local* o *función.* El conjunto de estados de las celdas en cualquier momento, se denomina *estado global* o *configuración* del Autómata Celular en el momento *t,* para el momento *t=0* se dice que el Autómata se encuentra en la *configuración inicial.*

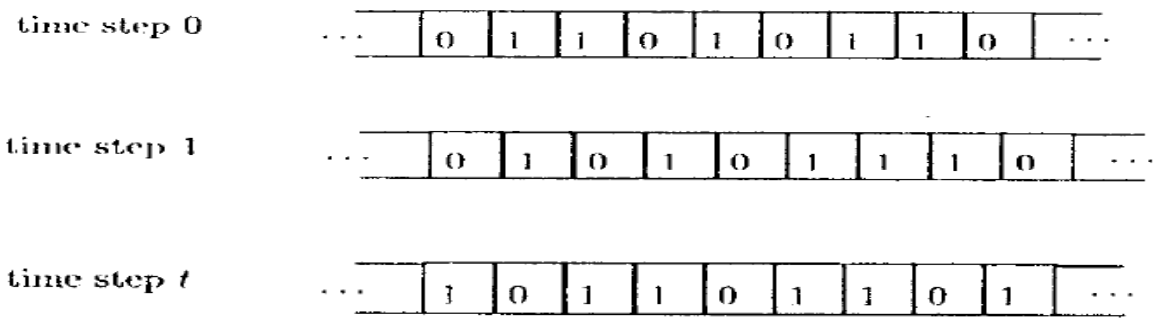


Fig. 1.1. Transformación o evolución de un Autómata Celular de dimensión lineal

De lo dicho se establece que los Autómatas Celulares, están constituidos de cuatro características:

- La geometría.
- La regla local de transición.
- El conjunto de estados.
- El vecindario.

1.1.3.1 La geometría de los Autómatas Celulares

La geometría es la matriz o arreglo de dimensión *n,* donde se encuentran las celdas o células, el límite de la matriz podría ser infinito o finito, para el caso de que la matriz sea finita se han hechos propuestas con respecto a los límites, en las que las celdas que corresponden a los límites de la matriz, se ven afectados de forma distinta con respecto a la regla local de transición.

1.1.3.2 La regla local de transición

La *regla local* o función es la que permite el cambio de estados de las celdas dentro de la matriz, aunque generalmente la regla local es una para todas las celdas, existen Autómatas en los que cada celda o grupos de celdas tienen su propia regla local, estos autómatas celulares se conocen como Híbridos. Así también se tienen celdas que con cada ciclo de tiempo pueden cambiar su *regla local*, en estos casos se trata de un AC Programable.

Si para una *configuración inicial A* de un AC llegamos a la *configuración B* en un momento *t,* por medio de la *regla local* ρ, entonces se dice que existe una *regla local* ρ^{-1}, de tal forma que aplicándola volveremos de la *configuración B* a la *configuración inicial A*. De esta forma se dice que un AC es invertible si su regla local es invertible.

1.1.3.3 El conjunto de estados de las celdas

Las celdas de un AC pueden tomar un solo estado de un conjunto posibles de estados en un momento determinado. Existen también autómatas celulares cuyas celdas pueden tener diferentes conjuntos de estados.

1.1.3.4 El vecindario de las celdas

La geometría del Autómata determinará el tipo de vecindario que tendrán las celdas, sin embargo existen tipos particulares, como para los arreglos de dos dimensiones: el *vecindario Von Neumann* u *ortogonal,* el cual define que el

vecindario de una celda dada estará determinado por las celdas aledañas que se encuentran en el norte, sur, este y oeste, es decir de forma ortogonal y *El vecindario de Moore*, en el cual las 8 celdas que rodean a una celda forman el vecindario de la misma. Existen muchas definiciones de vecindarios para AC, sin embargo los anteriormente mencionados son los más conocidos y utilizados.

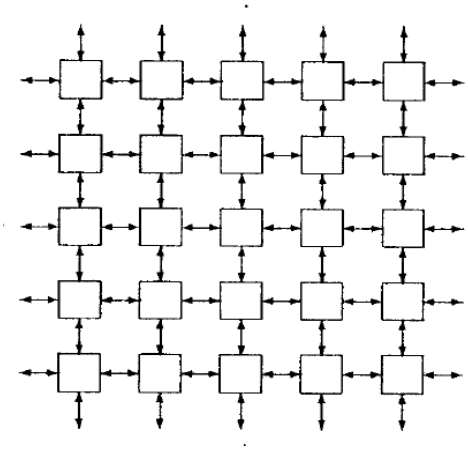


Fig. 1.2. Un Autómata Celular de dos dimensiones con vecindario ortogonal

El vecindario de una celda es el que determinará el estado siguiente de esa celda, siempre en relación con lo que determine la regla local.

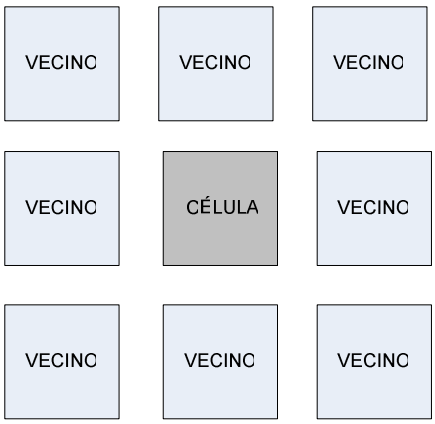


Fig. 1.3. Una célula con vecindario de Moore

1.1.3.5 El juego de la vida

El juego de la vida es el mejor ejemplo de un AC, fue diseñado por el matemático inglés John Horton Conway en el año de 1970, se dio a conocer por medio de la revista *Scientific American*.

La idea original era diseñar un conjunto de reglas simples que permitiesen estudiar el comportamiento general o macroscópico de una población. El criterio principal para la selección de las reglas, fue basado en el principio de que el crecimiento o decrecimiento de una población no es fácilmente predecible.

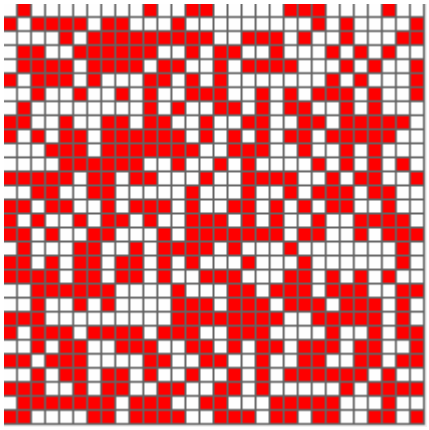
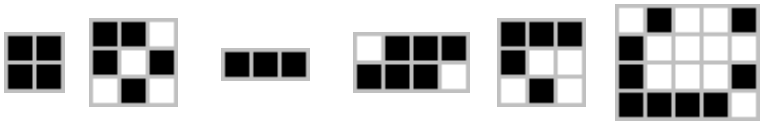


Fig. 1.4. El juego de la vida

Después de un sinnúmero de estudios e investigaciones, Conway determinó que la población estaría determinada por las celdas que se encontrarían en un arreglo de dos dimensiones, en el cuál se utilizaría el *vecindario de Moore,* cada celda puede tener uno de dos estados con el valor uno o cero; la regla local se define de la siguiente manera:

- La supervivencia: si una celda o célula se encuentra en estado 1 (está viva) y tiene dos o tres vecinos en estado 1, entonces la celda sobrevive, es decir se mantiene en estado 1.
- Los nacimientos: si una celda se encuentra en estado cero, y se encuentra rodeada exactamente de 3 vecinos en estado 1, entonces en el siguiente ciclo de reloj la celda pasará a estado 1.
- Las muertes: si una celda se encuentra en estado 1, pasa a estado 0 (muere), cuando se encuentra solitaria, es decir cuando hay cero o un vecino en estado 1, igualmente si hay sobrepoblación, es decir si cuatro o más vecinos están en estado 1.



Bloque *Block* Barco *Boat* Parpadeador *Blinker* Sapo *Toad* Planeador *Glider* Nave ligera *LWSS*

Fig. 1.5. Distintos patrones estudiados en el Juego de la Vida

El juego de la vida de Conway ha sido objeto de varios estudios, especialmente matemáticos y científicos se han preocupado por la complejidad de los patrones obtenidos a partir de reglas muy sencillas, estos factores, han permitido el análisis de secuencias armónicas o caóticas de crecimiento, decrecimiento y estabilización de poblaciones. Por ejemplo se inicia la evolución con un alto nivel de entropía, es decir el grado de desorden de un sistema, pero con el transcurso de las iteraciones, observamos que esta entropía disminuye, y el sistema se va estabilizando, podríamos decir que en este caso el AC presenta una tendencia auto organizativa.

Fig. 1.6. Patrones complejos realizados con autómatas celulares

1.1.3.6 Clasificación de los AC

S. Wolfram, uno de los más grandes investigadores de AC del mundo, planteó una clasificación de los AC de acuerdo a su conducta, basada en la medida de la entropía.

La entropía es la medida del desorden o caos de un sistema, para los AC, se ha determinado empíricamente que la medida de la entropía está dada por:

$$\sum p_i \log p_i$$

Donde p_i es la probabilidad de que ocurra la *configuración i*.
En base a este criterio la clasificación para los AC sería de cuatro clases:

- La evolución conduce a un estado homogéneo.
- La evolución conduce a un conjunto de estructuras simples estacionarias o periódicas.
- La evolución conduce a un patrón caótico.
- La evolución conduce a estructuras complejas, las cuáles son, en muchas ocasiones, de larga vida.

Existen otros intentos por formalizar esta clasificación, como el criterio de familias de reglas, pero la idea principal subyace en esta clasificación.

1.1.3.7 Aplicaciones de los autómatas celulares

Científicos de todas las áreas han investigado los autómatas celulares, para poder aplicarlos en sus respectivas ramas con fines prácticos, ya que los AC permiten modelar sistemas complejos, por lo tanto son una alternativa a las ecuaciones diferenciales, y son utilizados para modelar sistemas físicos como la interacción entre partículas, la formación de galaxias, la cinética de sistemas moleculares, entre otros, pero especialmente en el campo de la biología, donde los AC han servido para modelar sistemas biológicos, a nivel celular, multicelular y poblacional, estos AC se han denominado dinámicos, puesto que las celdas de los mismos pueden aparecer o desaparecer en el tiempo.

Con lo dicho anteriormente podemos observar que los AC se convierten en una herramienta, tanto para la Inteligencia Artificial, como para la Vida Artificial.

1.1.4 RELACIÓN DE ALIFE CON LA INTELIGENCIA ARTIFICIAL

Se puede considerar que la Inteligencia Artificial es a la Psicología, como la Vida Artificial es a la Biología, es decir, una forma de sintetizar o abstraer un modelo natural para representarlo en un medio artificial, la inteligencia artificial abstrae los procesos cognitivos, la vida artificial los procesos biológicos.

La Vida Artificial se sirve de la IA para simular ciertas características de los seres vivos, especialmente en lo que se refiere a la evolución y el aprendizaje.

Para la evolución son los algoritmos evolutivos como los algoritmos genéticos los que prestan su valioso aporte. Para el aprendizaje se pueden utilizar diversos métodos, sin embargo, son las redes neuronales, las más utilizadas. También se utilizan distintos métodos heurísticos de búsqueda entre otros.

1.1.4.1 Algoritmos Genéticos (AG)

A mediados de la década de 1960, las investigaciones en el campo de la Inteligencia Artificial dan como resultado la invención de los Algoritmos Genéticos. John Holland de la universidad de Michigan, en Estados Unidos, tuvo la idea de crear programas informáticos que pudieran evolucionar para cumplir con ciertas tareas.

Los Algoritmos Genéticos nacen de la teoría de la evolución de las especies, en la cual, los más aptos se adaptan a las circunstancias del entorno cambiante, sobreviven y transmiten sus cualidades a su descendencia.

En la obra *El origen de las especies,* publicada en 1859, Charles Darwin, enunció su teoría de la *Evolución de las especies mediante la selección natural*, la cuál ha

sido aceptada en el mundo científico, siendo objeto de varios análisis y revisiones, sin embargo podemos resumir dicha teoría en los siguientes puntos:

- Cada individuo tiende a transmitir rasgos a su progenie.
- La naturaleza produce individuos con rasgos diferentes.
- Los individuos más adaptados, es decir aquellos que tienen los rasgos más favorables, tienden a tener más progenie, que aquellos con rasgos menos favorables.
- Durante largos períodos de tiempo, las especies evolucionaran para adaptarse a entornos particulares, produciendo nuevas especies.

1.1.4.1.1 Los cromosomas y los rasgos hereditarios

En los organismos superiores, cada célula contiene un solo núcleo, el cual encierra a los *cromosomas,* éstos son los custodios de los factores determinantes de los rasgos hereditarios, conocidos como *genes,* que se transmiten cuando las células se dividen, así como en el proceso de reproducción.

Los *cromosomas* generalmente vienen por pares, y en la *reproducción sexual* cada padre contribuye con un *cromosoma* para cada nueva pareja de *cromosomas* del descendiente. Mediante un proceso conocido como *recombinación genética,* los *genes* de los *cromosomas* de los padres se mezclan, dando lugar a las características genéticas hereditarias del nuevo ser que ha sido engendrado; en ciertas ocasiones, en el proceso de duplicación del *cromosoma* existe una alteración que produce un gen nuevo distinto al del progenitor, este proceso se conoce como *mutación.*

Desde un punto de vista molecular, la *selección natural* se produce a partir de la *recombinación genética* que integra *genes* existentes en nuevas combinaciones, y la *mutación* que produce nuevos *genes*.

1.1.4.1.2 Los Algoritmos Genéticos como herramienta para la optimización y simulación evolutiva

Los AG son utilizados para encontrar soluciones óptimas, se puede resumir los AG en los siguientes puntos:

- Se debe establecer los valores, parámetros o características de un problema determinado como el código genético de un cromosoma dado.
- Se debe combinar los cromosomas para combinar el código genético.
- Se puede establecer un parámetro de mutación en los códigos genéticos, para crear mayor variedad de códigos genéticos.
- En base a una función que determine la “aptitud” de un código genético, se puede determinar si dicho código sobrevivirá o morirá en la siguiente generación.
- De esta forma los códigos genéticos que muestren mayor “aptitud” para resolver el problema se combinarán en cada ciclo o iteración de recombinación genética.
- Finalmente cuando se alcanza una estabilización del código genético más “apto”, la solución óptima para el problema es hallado en los valores del código genético.

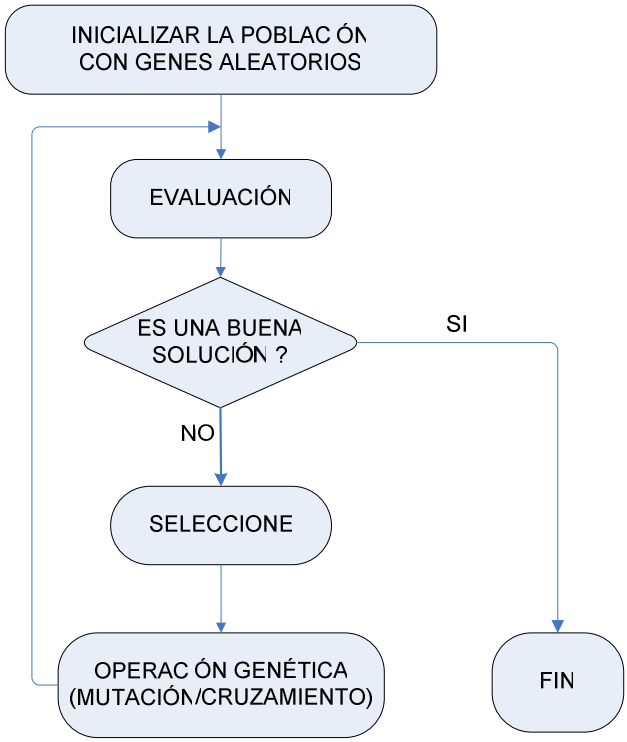


Fig. 1.7. Generalización de los algoritmos genéticos.

Para que los AG tengan éxito en un campo particular de la investigación, es necesario saber modelar correctamente el problema planteado en base a las leyes de la evolución y transmisión hereditaria del código genético descritas en la sección anterior, es decir, es importante hacer una analogía entre el problema propuesto y el modelo biológico de la evolución, para poder realizar un modelo computacional adecuado a la investigación que se realice.

1.1.4.2 Evolución artificial

En ALife podemos hablar de evolución artificial mediante AG, y para esto es necesario definir los conceptos de *genotipo* y *fenotipo.*

Genotipo: En biología el genotipo está compuesto de ADN que contiene las instrucciones (código genético) para el desarrollo del organismo, para Alife el *genotipo* será una representación codificada de un posible organismo artificial. Los AG generalmente representan los genotipos con cadenas de números binarios o valores de parámetros.

Fenotipo: En biología el fenotipo es considerado la manifestación visible del genotipo, como la morfología, la rapidez, la fuerza, etc. En Alife el fenotipo podrá representar la morfología de una *criatura virtual,* así como su forma de desplazamiento, velocidad, entre otras características visibles.

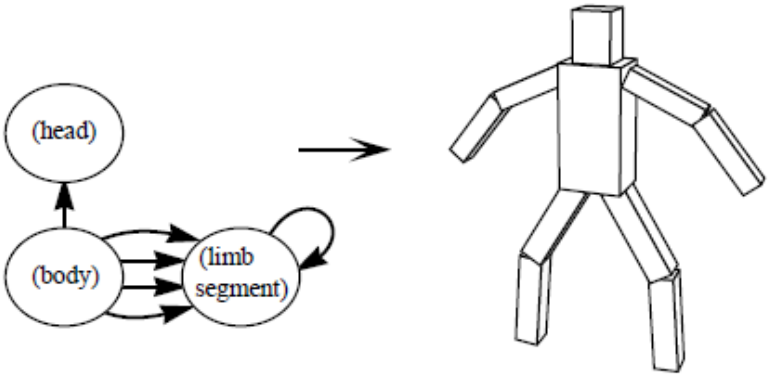


Fig. 1.8. Los genotipos representados como grafos determinarán el fenotipo y por lo tanto la morfología del organismo artificial.

1.1.5 CRIATURAS VIRTUALES

El desarrollo de los ambientes virtuales en Tercera Dimensión o 3D, ha permitido que se generen entornos virtuales, donde *seres* o *criaturas artificiales* se desarrollen, es decir cumplan el ciclo vital de un ser orgánico, esto es: nacer, crecer, reproducirse, envejecer y morir.

Estas *criaturas virtuales* suelen ser programas de software, que en base a algoritmos de inteligencia artificial interactúan en un medio ambiente simulado, pueden estar a aislados o en interacción con otros seres virtuales, en este último caso podemos hablar ya de una *sociedad artificial virtual*.

Los algoritmos de inteligencia artificial permiten a estas criaturas virtuales, adaptarse a los cambios del entorno, aprender, y evolucionar. Para poder llamarse criaturas virtuales, es necesario que la simulación se lleve a cabo en una interfaz gráfica que represente un mundo 3D, en el que las criaturas podrán adoptar formas o diseños (morfología) que emulen la vida orgánica conocida, o tomar formas variadas, en una estructura física tridimensional.

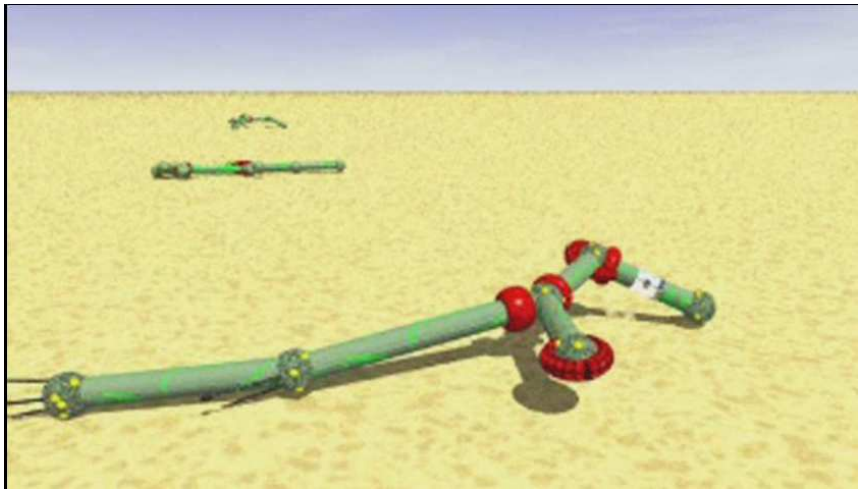


Fig. 1.9. Imagen de una Criatura Virtual en su entorno.

Mediante procesos de animación gráfica las criaturas virtuales, pueden realizar distintas actividades, tales como caminar, nadar, volar, comer, reproducirse, entre otras, es decir interactuar con el entorno, por lo tanto la arquitectura y diseño de estos sistemas incluyen distintos tipos de módulos, para poder controlar los diversos aspectos de la simulación de una forma coherente.

El *entorno virtual o ecosistema virtual,* va a ser el mundo 3D que emula un ecosistema, es decir el medio ambiente o entorno, donde se encuentran los recursos diversos que necesitan las criaturas virtuales para vivir e interactuar.

Para que las criaturas virtuales interactúen con el entorno, es necesario proveer un sistema de control que permita el desplazamiento y movimiento de cada entidad virtual, para esto generalmente se utiliza una estructura basada en:

- Sensores: que reciben señales provenientes del entorno, o del estado interno del organismo.
- Neuronas: que procesan las entradas de las señales producidas por los sensores.
- Efectores: que son las señales de salida producidas por las neuronas las cuáles determinaran una acción determinada sobre el entorno o en sí mismo.

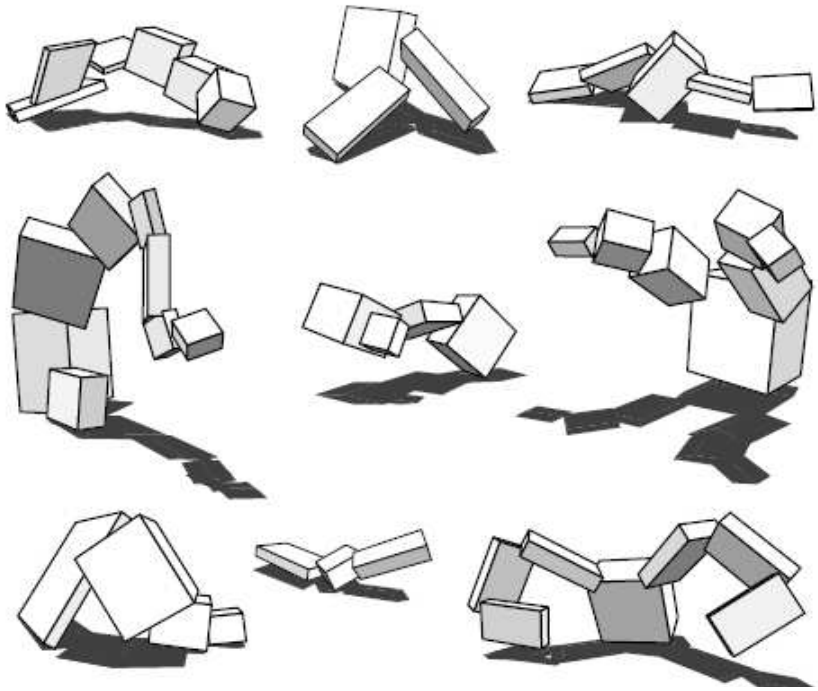


Fig. 1.10. Criaturas Virtuales que han evolucionado en un Ecosistema Virtual, mediante Algoritmos Genéticos para desplazarse de forma óptima en el entorno.

A través de la morfología de las criaturas virtuales y la evolución artificial mediante algoritmos de optimización como los algoritmos genéticos, se puede encontrar diseños óptimos para ciertos problemas, que pueden asombrar a ingenieros y diseñadores en áreas específicas. También el mundo de los juegos de video se vería beneficiado por la participación de criaturas virtuales que evolucionen y se adapten a condiciones diversas, creando situaciones inesperadas para los videojugadores. También los sistemas basados en Ambientes Virtuales Cooperativos en Red o NCVE (Networked Collaborative Virtual Environments) se podrían beneficiar de la presencia de criaturas virtuales autónomas, que cumplan funciones específicas, para las personas que interactúan en el ambiente virtual, estos seres virtuales podrían aprender y evolucionar en el entorno y de esta forma brindar nuevas posibilidades a estos tipos de sistemas.

Karl Sims, biólogo y especialista en gráficos de computadores, fue el pionero en el planteamiento de la evolución de criaturas virtuales, su artículo “Evolving Virtual Creatures” de 1994 es un hito en ALife, así también los videos y conferencias

sobre estos seres virtuales creados por Sims han inspirado a muchos investigadores a proseguir en esta línea de investigación.

1.1.6 ORGANISMOS DIGITALES

En el mundo de ALife se define a un organismo digital como un programa capaz de auto replicarse y evolucionar, los organismos digitales nacen a partir de la idea del juego *Core War,* que en 1990 fue muy popular entre programadores de lenguaje ensamblador, este juego se basa en programas que compiten por los recursos del computador, el mejor programa elimina a los otros y se apropia de los recursos, Steen Rasmussen crea entonces el *Core World,* donde los programas eran auto replicantes y podían mutar, sin embargo el modelo no era coherente con lo que se quería representar; posteriormente, en base a las mismas ideas, el biólogo Tom Ray diseña el sistema *Tierra,* el cuál es un hito en Alife y precursor del sistema *AVIDA* que describimos a continuación como un ejemplo de un sistema que crea y manipula vida artificial, y más concretamente organismos digitales.

La microbiología es una de las ciencias que más se ha beneficiado con los organismos digitales, pues permite la representación simulada de poblaciones y comportamientos de virus, bacterias, células, entre otros.

1.1.6.1 AVIDA como ejemplo de experimentación con organismos digitales

Como un ejemplo de la investigación y aplicaciones que existen actualmente en Alife indicaremos AVIDA, el cuál es una plataforma de software ampliamente difundida de Alife, que permite realizar experimentos sobre organismos digitales, provee un detallado control sobre los parámetros de experimentación, y tiene también herramientas sofisticadas de medición y análisis de los datos obtenidos en los experimentos.

En AVIDA un *genoma* (conjunto de genes) es una secuencia circular de n instrucciones de procesador, en analogía con las instrucciones que conlleva el código genético para los seres vivos. Las instrucciones se ejecutan

secuencialmente, y estas pueden realizar saltos a otras instrucciones, se ejecuta solamente una instrucción al mismo tiempo, el conjunto de instrucciones disponibles son 26, en la secuencia o *genoma* se pueden repetir estas instrucciones muchas veces.

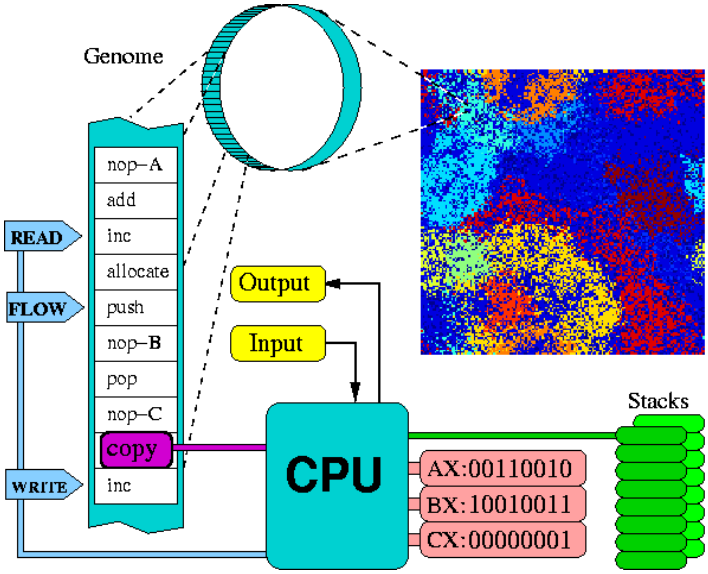


Fig. 1.11. Diseño del funcionamiento de un organismo digital en AVIDA

La replicación o reproducción de los organismo digitales de AVIDA se realiza de forma asexual copiando su *genoma* o conjunto de instrucciones lógicas, con una pequeña alteración o mutación del mismo. Los organismos compiten por la utilización de CPU en el computador para ejecutar sus instrucciones, y de esta forma poder "vivir" y reproducirse.

La utilización del CPU está determinada por la longitud del *genoma, así* como por las operaciones lógicas que pueda realizar con cadenas binarias de 32 bits, así la utilización del CPU para estos organismos digitales, tendría una analogía directa con los seres vivos en cuanto tiene que ver con la aptitud y competencia por los recursos.

La muerte de los organismos se determina por un factor aleatorio, o porque en un determinado número de ciclos no ha realizado ninguna instrucción mediante el CPU.

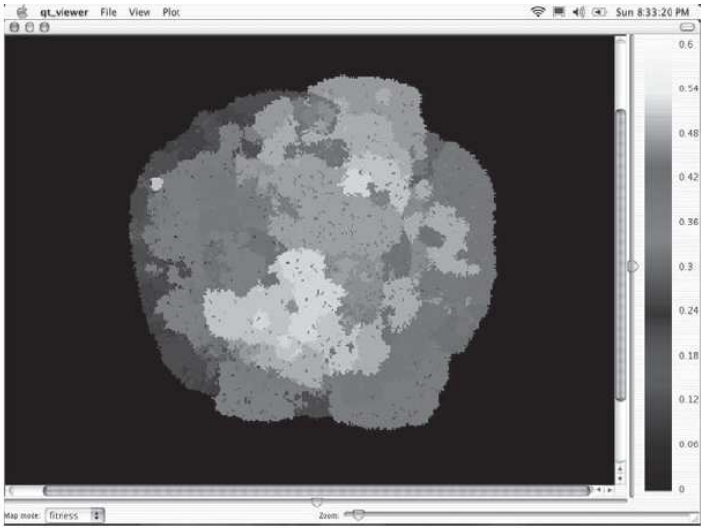


Fig. 1.12. Ejecución de un experimento de organismos digitales utilizando AVIDA

AVIDA ha permitido que universidades y laboratorios del mundo entero realicen experimentos de los más variados, entre los cuales se puede destacar investigaciones sobre la evolución, la reproducción sexual o asexual, las mutaciones, las poblaciones, etc. El mantenimiento del sistema AVIDA se lleva a cabo por la Universidad de Michigan y el Instituto de tecnología de California, en E.E.U.U.

1.1.7 ALIFE EN MULTIMEDIA, ANIMACIÓN Y ARTE

Las técnicas y métodos de ALife han sido utilizados también en campos como el arte, el diseño gráfico, la música y la animación. Por medio de los principios de la computación evolutiva, se han determinado patrones multimedia que pueden evolucionar según los principios de los algoritmos genéticos.

Cuando el mismo usuario selecciona los patrones que prefiere, en vez de que el algoritmo genético lo haga por él, entonces hablamos de *Computación Evolutiva Interactiva* ya que el usuario interviene en la evolución de los patrones o genotipos, en base a criterios subjetivos como puede ser el gusto de una melodía o de una imagen.

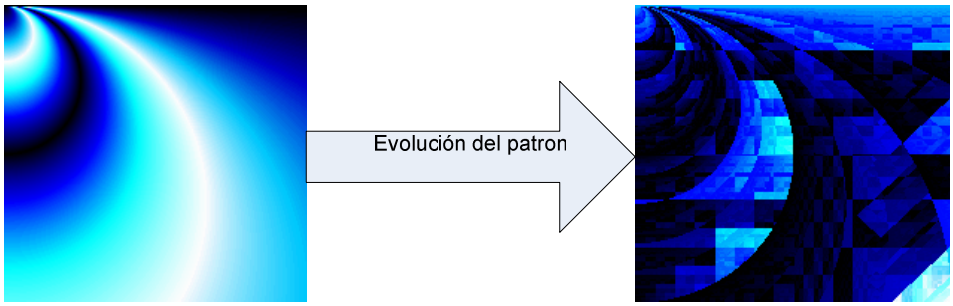


Fig. 1.13. Evolución de patrones o “individuos” seleccionados por los usuarios.

También los complejos patrones emergentes obtenidos por medio de los autómatas celulares, sirven para formar imágenes y sonidos que resultan fascinantes.

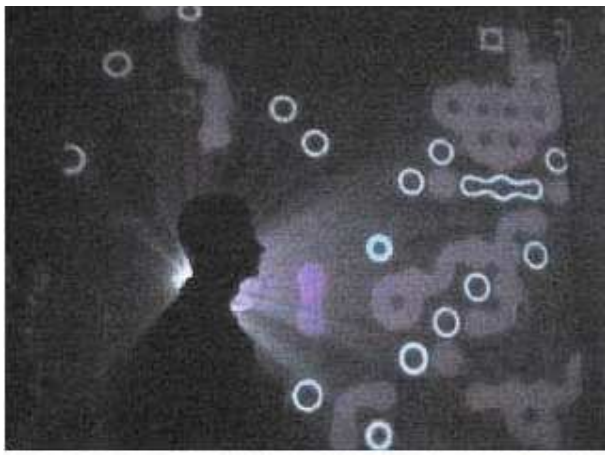


Fig. 1.14. Galería de Arte basada en el funcionamiento de los Autómatas Celulares

En el mundo de los videojuegos, muchas casas desarrolladoras de software utilizan técnicas de ALife para formar seres virtuales con distintas propiedades que interactúan con los video jugadores.

De esta forma muchos artistas, músicos y diseñadores están utilizando herramientas que usan técnicas de ALife y computación evolutiva para ampliar sus capacidades y explorar nuevos métodos de creación artística.

1.1.8 CRÍTICAS Y DEBATES SOBRE LA VIDA ARTIFICIAL

La Vida Artificial ha sido objeto de duras críticas por parte de algunos científicos y biólogos, los cuales opinan que la Vida Artificial no es una ciencia seria que se

proponga simular los procesos biológicos, y se basan muchas veces para estas afirmaciones, en que los científicos e ingenieros que se ocupan de ALife no trabajan conjuntamente con especialistas en el área de la biología.

El famoso biólogo y genetista británico John Maynard Smith ha hecho muchas críticas a ciertos trabajos de ALife por considerarlos poco rigurosos[1]. Sin embargo las investigaciones científicas serias respecto de ALife, así como el trabajo conjunto de biólogos, genetistas e ingenieros ha disipado muchas de estas críticas.

Ciertamente ALife es una disciplina reciente que se propone sintetizar o simular la vida en un medio artificial, los métodos y técnicas de que dispone actualmente se irán perfeccionando con el tiempo y la investigación, así como también los avances en nanotecnología e informática permitirán ampliar el campo de estudio de ALife y su relación con otras ciencias como la biología.

Actualmente existen dentro del área de ALife dos tipos de posturas:

- *Vida Artificial Fuerte:* que considera que la vida puede ser "abstraída o sintetizada" en cualquier medio particular, que sea orgánico o no.
- *Vida Artificial Débil:* que considera que no se puede reproducir un proceso viviente si no es en materia orgánica y mediante procesos químicos.

Las posturas anteriores, son el origen de controversias científicas y filosóficas, puesto que nos plantean la siguiente pregunta ¿Puede existir la vida en un medio no orgánico? Pero este es un asunto del que se encargará la ciencia y la filosofía de resolver.

1.2 SOCIEDADES ARTIFICIALES (SA)

Las ciencias exactas como las matemáticas, la química y la física tienen la ventaja de que pueden recrear ciertas condiciones particulares para realizar

[1] Horgan, J: "From Complexity to Perplexity". Scientific American. p107

experimentos, y de esa forma deducir leyes y principios, pero las ciencias como la Sociología, Sociobiología, Demografía, Economía Política, Antropología, entre otras relacionadas con los sistemas sociales, no tienen la facilidad de disponer con laboratorios donde se puedan controlar las condiciones que se desean estudiar.

El avance de las ciencias de la computación, así como el modelamiento matemático de sistemas complejos, como son los sistemas sociales, permite que se puedan emular en entornos artificiales, algunas características del comportamiento colectivo social, humano o no humano.

Las Sociedades Artificiales (SA), son por lo tanto, modelos computacionales que permiten simular la interacción de entidades pertenecientes a una colectividad o sociedad, para el estudio y análisis de distintas áreas de interés.

Como una definición formal podemos citar: “Una Sociedad Artificial se refiere a *modelos de simulación basados en agentes*, utilizados para descubrir estructuras sociales globales y conducta colectiva, producidas por reglas simples y mecanismos de interacción”[2]

El término Sociedad Artificial fue introducido por Joshua Epstein y Robert Axtell que en su obra *Growing Artificial Societies* del año 1996, donde plantean que se puede analizar complejos procesos sociales a través de reglas simples que pueden ser simuladas en un computador.

Las Sociedades Artificiales tratan sobre la interacción de individuos u organismos simulados, que pueden ser considerados *agentes*, los cuales pueden estar diseñados en base a los principios de Alife que hemos descrito, según esto, las SA pueden emular el comportamiento individual y colectivo de seres simples, microorganismos, animales superiores o seres humanos.

[2] Barry Lawson, Steve Park:”Asynchronous Time Evolution in an Artificial Society Mode”, *journal of artificial societies and social simulation Vol. 3, Num. 1 .*

Si bien, actualmente es muy difícil emular todos los parámetros de una sociedad humana, los avances tecnológicos, y las investigaciones en dichas áreas prometen a largo plazo que estas Sociedades Artificiales se acerquen cada vez más a la realidad que se quiere simular.

1.2.1 SISTEMAS SOCIALES COMO SISTEMAS COMPLEJOS

Un Sistema Complejo podría ser considerado como un conjunto de varios elementos que interactúan, pero donde la *Complejidad* del sistema está en relación directa con el número de elementos y de sus relaciones, además se considera que cada elemento de un sistema complejo, puede ser considerado como otro sistema complejo.

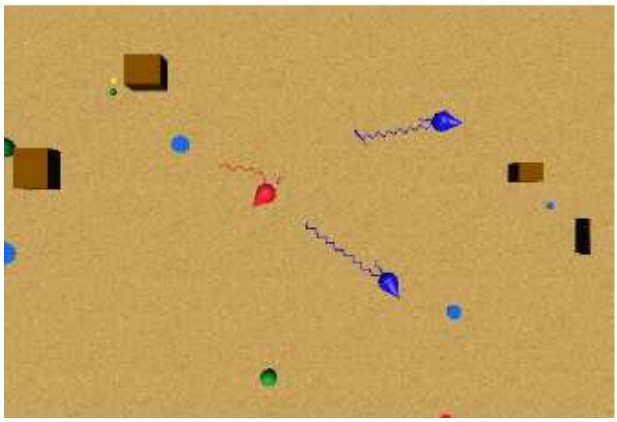


Fig. 1.15. Patrones de conducta derivados de la interacción entre los individuos de la colectividad en un entorno virtual.

Si bien no existe un límite muy diferenciado entre Sistemas Simples y Complejos, podemos añadir, que un Sistema Simple es fácilmente predecible, y sus elementos e interacciones son pocos y bien conocidos, en cambio, los Sistemas Complejos son todo lo contrario.

Como ejemplos de Sistemas Simples podemos nombrar:

- El movimiento de un proyectil en condiciones normales.
- El giro de una rueda.
- Sistema de ventilación.

- Sistema informático de inventarios.

Los sistemas nombrados son fácilmente predecibles, y pueden ser descritos por reglas, formulas o diagramas, pero en la naturaleza podemos encontrar ejemplos de complejidad:

- El ser humano.
- Un enjambre de abejas.
- Una colonia de hormigas.
- Ecosistema

La sociedad al ser una agrupación de individuos que interactúan entre ellos presentará las características de un sistema complejo, puesto que el número de variables y relaciones del sistema es muy grande, no es fácil predecir el comportamiento social, además que cada elemento es un sistema complejo por sí mismo.

1.2.1.1 Emergencia y propiedades emergentes de los sistemas complejos

Es importante mencionar el concepto de *emergencia (emergence)* dentro de la teoría de sistemas, puesto que la *emergencia* es el surgimiento de sistemas y patrones complejos que nacen o *emergen* a partir de múltiples interacciones simples o relativamente simples. “Emergencia puede ser definida como el surgimiento de novedosas y coherentes estructuras, patrones y propiedades durante el proceso de auto organización en sistemas complejos”[3]

Al ser la sociedad un sistema complejo, podemos apreciar un ejemplo de *emergencia,* en el surgimiento de una cultura específica en una sociedad humana.

Cuando elementos de un sistema interactúan produciendo espontáneamente conductas y propiedades simples o complejas en el sistema podemos hablar del

[3] *Goldstein, Jeffrey (1999), "Emergence as a Construct: History and Issues", Emergence: Complexity and Organization **1**: 49-72*

aparecimiento de *conductas y propiedades emergentes,* éstas no son fáciles de predecir y son consecuencia del gran número de interacciones entre los componentes del sistema, así como de factores aleatorios y variables que puedan influir.

Las sociedades humanas al ser sistemas complejos presentarán la aparición de propiedades emergentes, éstas también son características propias de los sistemas biológicos, y pueden representarse por una interacción de criaturas digitales hechas con tecnología de ALife dentro de una sociedad artificial.

1.2.1.2 Sistemas Complejos Adaptativos (CAS Complex Adaptive Systems)

Los sistemas complejos adaptativos son un caso particular de los sistemas complejos, la definición de John H. Holland - uno de los creadores del término - es la más apropiada: "Un Sistema Complejo Adaptativo es una red dinámica de varios agentes (que pueden representar células, individuos, organizaciones, naciones, etc.) actuando en paralelo, constantemente interactuando y reaccionando a lo que hacen los otros agentes, el control de los CAS tiende a ser altamente distribuido y descentralizado. Si existe alguna conducta coherente en el sistema, esta surge de la cooperación o de la competición entre los mismos agentes. La conducta global del sistema es el resultado de un enorme número de decisiones hechas por varios agentes individuales"[4]

En resumidas cuentas CAS es un sistema complejo, que mediante la interacción de sus elementos tiende a auto organizarse, es decir los elementos o agentes del sistema aprenden y se adaptan a los constantes cambios del entorno.

Por lo tanto los sistemas sociales caen dentro de este particular enfoque de los sistemas complejos, ya que las sociedades y organizaciones tienden a adaptarse y a autoorganizarse.

[4] M. Mitchell Waldrop:" Complexity: The Emerging Science at the Edge of Order and Chaos".

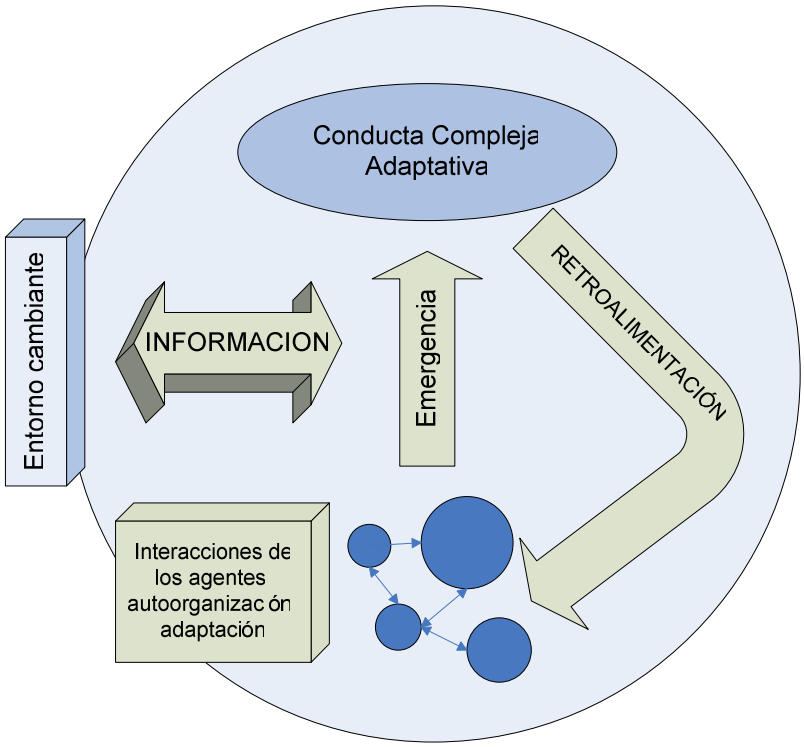


Fig. 1.16. Diagrama esquemático de un sistema complejo adaptativo.

1.2.2 SIMULACION SOCIAL

La simulación social trata sobre el análisis y modelamiento de los fenómenos y dinámicas sociales para poder ser simuladas en computadores. Los principales enfoques para la simulación de sistemas sociales son:

- Modelamiento y simulación a nivel de sistema
- Modelamiento y simulación basada en agentes

Modelamiento y simulación a nivel de sistema

Es la más antigua técnica de simulación social, ésta se enfoca en la situación global, sin considerar el comportamiento ni la interacción de los elementos sociales individuales, se basa en variables globales, ecuaciones complejas y datos que alteran el entorno social. Se dice que es un enfoque *Top-down* o *de Arriba hacia abajo.* Por su metodología, está técnica no tiene relación con ALife, puesto que no se considera la actividad de cada individuo o elemento dentro del entorno.

Modelamiento y simulación basada en agentes

En esta técnica se considera el comportamiento y actividad que cada elemento o agente tiene en el entorno social, es así, como las interacciones simples de los agentes, producen resultados inesperados y comportamientos complejos a nivel global. Por su características, esta técnica de simulación social es la que más se vincula con ALife, puesto que cada elemento "viviente" de un entorno artificial puede ser considerado un agente. Se dice que es un enfoque *Bottom-Up* o *de Abajo hacia arriba*. Cuando una simulación social se basa en agentes, podemos hablar entonces de una sociedad artificial.

La simulación social es objeto de varias investigaciones para entender el comportamiento de organizaciones, conducta de clientes, conducta grupal, y en fin muchos fenómenos y procesos sociales, por esta razón ha surgido una nueva rama de la sociología llamada *sociología computacional,* que utiliza las ciencias de la computación para el estudio de los fenómenos sociales.

Cabe mencionar que la comunidad de investigadores sobre simulación social ha crecido en todo el mundo, y en agosto del año 2006, en la ciudad de Kyoto en Japón, se reunió el Primer Congreso sobre Simulación Social, el cuál marca un hito en este campo del conocimiento, hay que destacar que este evento fue patrocinado por organismos internacionales encargados de la difusión e investigación sobre simulación social:

- North American Association for Computational Social and Organizational Sciences (NAACSOS)
- European Social Simulation Association (ESSA)
- Pacific Asian Association for Agent-Based Approach in Social Systems Science (PAAA)

1.2.3 MODELAMIENTO Y SIMULACIÓN BASADA EN AGENTES

Los modelos basados en agentes, son modelos computacionales que utilizan *agentes* como entidades autónomas que interactúan en un sistema o entorno.

Un agente es un sistema en sí mismo, y un elemento que tiene objetivos concretos que cumplir dentro de un entorno, el cuál puede ser complejo y dinámico, un agente puede percibir su ambiente, y actuar sobre él. Un robot que interactúe con el ambiente puede ser considerado un agente, así como también programas de software que cumplan tareas específicas dentro de un macro-sistema.

La conducta de los agentes que interactúan en un sistema complejo adaptativo, puede ser modelada en base a los siguientes criterios:

- Autonomía: lo que implica que cada agente actuará independientemente, sin la dirección de un control central.
- Emergencia: se presentan conductas espontáneas emergentes.
- Adaptación: los agentes pueden cambiar su comportamiento, como respuesta a los cambios del entorno y de su estado interno.
- Auto-organización: son capaces de organizarse para cumplir sus metas.

Los tipos de conducta que pueden manejar los agentes son:

- Conducta primitiva o irreflexiva: es el comportamiento de un agente, que está predefinido por un conjunto de reglas, y que se presenta ante un determinado estimulo.
- Conducta emergente: es el comportamiento de uno o más agentes que puede surgir de forma imprevista, y que por lo tanto no estaba pre programada.
- Conducta con propósito: es la conducta que permite a un agente alcanzar sus objetivos.

- Conducta emergente con propósito: es la conducta emergente que esta enfocada en alcanzar una meta.

Los factores de adaptación para los agentes pueden ser:

- Adaptación pre programada, es decir, que se programan las reglas que permitirán adaptarse al agente según una condición previamente conocida.
- Adaptación por aprendizaje, en este caso se utilizan métodos de aprendizaje como pueden ser redes neuronales.
- Adaptación evolutiva, cuando el entorno cambia, los agentes pueden evolucionar mediante algoritmos evolutivos que permitan optimizar las capacidades de los agentes para las nuevas condiciones.
- Adaptación emergente, si el entorno cambia los agentes podrían adaptarse de varias formas para cumplir sus objetivos.

Los agentes que cumplen con las condiciones anteriormente descritas se conocen como agentes autónomos adaptativos o agentes inteligentes.

CARACTERÍSTICA	**PROGRAMACION ORIENTADA A OBJETOS**	**PROGRAMACIÓN ORIENTADA A AGENTES**
ELEMENTO BASICO	Objeto	Agente
CARACTERIZACIÓN DE UN ELEMENTO BÁSICO	Variables y funciones miembro	Creencias, decisiones capacidades y obligaciones
INTERACCIÓN	Herencia y envío de mensajes entre objetos.	Mensajes entre agentes, informes, peticiones, ofertas, promesas.
APROPIADO	Modelamiento de sistemas, programación modular.	Desarrollo de sistemas distribuidos, solución de problemas distribuidos.

Tabla 1.1. Comparación entre paradigmas de programación POO y POA.

Ahora podemos hablar específicamente de Agentes de Software, los cuáles son programas de software que cumplen con las propiedades de los agentes descritas anteriormente. Los avances en las ciencias de la computación han tomado el enfoque de los agentes, para desarrollar nuevas técnicas, como son, la Ingeniería de Software Basada en Agentes, y la Programación Orientada a Agentes, que amplían las posibilidades de la Programación Orientada a Objetos. Aunque cabe recalcar que un agente es un objeto y por lo tanto la POO es perfectamente aplicable en agentes.

Los agentes de software son utilizados en simulaciones de SA, éstos incluyen muchas técnicas y características de Alife, así como también técnicas de Inteligencia Artificial que permiten que los agentes se comporten de forma autónoma e “inteligente”.

Los agentes serán las entidades que interactuarán con otras semejantes para formar la Sociedad Artificial.

1.2.4 INVESTIGACIONES SOBRE SOCIEDADES ARTIFICIALES (SA)

A continuación nombraremos algunos de los trabajos más destacados que se han realizado en el campo de las SA alrededor del mundo.

El científico Mitchel Resnick en su libro *Turtles, Termites and Traffic Jams,* en 1994, mostró como el comportamiento de termitas y hormigas, así como también atolladeros de tráfico, podían ser representados en un *autómata celular* mediante reglas sumamente sencillas, demostrando así que cuando los individuos de una colectividad siguen reglas simples, todo el sistema se comporta de una forma compleja que muestra la interacción social a gran escala.

Un trabajo clásico en el estudio de las SA fue el de los científicos Joshua M. Epstein y Robert Axtell en 1996, en dicho trabajo se utilizó un *autómata celular,* conocido como *SUGARSPACE*, que a partir de ciertas reglas permitió el

surgimiento de complejos procesos de interacción, tales como la migración, la guerra e inclusive la evolución cultural. *Sugarspace* puede ser considerado como un mundo artificial poblado de agentes donde crecen “cañas de azúcar” a ritmos variables, los agentes tienen varias características como metabolizar el azúcar, visión, salud, velocidad. El comportamiento de los agentes está dado por reglas simples, los agentes viven recolectando azúcar y consumiendo un poco de lo que almacenan, los agentes se desplazan a sitios donde pueden ver que hay altos niveles de azúcar, si dos agentes se encuentren en buen estado y están juntos, entonces pueden reproducirse, el hijo resultante de esta unión heredará las características de sus padres. Con éstas sencillas reglas y algunas variaciones se han hecho estudios de comportamientos emergentes en sociedades artificiales.

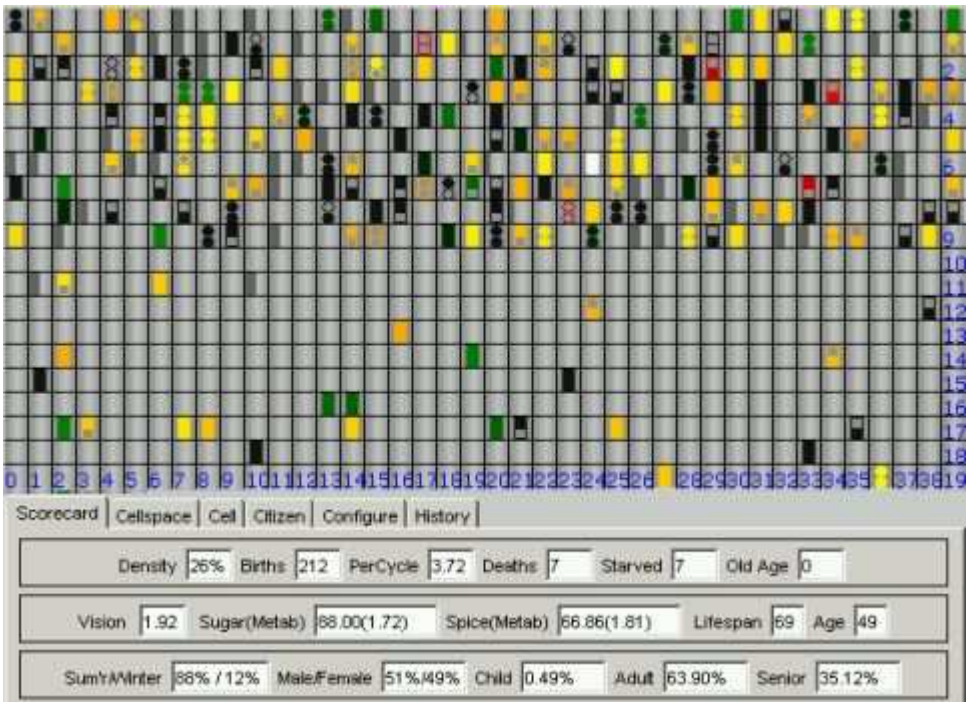


Fig. 1.17. Sugarspace

Jim Doran en 1998 utilizó una simulación basada en agentes, para estudiar las ideologías, en sociedades humanas. El mismo año Dwight Read por medio de SA ha estudiado la relación entre Cultura y comportamiento.

En el año 2004 la comisión europea para la investigación de tecnologías futuras y emergentes financia uno de los proyectos más ambiciosos en el campo de las

sociedades artificiales, el proyecto *New and Emergent World models Trough Individual, Evolutionary, and Social learning (NEW TIES),* el objetivo del proyecto es desarrollar una sociedad artificial sumamente compleja donde se producirán propiedades emergentes como el lenguaje y la cultura entre otras. El proyecto es de código abierto y cualquier persona puede colaborar en su desarrollo.

El mundo del arte y del entretenimiento también se ha visto influido por las SA, en base a métodos gráficos una SA puede convertirse en una imagen, video o inclusive con algoritmos especiales música. De esta forma el mundo del multimedia se ve enriquecido con las SA. Por ejemplo en la figura 1.18 podemos observar la complejidad gráfica y artística de una obra de arte hecha con tecnología de sociedades artificiales.

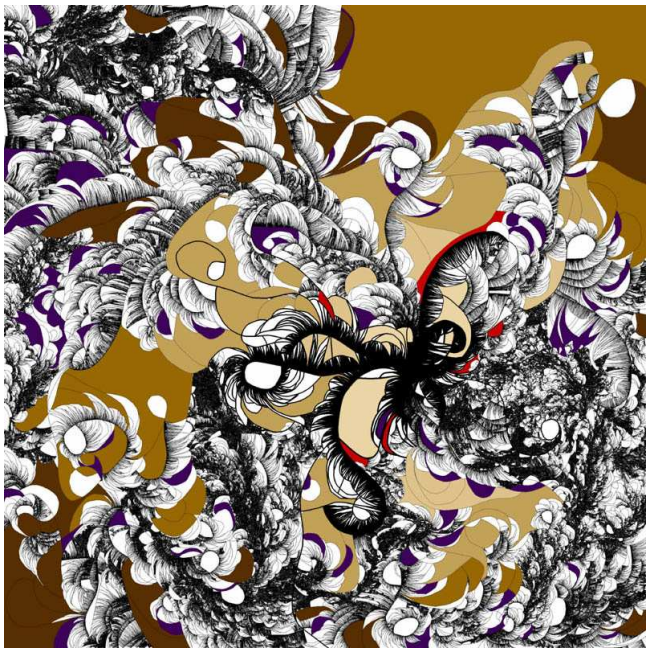


Fig. 1.18 Obra de arte moderno realizado mediante sociedades artificiales.

Las sociedades artificiales tienen varios campos de aplicación, y actualmente se utilizan en campos específicos de experimentación, por ejemplo se puede crear una sociedad artificial para estudiar las dinámicas de la guerra, el comportamiento de personas ante una situación determinada, se puede realizar estudios de mercado y tendencias en la población, en fin las posibilidades son muy bastas. Por ejemplo en la figura 1.19 podemos observar una simulación multi–agente que

representa el comportamiento de una muchedumbre ante un incendio, dentro del piso de un centro comercial. Cada punto es una agente que representa un ser humano, se puede observar la estructura del piso y el amontonamiento que ocurre en las zonas de evacuación, en base a este estudio se podría diseñar la arquitectura del lugar, así en caso de incendio la evacuación del sitio será más fácil para prevenir bajas humanas.

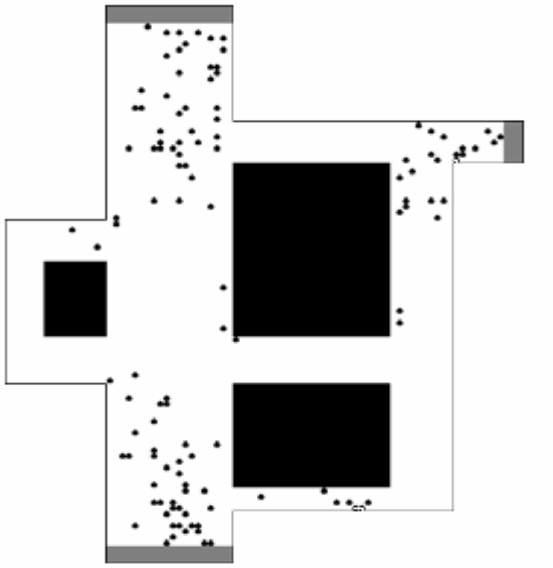


Fig. 1.19. Aplicación de una sociedad artificial.

1.2.5 LA SOCIEDAD HUMANA

Como hemos observado en las secciones precedentes, las Sociedades Artificiales pueden emular una sociedad de entidades artificiales, organismos o animales, en la que se representa funciones básicas de los seres vivos como son: reproducirse, evolucionar, alimentarse, descansar y otras similares, sin embargo cuando hablamos de una sociedad humana como tal, la complejidad se incrementa de gran manera, ya que la interacción de los seres humanos tiene un gran número de variables, lo cual hace sumamente complejo la tarea de representar una SA que represente todos los aspectos de una sociedad humana, sin embargo, cabe mencionar que se han realizado estudios para analizar y simular algunos de los aspectos de las sociedades humanas de forma aislada.

Se puede conceptuar una sociedad humana como un sistema que abarca la reunión o agrupación de seres humanos, familias, pueblos o naciones los cuáles

interactúan entre sí. Puesto que el presente proyecto de titulación se concentrará en lo que es una sociedad humana, es necesario indicar cuáles son los grandes rasgos que la definen, para de esta forma tratar de interpretar, modelar y diseñar las características principales que tendrá la Sociedad Artificial que nos hemos propuesto simular. Para esta tarea nos apoyaremos en la ciencia de la Sociología.

1.2.5.1 La Sociología como marco referencial para el desarrollo de una Sociedad Artificial humana

La sociología es la disciplina que estudia el origen, desarrollo y estructura de las sociedades humanas y la conducta de los individuos y grupos dentro de las mismas. La sociología se apoya en otras áreas del conocimiento como son la historia, la psicología social, la demografía, la antropología, la estadística, entre otras, sin embargo sus principales métodos se basan en la observación directa del fenómeno o acontecer social y en el análisis estadístico de registros, censos y otras fuentes de información que permitan determinar causas y efectos de los acontecimientos sociales, para de esta forma poder inferir leyes mediante análisis deductivos e inductivos. Sin embargo la complejidad de la conducta humana no permite formular leyes de exactitud matemática, como en las ciencias exactas, pero esto tampoco implica que la sociología no sea una ciencia seria, ya que permite realizar predicciones y análisis de comportamientos sociales que se acercan bastante a la realidad.

En base a los criterios sociológicos podemos enumerar las siguientes características como factores imprescindibles al momento de considerar una sociedad humana:

- La cultura
- El comportamiento individual y colectivo
- Población
- Estructura social
- Cambio social

1.2.5.2 La Cultura

La cultura puede ser considerada como el conjunto de rasgos distintivos ideológicos y materiales que caracterizan a una sociedad humana en un momento determinado. La cultura puede ser considerada como el *alma* de una sociedad, esto incluye el modo de vida, las ceremonias y tradiciones, el sistema de valores y la tecnología, el lenguaje, entre otros factores.

El origen y desarrollo de una cultura, es un aspecto sumamente complejo, si consideramos el origen de las primeras culturas humanas conocidas, ya que implica el aparecimiento de conductas y condicionamientos en los hombres, en relación con el medio ambiente, el clima, la geografía y otros grupos de seres humanos que hayan podido influir en el aparecimiento de ciertas características culturales. En base a las investigaciones sostenidas sobre el tema, cabe mencionar que el origen de la cultura está íntimamente vinculado con los procesos de aprendizaje de los seres humanos, los cuáles son capaces de compartir sus experiencias entre ellos, y su descendencia lo cual enriquece el proceso de adquisición de conocimientos. A continuación nombraremos factores importantes en la conformación de una cultura:

- Aprendizaje, adaptación y lenguaje
- El medio ambiente geográfico
- Factores biológicos de los seres humanos

1.2.5.2.1 Aprendizaje, adaptación y lenguaje

Como se mencionó anteriormente la capacidad de aprender y adaptarse de los seres humanos han sido uno de los factores principales para el surgimiento de las culturas, puesto que han permitido que el conocimiento adquirido por una generación determinada no se pierda en el olvido, de forma que sea transmitido a las siguientes generaciones, para lo cual es necesario el proceso del lenguaje.

El lenguaje es una característica propia de los seres humanos que ha permitido formar a las culturas, puesto que es la principal forma de comunicación que

poseen los seres humanos, el estudio de los lenguajes permite dar luz sobre aspectos ocultos del origen de las culturas y su relación con culturas muy distantes en el tiempo y el espacio.

1.2.5.2.2 El medio geográfico

Ha sido objeto de varios estudios el impacto y la relación que tiene un entorno o ambiente geográfico con el aparecimiento de la cultura, para este factor es de importancia la consideración de características tales como:

- El clima y las estaciones, factor importantísimo que regula ciclos de actividades en las sociedades.
- La latitud y longitud geográfica, el emplazamiento físico de las sociedades y su relación con otras sociedades geográficamente cercanas.
- La disponibilidad de recursos, como minerales, recursos hídricos, suelos fértiles entre otros que propicien o inhiban la subsistencia, el comercio u otros factores.
- El ecosistema, es decir la interacción de la flora y fauna, que influye en el desarrollo de la cultura.

1.2.5.2.3 El factor biológico

Por medio del proceso de evolución biológica, los seres humanos se han adaptado a distintos medios geográficos, con el aparecimiento de distintos grupos raciales, que poseen características propias, estas características pueden influir en el aparecimiento de rasgos culturales determinados. Si bien por razones éticas y en base a algunas investigaciones se sostiene que todos los seres humanos somos iguales sin importar la raza, la verdad es que existen investigaciones científicas serias que indican que las razas están mejor adaptadas biológicamente a ciertos ambientes y por lo tanto muestran habilidades naturales en actividades distintas.

Todos los factores anteriormente mencionados contribuyen al aparecimiento de las culturas y sus rasgos específicos, sin embargo es muy difícil para la ciencia

sociológica la predicción y análisis del aparecimiento o conformación de una cultura, debido a la gran cantidad de variables que contribuyen al desarrollo de la civilización y cultura. La ciencia que se encarga del estudio de la cultura humana es la antropología.

1.2.5.3 El comportamiento individual y colectivo

La conducta individual y colectiva están íntimamente relacionados, puesto que la una influye sobre la otra y viceversa, el individuo es influido por el medio social, así como también los individuos influyen sobre la sociedad en mayor o menor grado.

1.2.5.3.1 El comportamiento individual y la personalidad

Los estudios demuestran que la personalidad de los individuos, es decir los rasgos de pensamiento y conducta que caracterizan a un ser humano, está determinado por tres factores principales:

- Herencia biológica: este es el factor natural dado por las características genéticas hereditarias del individuo.
- El ambiente: es decir el medio social que influye sobre el individuo, así como la cultura, la familia, los grupos e instituciones sociales.
- Las experiencias: es decir todas las vivencias que el individuo tiene a lo largo de su vida; especialmente son importantes en la formación de la personalidad, lo que el individuo aprende en sus primeros años de vida, ya que estas experiencias son generalmente las que marcan el comportamiento para el resto de la vida.

1.2.5.3.2 El comportamiento colectivo mediante los grupos sociales

El ser humano es por naturaleza un ser gregario, que tiende a unirse a otros de su misma especie para agruparse, ya sea por seguridad y subsistencia, pero también por afinidad de ideas, sentimientos, intereses o cualquier otro factor, es bajo este aspecto que el concepto de *grupo social* es un concepto de suma

importancia para la sociología, puesto que dentro de una sociedad dada existirán grupos con expectativas y conductas propias que interactúen con otros grupos.

Estos grupos norman muchas veces las conductas de sus miembros, estableciendo los lineamientos generales de comportamiento que un individuo llevará en la sociedad, pues caso contrario el grupo o la misma sociedad lo rechazará; a la interacción entre los miembros de un grupo social se conoce como *proceso social.*

1.2.5.3.3 Cooperación, Competencia y Conflicto en los procesos sociales

En base a los estudios biológicos sobre la *supervivencia del más apto*, se ha llegado a pensar que la *competencia* es la única regla para la evolución y subsistencia de los seres vivos, pero esto es un error de conceptualización, puesto que la *simbiosis* en la naturaleza es la *cooperación* de las especies en un ecosistema para beneficiarse mutuamente y poder sobrevivir, es así como junto a la *competencia*, la *cooperación* es un factor imprescindible de la subsistencia y la evolución de los seres vivos.

La sociedad humana no está al margen de estos acontecimientos, es así como la cooperación y la competencia son los dos procesos básicos de la convivencia en grupo.

La cooperación permite a los seres humanos ayudarse mutuamente para alcanzar objetivos comunes y beneficiarse ampliamente entre todos para vencer obstáculos y limitaciones.

La competencia es un factor igualmente natural que aparece cuando la demanda supera a la oferta, es decir cuando los recursos o bienes que se desea son limitados, lo cual producirá *conflictos* entre los individuos o grupos que busquen poseer el recurso. El objetivo de la competencia es el de vencer al competidor u oponente.

Las sociedades pueden favorecer la cooperación o la competencia entre los individuos, sin embargo estos son factores que existirán en toda sociedad en un mayor o menor grado.

Debemos mencionar que generalmente los grupos sociales incentivan la unión y colaboración entre sus miembros, pero en cambio, se muestran hostiles hacia otros grupos con los cuales pueden competir, esto no es una ley, sin embargo se muestra que es una tendencia general.

Las fuerzas que mueven a la sociedad pueden ser asociadas con las fuerzas de atracción y repulsión de la física, "la organización social de una comunidad en un tiempo dado representa el equilibrio entre las fuerzas centrífugas y centrípetas"[5]

1.2.5.4 Población

La población de una sociedad determina el número de seres humanos que habitan en una determinada área geográfica, y éste es un factor de suma importancia que condiciona los procesos sociales. Para la consideración de este factor, consideraremos las siguientes características:

- Las comunidades
- La distribución de la población
- Variación de la población
- Índices demográficos

1.2.5.4.1 Las comunidades

Se puede entender la comunidad como la agrupación organizada de los seres humanos dentro de un área determinado, esto se relaciona íntimamente con los poblados, urbes y metrópolis, y por supuesto con el urbanismo, es decir la forma como se dispone la población en un área específica de terreno.

[5] William F. Ogburn, Meyer F. Nimkoff: “Sociología”; Pág. 213

1.2.5.4.2 Distribución de la población

Los seres humanos se han desplazado a través de los tiempos en las diversas regiones geográficas del mundo, alternando entre estilos de vida sedentarios y nómadas, estos procesos migratorios han determinado como se concentran los seres humanos, y por lo tanto las sociedades y las culturas en las áreas geográficas.

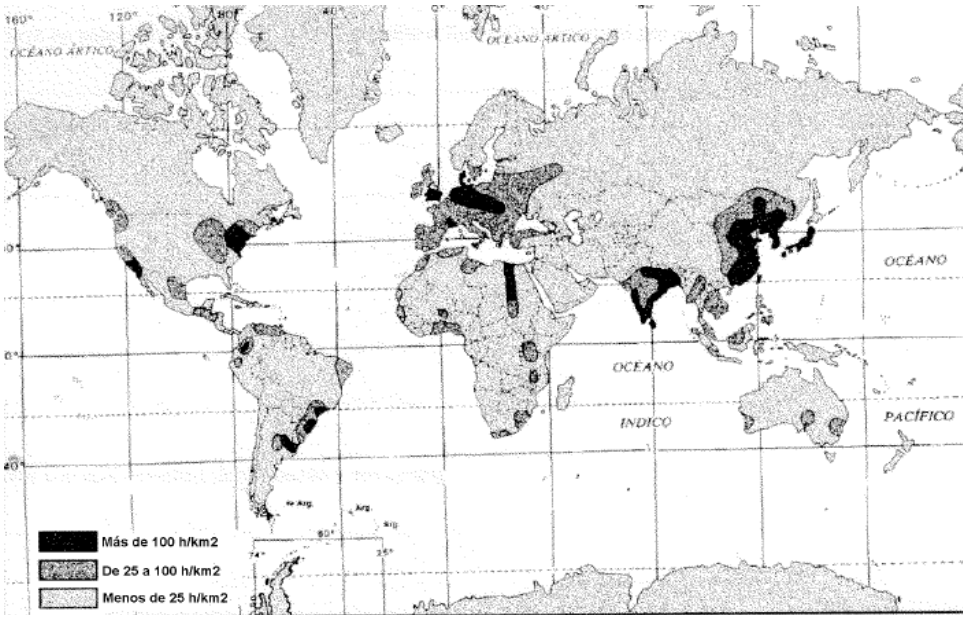


Fig. 1.20. Mapa que indica la distribución de la población a nivel mundial en el año 2000.

La distribución de la población se puede considerar por áreas, regiones o comunidades, los principales índices para determinar la distribución de la población en un área son las siguientes:

- Población Absoluta*:* que es el número total de habitantes de un área determinada.
- Densidad de población*:* que indica el número de habitantes por unidad de superficie, de la siguiente forma:.
  Densidad de población = Población Absoluta / superficie total del área

1.2.5.4.3 Variación de la población

Las poblaciones pueden crecer y decrecer numéricamente en un periodo de tiempo, debido a las migraciones, nacimientos y defunciones, como parámetros de variación de las poblaciones se consideran:

- La migración, es decir los seres humanos que salen de una comunidad o *emigrantes,* y las personas que llegan a una comunidad o *inmigrantes.*
- El índice de natalidad: factor que indica el número de nacimientos en una determinada población en un periodo de tiempo, este índice calcula el total de nacidos vivos por cada 1000 habitantes en un año.
- El índice de mortalidad: factor que indica el número de fallecimientos en una población para un periodo de tiempo, este coeficiente calcula el total de fallecidos por cada 1000 habitantes en un año.

1.2.5.4.4 Índices demográficos

La ciencia que estudia a las poblaciones humanas es la Demografía, por lo tanto los índices sociales y económicos que determina esta ciencia son de importancia para la sociología, además de los índices mencionados anteriormente, existe una gran variedad de indicadores demográficos, sin embargo los más importantes se agrupan en las siguientes características:

- Nivel de Vida: información sobre el nivel de bienestar material que tienen los habitantes de una región determinada.
- Esperanza de vida: datos sobre el promedio de vida que una persona puede tener en un lugar y tiempo determinados.
- Matrimonios: información sobre los matrimonios efectuados.
- Delincuencia: datos sobre las agresiones, asaltos y otras actividades delictivas y antisociales.
- Salud mental y física: estadísticas sobre las enfermedades mentales y físicas que pueden existir en una población.
- Sexo y Edad: estadísticas sobre el sexo y la edad de las personas.

1.2.5.5 Estructura Social

La estructura social hace referencia a la forma en la que se organizan las sociedades para su existencia. Para considerar la estructura y organización social podemos considerar:

- La familia
- Castas y clases sociales
- Gobierno

1.2.5.5.1 La familia

La familia ha sido considerada la célula de la sociedad, es decir el grupo social básico que existe en toda sociedad, la familia se forma generalmente por vínculos sociales como el matrimonio, el parentesco u otros y conlleva muchas veces la reproducción de la especie, para que dentro del ambiente familiar las crías humanas puedan crecer y desarrollarse. La estructura o conformación de una familia puede cambiar según la sociedad, como un ejemplo, la típica familia occidental del siglo XXI es la que está compuesta del padre, la madre y los hijos.

1.2.5.5.2 Castas, clases sociales y jerarquía social

A través del proceso de organización social los seres humanos han establecido el *estatus* es decir el rango o posición que ocupa cada ser humano en la sociedad, debido a varios factores como puede ser la riqueza, el talento, la nobleza, entre otros.

Estos factores han determinado que unos seres humanos gobiernen sobre otros, dando paso a lo que se conoce como *Castas,* la cuál es una estructura jerárquica social muy rígida que no permite que sus miembros cambien de jerarquía y que generalmente es hereditaria, como ejemplo tenemos el sistema de castas de la india. También los factores anteriores han determinado lo que se conoce como *clases sociales,* las cuales también son una estructura social jerárquica, pero permiten la movilidad de los individuos de una clase a otra, como ejemplo tenemos las clases sociales occidentales modernas.

1.2.5.5.3 Gobierno

Como parte de la organización, las sociedades han adquirido diversas formas para controlar a sus elementos, naciendo así los gobiernos que rigen sobre los elementos de una sociedad, las formas de gobierno han variado de muchas maneras, dependiendo mucho de la cultura.

1.2.5.6 Cambio social

La interacción de los individuos y grupos en la sociedad produce constantes cambios sociales, los cuáles pueden tener infinidad de causas e innumerables consecuencias, sin embargo podemos nombrar algunas de las características que influyen en los cambios sociales:

- Transformación cultural
- La tecnología
- Economía y comercio
- Cultos y religiones
- Guerras y revoluciones
- Desorganización social

1.2.5.6.1 Transformación Cultural

La acumulación de conocimientos, así como la interacción entre sociedades ha permitido que las culturas crezcan y prosperen hasta alcanzar grados de esplendor como en las grandes civilizaciones del pasado, sin embargo la historia nos muestra que también hay periodos de involución y destrucción de la cultura y civilización.

1.2.5.6.2 Tecnología

Como un factor sumamente importante de la cultura de una sociedad se encuentra el nivel tecnológico de una sociedad, que determinará cambios en la estructura social, así como en la cultura misma. Dentro de este campo podemos

incorporar los medios de comunicación y transporte de una sociedad que permiten el incremento de interacción con otras sociedades.

1.2.5.6.3 Economía y comercio

La economía, el comercio, los medios de producción han sido factores vitales de la existencia y transformación de las sociedades, ya que sin estos, la misma subsistencia de los pueblos se hubiera visto afectada.

1.2.5.6.4 Cultos y religiones

A través del tiempo los seres humanos han creado doctrinas, mitos, rituales y leyendas, con lo cual han aparecido los diversos cultos, dogmas y creencias como una parte fundamental de las sociedades. Los cultos y religiones son un aspecto vital de las culturas y tiene una fuerte implicación psicológica sobre cada ser humano que sea parte del culto. Muchos procesos de cambio y reestructuración social son debidos a este factor.

1.2.5.6.5 Guerras y revoluciones

Cuando nos referíamos a la competición y conflicto entre grupos sociales, este puede desencadenarse en un conflicto armado de pequeña o de gran escala, por lo tanto, la organización social ha creado muchas veces instituciones especializadas para la guerra. La guerra es un conflicto que involucra a dos o más grupos sociales, perjudicial para la economía, la vida humana y en general para toda la estructura social. Las revoluciones son conflictos que cambian la estructura social muchas veces desde el mismo interior del grupo social.

1.2.5.6.6 Desorganización Social

La desorganización social es un proceso por el cual se rompe el equilibrio de la organización social de forma nociva para la cultura, algunos de los factores de desorganización social son:

- Pobreza: factor que puede ser considerado como la carestía de recursos que impide el desarrollo de una sociedad.

- Delincuencia: factor que pone en peligro el orden y control social, perjudicando a sus elementos.
- Enfermedades colectivas: tales como pestes, enfermedades infecto-contagiosas a pequeña o gran escala que pueden poner en peligro a las sociedades.
- Conflicto Cultural: cuando un nuevo elemento cultural ingresa a una sociedad, este puede hacer tambalear los principios en que se basa la organización social.

1.2.6 ELEMENTOS DE LA SOCIEDAD HUMANA QUE PUEDEN SER REPRESENTADOS MEDIANTE VIDA ARTIFICIAL

Hemos bosquejado a grandes rasgos las características generales de una sociedad humana, sin embargo como se puede analizar la complejidad de la simulación social radica generalmente en el surgimiento y desarrollo de la cultura, el estudio de la cultura a través del tiempo y espacio es tarea propia de la Antropología.

La mayoría de elementos de la sociedad humana está dada por la cultura, puesto que la cultura forja elementos tales como el lenguaje, las creencias, comportamientos de los seres humanos, la cultura también influye en la forma de producción, en la forma de gobierno, la estructura social, en la tecnología, entre otros.

Por ejemplo una cultura nómada y una sedentaria, la primera se dedica a la obtención de recursos en la forma de caza, pesca y recolección de alimentos, generalmente desconoce la agricultura y la ganadería, la segunda- la sedentaria- conoce la agricultura y/o la ganadería, como vemos las dos sociedades son humanas, sin embargo podemos observar que la cultura determina la forma en que se obtienen los recursos, así las variaciones en las sociedades humanas son infinitas en todos los ámbitos analizados, según el tiempo y el lugar.

De ahí que los factores culturales que puedan surgir en una sociedad artificial son propiamente otro ámbito conocido como Cultura Artificial, la cual demanda una gran complejidad cognitiva de los agentes que formarán esa Cultura Artificial, es decir la complejidad en la inteligencia artificial de los agentes de esa sociedad artificial debe ser lo suficientemente elevada para que los agentes generen lenguaje y otros aspectos propios de la Cultura.

En nuestro trabajo no vamos a realizar Cultura Artificial, sino que representaremos la sociedad artificial con los elementos que pueden ser sintetizados simplemente con ALife, para esto podemos extraer los elementos comunes que son comunes a las sociedades humanas sin importar el tiempo, el lugar y la cultura.

Los factores generales que hemos considerado son:

- El ser humano es un ser gregario por lo tanto social.
- Los hombres se reproducen con las mujeres para procrear la especie.
- Los padres transmiten la herencia biológica a sus hijos.
- El ser humano necesita recursos que consume para su subsistencia.
- El ser humano cumple un ciclo de vida.

Con estos sencillos elementos se puede generar comportamientos emergentes en una sociedad artificial, que represente elementos más complejos de una sociedad.

Los distintos aspectos de la sociedad humana que hemos analizado merecen un estudio particular y son actualmente investigados específicamente por científicos e investigadores para poder ser simulados y modelados.

1.2.7 CULTURA ARTIFICIAL

Es necesario mencionar la Cultura Artificial, puesto que la complejidad de la simulación social humana necesitará una mayor investigación en esta área.

Con respecto a la complejidad de la simulación o síntesis de la cultura de una sociedad humana, la más representativa e importante idea al respecto es la Cultura Artificial, término acuñado por Nicholas Gessler en el simposio ALIFE IV en 1994.

La Cultura Artificial está profundamente relacionada con Alife, con la Antropología que es la ciencia que estudia las culturas humanas y sus subdisciplinas como son la arqueología, la etnografía, la etnología.

En base a estudios anteriores[6] la Cultura Artificial considera un *patrón universal de cultura* que posee los elementos y estructura indicados en la figura 1.21.

La cultura es una propiedad emergente de la sociedad humana, así mismo en una Sociedad Artificial la Cultura Artificial debe ser una característica emergente.

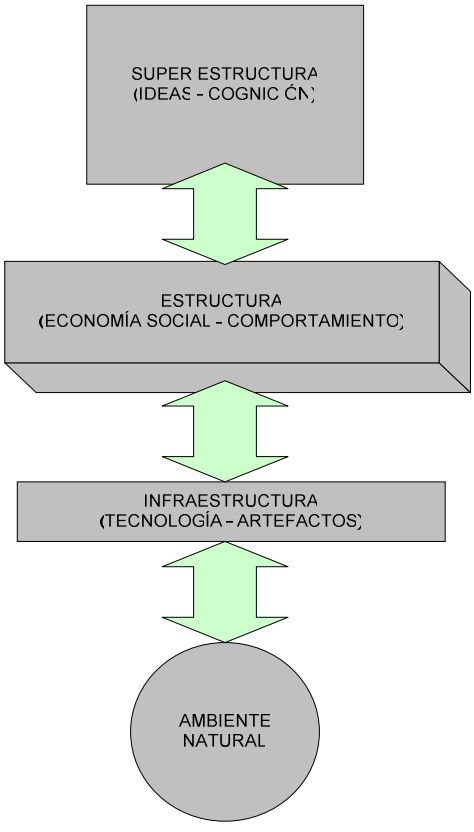
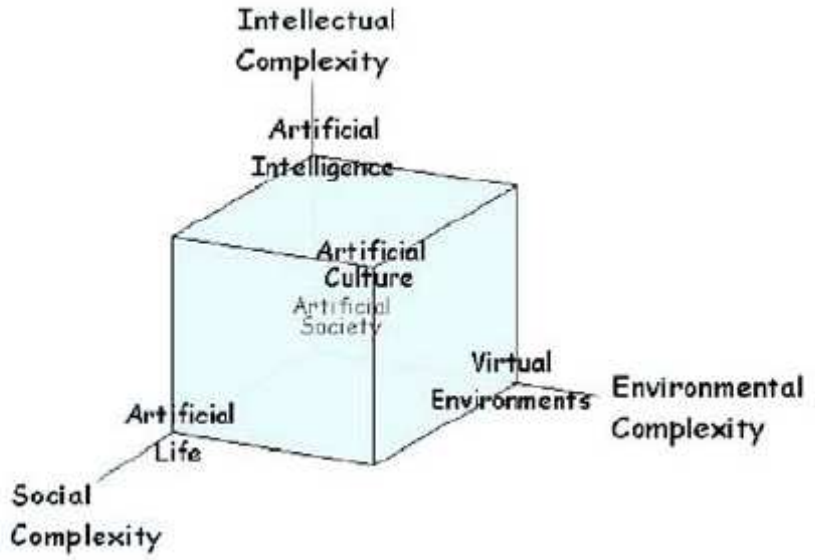


Fig. 1.21 Diagrama esquemático del patrón universal de cultura artificial

[6] Harris, Marvin 1991. *Cultural Anthropology*. Third edition. New York: Harper-Collins.

Gessler ha determinado[7] los ejes que podrían considerarse para el surgimiento de una cultura artificial emergente dentro de una sociedad artificial, como se puede observar en la figura 1.21 la complejidad cognitiva de los agentes que intervienen en la simulación es el principal factor. Podemos observar que Gessler considera la vida artificial, la inteligencia artificial, y los ambientes virtuales como los elementos que permiten generar una sociedad artificial.



Son Fig. 1.17 Ejes de complejidad de una Cultura Artificial emergente.

[7] Gessler, Nicholas. 2003. “Evolving Cultural Things-That-Think.” In Computational Synthesis: From Basic Building Blocks to High Level Functionality.

CAPITULO II: DESARROLLO DE LA APLICACIÓN

2.1 METODOLOGÍA DE DESARROLLO

Existen muchos paradigmas de procesos para el desarrollo del software, cada metodología tiene sus características particulares que lo hacen más o menos apropiado para ciertos tipos de aplicación.

En nuestro caso el software que se desea producir es un sistema de simulación basado en agentes para recrear una sociedad artificial, para seleccionar adecuadamente el proceso de desarrollo realizaremos una comparación entre algunos enfoques importantes para seleccionar el más apropiado.

Paradigma	Modelo de proceso	Apropiado para:	Fases-Actividades
Proceso Unificado (PU)	Iterativo e Incremental	Se adapta a cualquier proyecto, especialmente sistemas de información, pero enfocándose en la arquitectura.	Inicio Elaboración Construcción Transición
CleanRoom	Formal Matemático	Aplicaciones científicas y de ingeniería donde la confiabilidad es crítica	Análisis y diseño formales Implementación iterativa Pruebas Estadísticas
DSDM (Método dinámico de desarrollo de	RAD, desarrollo ágil	Sistemas de Información donde el tiempo y el presupuesto son muy	Estudio Modelo funcional Diseño Construcción

sistemas)		limitados	Implementación
ICONIX	Desarrollo ágil, iterativo	Varios tipos de proyectos concentrándose en el análisis y diseño por medio de UML y casos de uso.	Requerimientos Diseño preliminar Diseño detallado Despliegue.
Programación Extrema XP	Desarrollo ágil, iterativo.	Proyectos donde los cambios de requerimientos son continuos y constantes	Codificación Pruebas Retroalimentación Diseño
MSF (Microsoft Solutions Framework)	Iterativo, incremental, evolutivo	Varios tipos de proyectos, la metodología permite adaptarse al proyecto, especialmente para equipos de desarrolladores.	
PSP (Proceso de Software Personal)	Modelo de madurez de capacidad, iterativo.	Se enfoca en la prevención de defectos durante el desarrollo del software.	Recolección de datos Estimación y planificación Administración de defectos y resultados Control de calidad

Tabla 2.1. Comparación de frameworks de procesos para desarrollo de software.

Para desarrollar un software que permita una simulación basada en agentes para recrear una sociedad artificial, el factor más importante a considerar es la arquitectura del sistema, puesto que el modelamiento y diseño del software permitirá establecer funcionalidades específicas que integradas de forma adecuada permitan a los subsistemas integrarse como un todo para formar el software deseado.

Al concentrarnos en la arquitectura, realizamos un correcto trabajo de ingeniería, es decir enfocarnos en el análisis y diseño para crear los planos y documentos del sistema que establezcan la eficiencia, calidad, sencillez y modularidad que un buen software debe poseer. Una arquitectura correctamente definida nos permite dar un mantenimiento adecuado al software, ampliando y cambiando sus características según las necesidades.

La metodología de proceso de desarrollo que se enfoca principalmente en la arquitectura es el PROCESO UNIFICADO, además que su flexibilidad permite adaptarse a todo tipo de proyecto.

La Metodologías como ICONIX, CLEANROOM, MSF, PSP podrían ser utilizadas también sobre este proyecto, puesto que también consideran en mayor o menor grado la arquitectura, sin embargo el Proceso Unificado es un intento por unificar practicas exitosas de ingeniería de software lo cual es apropiado para este proyecto de titulación. Ahora bien el Proceso Unificado (PU) ha sido causa de varias adaptaciones y refinamientos, por lo tanto es necesario revisar brevemente algunas de las variaciones más conocidas del PU para establecer si una de ellas es más propicia al presente proyecto.

Variaciones	Características	Disciplinas
IBM RUP (Proceso Unificado Racional)	Es una especificación detallada del Proceso Unificado, un proceso adaptable para todo tipo de proyecto.	Modelamiento del negocio Requerimientos Análisis Diseño Implementación Pruebas Despliegue Administración de la configuración Adm. del proyecto Ambiente o Entorno

Proceso Unificado Ágil	Es una versión simplificada de RUP que hace énfasis en el desarrollo ágil de software.	Modelación Implementación Pruebas Despliegue Administración de la configuración Administración del proyecto Ambiente.
EUP (Enterprise Unified Process-Proceso Unificado Corporativo)	Es una extensión del RUP específicamente para software corporativo.	Todas las disciplinas del RUP y además: Modelamiento de negocio corporativo Administración de portafolio Arquitectura corporativa Reutilización estratégica Manejo de personal Administración empresarial Mejoramiento del proceso de software
Open UP OpenUP/Basic	Descarta las características opcionales de RUP, por lo cual es un proceso ágil e informal. Apropiado para proyectos pequeños y medianos, así como proyectos Open Source.	Requerimientos Arquitectura Desarrollo Pruebas Administración del proyecto Administración de la configuración y del cambio
EssUP (Proceso Unificado Escencial)	Revisión del RUP, con mayor flexibilidad y capacidad para integrar nuevos paradigmas.	

Tabla 2.2. Información de las variaciones del Proceso Unificado

El presente proyecto es pequeño, sin embargo al ser un software que trabaja con inteligencia artificial es necesario enfocarse en la arquitectura, por eso he pensado que la mejor opción es el Proceso Unificado Ágil (Agile UP) el cual combina lo mejor del paradigma ágil de desarrollo de software y la formalidad de RUP.

Por lo tanto en nuestro proyecto utilizaremos la metodología del Proceso Unificado Ágil.

2.1.1 BREVE EXPLICACION DEL PROCESO UNIFICADO AGIL

El proceso unificado ágil posee cuatro fases:

- **Inicio:** El objetivo es determinar el alcance del proyecto y la potencial arquitectura del sistema.
- **Elaboración:** El objetivo es modelar el sistema.
- **Construcción:** El objetivo es construir el software.
- **Transición:** El objetivo es probar y desplegar el sistema.

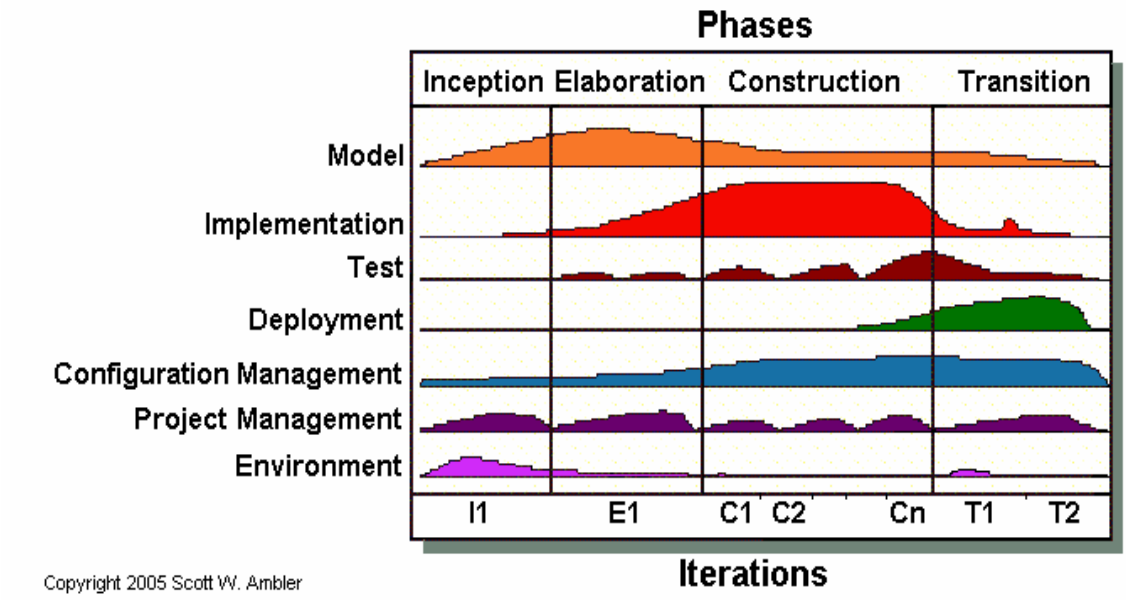


Fig. 2.1. Fases y Disciplinas de Agile UP

Las disciplinas del proceso unificado ágil son las siguientes:

- **Modelamiento:** El objetivo es entender el dominio del problema e identificar la solución viable.
- **Implementación:** El objetivo es transformar el modelo en código ejecutable y realizar pueblas apropiadas.
- **Pruebas:** El objetivo es garantizar la calidad del producto así como verificar que se cumple con los requerimientos.
- **Despliegue:** El objetivo es ejecutar un plan para hacer el sistema disponible a los usuarios.
- **Administración de la configuración:** El objetivo es administrar los productos de trabajo.
- **Administración del proyecto:** El objetivo es dirigir las actividades que tienen lugar en el proyecto.
- **Ambiente:** El objetivo de esta disciplina es proveer y establecer las herramientas, estándares, medios de comunicación, etc. que necesita el equipo de trabajo para cumplir sus metas.

Las actividades relacionas con cada fase son:

Inicio:

- Definir alcance del proyecto
- Estimar costo y cronograma
- Definir riesgos
- Determinar factibilidad del proyecto
- Preparar el ambiente del proyecto

Elaboración:

- Identificar Arquitectura
- Validar Arquitectura
- Desarrollar el ambiente para el proyecto
- Seleccionar el equipo de trabajo

Construcción:

- Modelar, construir y probar el sistema
- Desarrollar documentación de soporte

Transición:

- Pruebas del sistema
- Pruebas de usuario
- Mejorar el sistema
- Despliegue del sistema

2.2 DISEÑO DE LA APLICACIÓN

2.2.1 DEFINICIÓN DEL ALCANCE DEL PROYECTO

Al definir el alcance del proyecto estableceremos lo que hace el sistema a un alto nivel de abstracción, sin entrar en detalles de la implementación, definiendo al mismo tiempo el modelo de simulación basado en agentes, es decir las características de la sociedad artificial.

El presente proyecto realizará una simulación basada en agentes para representar el comportamiento de una sociedad humana, para esto el software permitirá mediante una interfaz gráfica de usuario observar los agentes en interacción en un mundo de 2 dimensiones, en este plano se podrá realizar acercamientos y alejamientos (zoom in, zoom out) para poder observar un mayor o menor detalle de la interacción, el movimiento de los agentes en el mundo 2D se realizará de forma toroidal, es decir que al llegar a un extremo del plano, el agente aparecerá en el extremo opuesto, como una continuidad de la dimensión; en un punto o celda del plano 2D puede encontrarse uno o más agentes al mismo tiempo.

El usuario podrá ingresar parámetros de inicio para empezar una simulación, igualmente podrá controlar algunos parámetros durante la ejecución de la simulación, durante una ejecución el usuario puede pausar la simulación, finalizar la simulación, inicializar otra simulación o grabar el estado de la simulación actual.

Durante la ejecución de una simulación, el usuario tendrá acceso a la información de varios parámetros por medio de la interfaz de usuario, así también el usuario podrá visualizar información estadística detallada de la simulación.

En la simulación cada agente o “humano virtual” nacerá, crecerá, se reproducirá y morirá, para sobrevivir necesitará consumir recursos que estarán dispersos en el mundo 2D, para reproducirse necesitará la disponibilidad de una agente del sexo opuesto, los agentes competirán entre ellos por los recursos.

Pues bien, cada agente tendrá sus propias características que podrá pasar por herencia ‘biológica’ a sus descendientes, a excepción de un factor de mutación que impide que la población converja después de cierto tiempo en características específicas; los aspectos o características propias que tienen los agentes son:

- *Edad:* edad del agente en tiempo del simulador.
- *Sexo:* género del agente.
- *Salud:* capacidad vital o energética que tiene el agente para realizar acciones.
- *Fertilidad:* número de hijos que puede generar un agente a través de su vida.
- *Aptitud de reproducción:* radio de búsqueda donde el agente puede encontrar una pareja disponible para reproducirse.
- *Frecuencia de reproducción:* rapidez con la que el agente puede engendrar.
- *Aptitud de supervivencia:* radio de búsqueda alrededor del agente o vecindario donde se puede encontrar recursos para sobrevivir.

Los recursos naturales que los agentes necesitarán para sobrevivir se encontrarán dispersos en el mundo 2D con mayor o menor proporción, los agentes que se encuentran en un rango determinado cerca de los recursos podrán utilizarlos según sus necesidades y sus aptitudes para la supervivencia, los recursos irán disminuyendo conforme se los utilice, pero se establecerá un parámetro para establecer la capacidad de regeneración de dichos recursos, lo cual es una simplificación de la capacidad de producción de las sociedades, así como de los recursos renovables.

La simulación se realizará de forma sincrónica, y el usuario podrá establecer la velocidad de ejecución de la simulación. El usuario puede seleccionar un agente y observar la información de ese agente en un momento dado.

La primera generación de agentes será dispersa en el mundo 2D de forma aleatoria al igual que los recursos los cuales se pueden concentrar o dispersar según un parámetro establecido, la primera generación de agentes obtendrá sus características particulares de forma aleatoria.

Los agentes pueden estar en un momento determinado en un estado interno, los estados en los que se puede encontrar un agente son:

- *Merodeando:* cuando el agente se mueve aleatoriamente en el entorno de celda en celda.

- *Buscando Pareja:* cuando un agente busca en su vecindario aledaño y en base a su aptitud reproductiva una pareja del sexo opuesto disponible (que también busque pareja) para reproducirse. Para reproducirse no es necesario que los dos agentes estén en un mismo sitio físico, sino que el agente solicitado este dentro del radio de aptitud de reproducción del agente que solicita la cooperación para reproducirse. Para reproducirse los agentes deben tener una edad mayor o igual al 30% de la edad máxima permitida para los agentes.

- *Con Pareja*: estado que se presenta cuando ha encontrado una pareja y se ha reproducido, en este estado no puede reproducirse, declinando las propuestas de agentes del sexo opuesto para reproducirse. El agente puede cambiar nuevamente de estado a *Buscando Pareja*, según su frecuencia de reproducción, en este caso puede reproducirse con otro agente. Ya que los matrimonios y uniones entre parejas son una creación cultural, con este modelo tratamos de representar los varios aspectos que se dan en las sociedades humanas como pueden ser Matrimonio, relación casual, poligamia, poliandria, poliginia, promiscuidad y otras que tienen como fin la reproducción de la especie.

- *Buscando Recursos:* estado que se presenta cuando la salud del agente ha llegado al 50% de su total disponible; con cada movimiento el agente gasta un punto de su salud total, al llegar al 50% buscará recursos para consumir lo cual incrementará la salud del agente al 100%. Los agentes tienen distintos niveles de salud. El agente podrá encontrar recursos en un radio alrededor de él mismo igual a la aptitud para sobrevivir o encontrar recursos.

- *Movilizándose hacia recursos:* cuando el agente ha encontrado un recurso disponible se moviliza inmediatamente hacia ese punto del plano para consumirlo, pero si en el viaje hacia ese recurso encuentra otro recurso disponible consumirá este ultimo en vez del primero, de esta forma los agentes pueden ganar el recurso a otros, por encontrase más cerca, o por cuestiones aleatorias.

- *Muerto:* un agente que haya llegado a la edad máxima permitida para los agentes, o su salud haya llegado a cero, está en estado muerto, y será retirado del mundo 2D inmediatamente.

Pues bien, ya que conocemos las características propias de cada agente y sus estados internos, podemos ahora conocer los parámetros generales para controlar el entorno:

- *Dimensiones X Y del entorno 2D:* representan las dimensiones del mundo 2D.
- *Población Inicial:* la población inicial cuyos caracteres serán aleatorios.
- *Población máxima:* límite para la población, al llegar a este punto los agentes no podrán reproducirse.
- *Vida máxima:* límite de vida para todos los agentes, es la esperanza de vida máxima que un agente puede alcanzar.
- *Salud máxima:* límite para la salud de los agentes.
- *Fertilidad Máxima:* es el valor máximo que un agente podría tener de fertilidad.
- *Aptitud reproductiva máxima:* máximo valor que un agente podría alcanzar para la característica *aptitud sexual.*
- *Aptitud supervivencia máxima:* máximo valor que un agente podría alcanzar para encontrar recursos.
- *Mutación:* característica que permite variar los caracteres hereditarios de la población.
- *Distribución de sexo:* porcentajes de la población que tendrán el género masculino y femenino. Puede ser manual o aleatoria.
- *Recurso máximo:* máximo valor de recursos que puede distribuirse en el entorno 2D.
- *Dispersión de los recursos:* determina una mayor o menor concentración de recursos en una misma área.
- *Regeneración recursos:* determina una probabilidad de que los recursos consumidos sean o no regenerados completamente. Los recursos se regeneran aleatoriamente a través del mundo 2D.
- *Frecuencia de regeneración de recursos:* determina la rapidez con la que se regeneran los recursos que consumen los agentes.

Hemos realizado la descripción del modelo basado en agentes que representa una sociedad artificial humana, un modelo es una simplificación de la realidad, como podemos observar esta simplificación se encuentra en un alto grado de

abstracción y generalización, muy lejos de alcanzar la verdadera complejidad de una sociedad humana, sin embargo, el modelo permitirá el surgimiento de complejos comportamientos emergentes a partir de interacciones simples entre los individuos que pertenezcan a una comunidad, éste es pues el alcance del software que procederemos a crear. El nombre del software será simplemente **Sociedad Artificial.**

2.2.2 SELECCIÓN DE LA HERRAMIENTA

La disciplina de ambiente en el proceso unificado ágil define un especialista de herramientas que selecciona la herramienta de programación más adecuada para el proyecto, debido a que en el modelamiento de la arquitectura del sistema es necesario conocer las librerías, clases y componentes que podemos disponer para reutilizar en nuestro software, es necesario antes de realizar el modelamiento arquitectónico, seleccionar la herramienta de desarrollo así como el lenguaje de programación, para esto debemos analizar algunas de las herramientas para programación de sistemas multi-agentes y/o simulación basada en agentes.

La siguiente lista de software es Open Source, excepto Net Logo que es Freeware.

NOMBRE	CARACTERÍSTICAS	URL
A-globe	Permite realizar pruebas de escenarios con agentes basado en java	http://agents.felk.cvut.cz/aglobe
ABLE	Es un framework para java y una librería de componentes que permite trabajar con agentes inteligentes	http://www.alphaworks.ibm.com/tech/able
Ascape	Permite desarrollar y experimentar con modelos basados en agentes	http://ascape.sourceforge.net
Breve	Permite construir simulaciones 3d de sistemas multi-agentes y de vida artificial basado en Python y OpenGL	http://www.spiderland.org/breve
Cormas	Permite desarrollar sistemas multi agentes	http://cormas.cirad.fr/en/o

	pero enfocándose principalmente en la administración de recursos	util/outil.htm
Cougaar	Arquitectura basada en Java que soporta aplicaciones distribuidas de agentes	http://www.cougaar.org/
EcoLab	Sistema de simulación de agentes en lenguaje C++	http://ecolab.sourceforge.net/
JADE	Java Agent Development Framework, Librerias para desarrollo de sistemas multi-agentes en Java.	http://jade.tilab.com/
JAS	Librería para simulación basada en agentes para lenguaje JAVA	http://jaslibrary.sourceforge.net/
MASON	Librería java para simulación multi agentes	http://cs.gmu.edu/~eclab/projects/mason/
metaABM	Meta modelador para modelos basados en agentes para frameworks basados en eclipse	http://www.metascapeabm.com/index.php?option=com_content&task=view&id=19&Itemid=61
REPAST	Herramienta para desarrollo de simulaciones basadas en agentes, para lenguaje java e IDE eclipse	http://repast.sourceforge.net/
SeSAm	Entorno para modelar y experimentar con agentes	http://www.simsesam.de/
SimPy	Lenguaje de programación basado en Python, que permite enfocarse en simulaciones basadas en agentes	http://simpy.sourceforge.net/
Swarm	Plataforma para modelamiento basado en agentes basado en Java y en Objective C	http://www.swarm.org
Zeus	Herramienta para desarrollo de software basado en agentes hecha en Java	http://labs.bt.com/projects/agents/zeus/
NetLogo	Plataforma de modelamiento programable multi agente basada en lenguaje Logo	http://ccl.northwestern.edu/netlogo/

Tabla 2.3. Características de algunas herramientas para simulación basada en agentes.

La mayoría de estas herramientas posee muy poca documentación por lo que dificulta el proceso de desarrollo, o restringe demasiado la creación de un software personalizado, sin embargo las excepciones son JADE (Java Agent Development Framework) y MASON (Multi Agent Simulator of Neghiborhoods), los cuales son librerías Open Source para lenguaje de programación Java que permiten trabajar con agentes, sin embargo la más apropiada para nuestro proyecto es MASON, puesto que se enfoca en la simulación basada en agentes.

Por lo tanto hemos seleccionado a MASON como librería base para la implementación del sistema, MASON se distribuye libremente en su página Web bajo la Academic Free License, de carácter Open Source, por lo cual el código fuente puede ser modificado sin ninguna restricción. MASON es desarrollado por la universidad George Mason.

Algunas de las características importantes son las siguientes:

- 100% Java
- Portable y sencillo
- La arquitectura separa el modelo de la visualización.
- Visualización 2D y 3D

Entre las desventajas cabe mencionar que al ser código Java, el software no es tan eficiente como puede ser un software codificado en C++ por ejemplo. Para simulaciones donde existen varios threads o hilos de control que se ejecutan paralelamente, el rendimiento es algo muy importante, sin embargo MASON es una de las mejores librerías Open Source que existen actualmente para simulación basada en agentes.

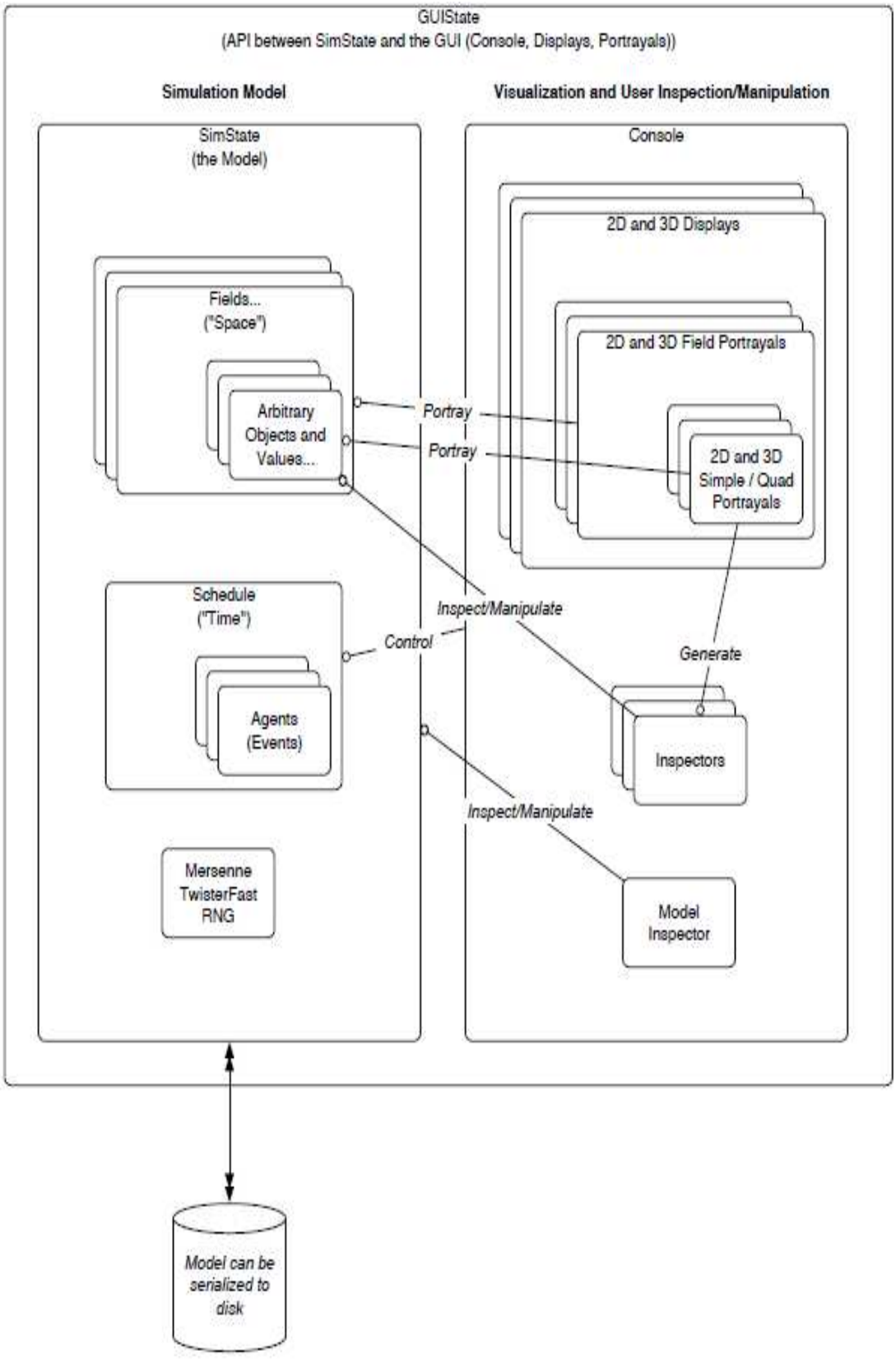


Fig. 2.2 Arquitectura genérica de MASON

2.2.3 MODELAMIENTO

2.2.3.1 Diagramas de casos de uso.

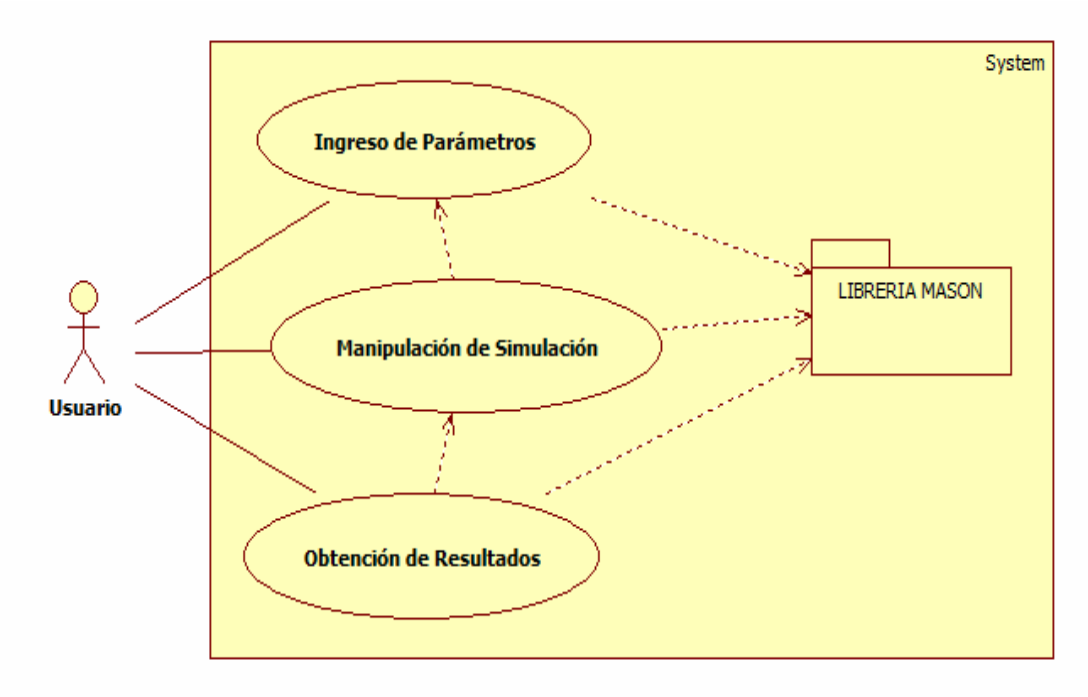


Diagrama 2.1. Diagrama de caso de uso de alto nivel: El Sistema.

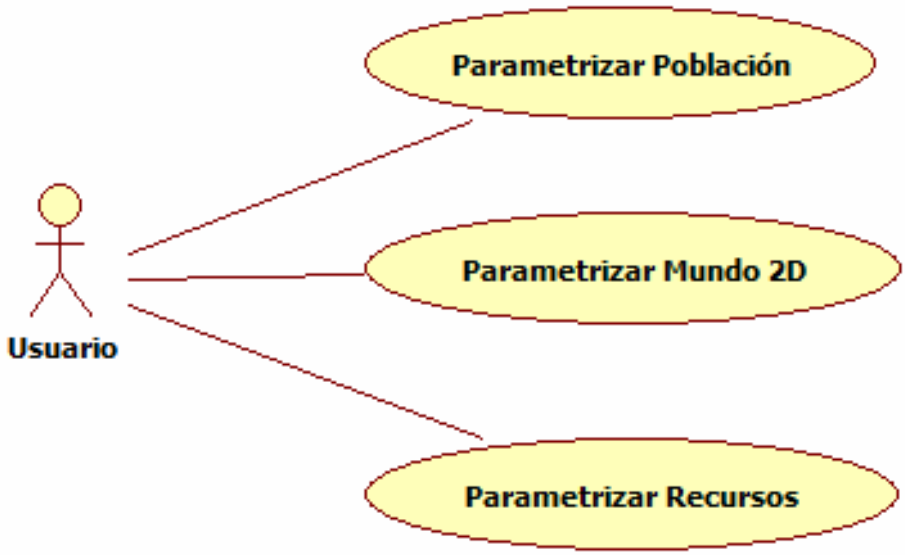


Diagrama 2.2. Diagrama de casos de uso elaborado para el caso de uso: Ingreso de parámetros.

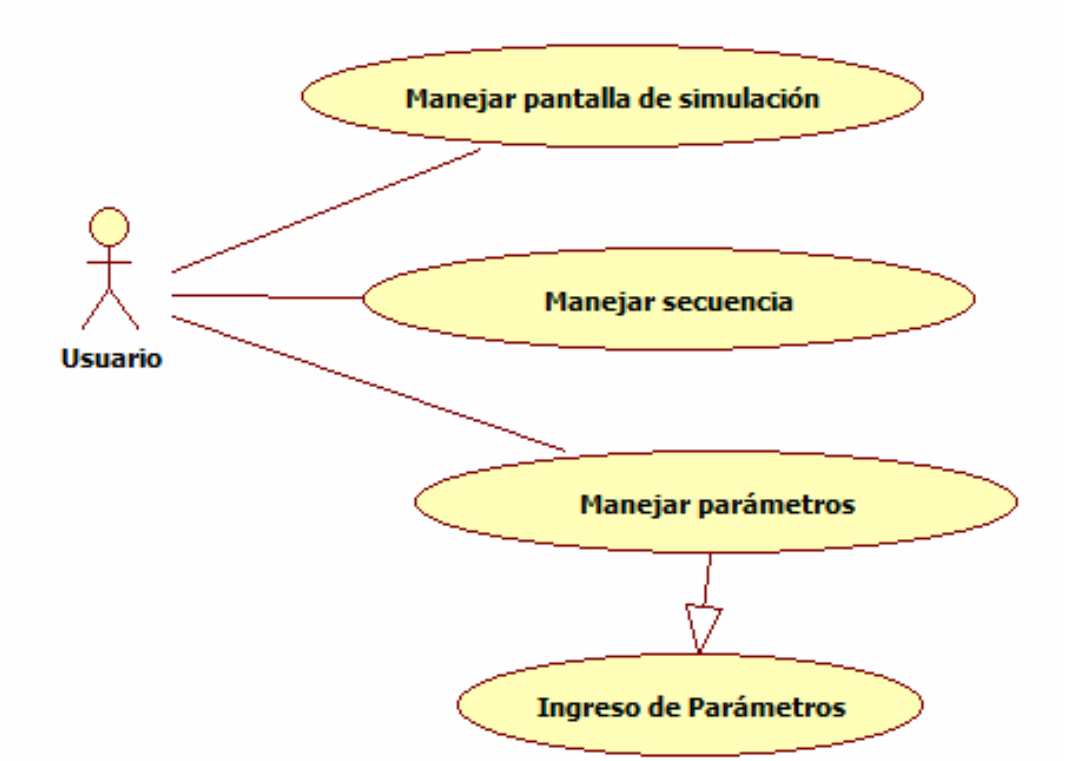


Diagrama 2.3. Diagrama de caso de uso elaborado para el caso de uso: Manipulación de Simulación.

Diagrama 2.4. Diagrama de caso de uso elaborado para el caso de uso: Obtención de resultados.

2.2.3.2 Especificación de casos de uso

Caso de uso: **INGRESO DE PARÁMETROS**
Descripción: Este caso de uso especifica el proceso de ingreso de parámetros por parte del usuario, esta información será necesaria antes de iniciar la simulación.
Precondiciones: • El usuario debe seleccionar una de las opciones disponibles. • El sistema puede estar en estado de simulación o no
Post condiciones: • El sistema podrá iniciar una nueva simulación • Se alterara el rumbo de la simulación actual
Flujo principal: **1.** El usuario accede por la interfaz de usuario hacia los parámetros de simulación **2.** El usuario Ingresa los valores correspondientes **3.** El usuario acepta los valores
Flujo alternativo: • En 3. El usuario cancela la opción
Flujo excepción: • En 2 No todas las opciones podrían estar disponibles

Caso de uso:
MANIPULACIÓN DE SIMULACIÓN

Descripción:
Este caso de uso especifica el manejo mismo de la simulación que el usuario puede realizar mientras se ejecuta la simulación.

Precondiciones:

- Se ha iniciado una simulación nueva
- Se ha cargado una simulación anterior

Post condiciones:

- Se finaliza la simulación en curso
- Continua la simulación actual

Flujo principal:

1. El usuario inicia la simulación
2. El usuario interactúa con la simulación
3. El usuario finaliza la simulación

Flujo alternativo: ninguno

Flujo excepción: ninguno

Caso de uso:
OBTENCIÓN DE RESULTADOS

Descripción:
El caso de uso especifica la obtención de información y resultados de la simulación.

Precondiciones:

- Se ha iniciado una simulación

Post condiciones:

- Se puede iniciar otra simulación
- Se puede continuar la simulación actual
- Se puede finalizar la simulación actual

Flujo principal:

1. El usuario puede visualizar la información en la interfaz de usuario
2. El usuario puede solicitar mayor detalle sobre información..
3. El usuario ajusta y manipula dicha información.
4. El usuario obtiene la información requerida

Flujo alternativo:

- En 2. El usuario no necesariamente va a solicitar mayor detalle de información.

Flujo excepción:

- En 2 No todas las opciones podrían estar disponibles

Caso de uso:
PARAMETRIZAR POBLACIÓN

Descripción:

Este caso de uso indica las características y opciones que el usuario podrá manejar para establecer una población inicial para empezar la simulación.

Precondiciones:

- El usuario debe acceder a los parámetros de simulación.
- El sistema puede estar en estado de simulación o no.

Post condiciones:

- Se puede iniciar la simulación con los parámetros establecidos.

Flujo principal:

1. El usuario accede a las opciones de población
2. El usuario configura el valor inicial de población.
3. El usuario configura el valor máximo de población.
4. El usuario configura el valor de vida máxima.
5. El usuario configura el valor de salud máxima.
6. El usuario configura el valor de distribución sexual de la población.
7. El usuario configura el valor de fertilidad máxima.
8. El usuario configura el valor de aptitud reproductiva máxima.
9. El usuario establece el valor de aptitud de supervivencia máxima.
10. El usuario establece el valor de mutación.

Flujo alternativo:

- El usuario puede dejar los valores por defecto de los parámetros y puede configurar solo los parámetros que el considere oportuno.

Flujo excepción:

- En 2. La población inicial solamente será un parámetro que se puede ingresar antes de iniciar una simulación.

Caso de uso: **PARAMETRIZAR MUNDO 2D**
Descripción: Este caso de uso indica las características y opciones que el usuario podrá manejar para establecer y generar el plano 2D donde se realizará la simulación.
Precondiciones: • El usuario debe acceder a los parámetros de simulación. • El sistema no debe estar en estado de simulación.
Post condiciones: • Se puede iniciar la simulación con los parámetros establecidos.
Flujo principal: 1. El usuario configura el ancho y el largo del mapa.
Flujo alternativo: • El usuario puede dejar los valores por defecto de los parámetros.
Flujo excepción: ninguno.

Caso de Uso: **PARAMETRIZAR RECURSOS**

Descripción:
Este caso de uso indica las características y opciones que el usuario podrá manejar para establecer los recursos que permiten sobrevivir a la población.

Precondiciones:

- El usuario debe acceder a los parámetros de simulación.
- El sistema puede estar en estado de simulación o no.

Post condiciones:

- Se puede iniciar la simulación con los parámetros establecidos.

Flujo principal:
1. El usuario configura el valor de recurso máximo.
3. El usuario configura el valor de regeneración de los recursos.
4. El usuario configura la distribución inicial de los recursos.
5. El usuario configura el valor de la frecuencia de regeneración..

Flujo alternativo:

- El usuario puede dejar los valores por defecto de los parámetros y configurar solamente lo que considere necesario.

Flujo excepción: NINGUNO

Caso de Uso:
MANEJAR PANTALLA DE SIMULACIÓN

Descripción:
Este caso de uso indica las características y opciones que el usuario podrá

manejar para visualizar y manipular la representación gráfica de la simulación con agentes.
Precondiciones: • La simulación puede iniciarse o no.
Post condiciones: • Se puede iniciar, pausar, detener o manipular la simulación. • El usuario puede observar los resultados.
Flujo principal: 1. El usuario configura opciones del modo gráfico. 2. El usuario puede aumentar o disminuir la visión de la simulación (zoom) 3. El usuario puede seleccionar las capas que visualizará en la pantalla. 4. El usuario configura la rapidez de actualización de la pantalla de simulación.
Flujo alternativo: • No es necesario el orden del flujo.
Flujo excepción: En 2. La simulación debe estar pausada o detenida solo para este aspecto (por cuestiones de implementación).

Caso de Uso: **MANEJAR SECUENCIA**
Descripción: Este caso de uso indica las características y opciones que el usuario podrá manejar para manipular el proceso secuencial de la simulación.

Precondiciones:

- Establecer parámetros de inicio de simulación.

Post condiciones:

- El usuario puede iniciar otra simulación.
- El usuario puede ver los resultados.

Flujo principal:

1. El usuario puede iniciar la simulación.
2. El usuario puede pausar la simulación.
3. El usuario puede detener la simulación.
4. El usuario puede reanudar la simulación.
5. El usuario puede guardar el estado actual de la simulación.
6. El usuario puede cargar un estado de simulación anterior.
7. El usuario puede seguir la simulación paso a paso.

Flujo alternativo:

- No es necesario el orden del flujo principal porque el usuario puede hacer lo que el necesite.

Flujo excepción: ninguno.

Caso de Uso:
MANEJAR PARÁMETROS

Descripción:

Este caso de uso indica las características y opciones que el usuario podrá manejar al respecto de la población y los recursos, en tiempo de ejecución de la simulación.

Precondiciones: • La simulación debe ser iniciada.
Post condiciones: • La simulación cambiará de acuerdo a los cambios establecidos en tiempo de ejecución de la simulación.
Flujo principal: 1. El usuario establece los valores de las características establecidas en PARAMETRIZAR RECURSOS Y PARAMETRIZAR POBLACIÓN, en tiempo de ejecución.
Flujo alternativo: ninguno
Flujo excepción: ninguno

Caso de Uso: **MANEJAR GRÁFICAS**
Descripción: Este caso de uso indica la capacidad que tendrá el usuario de generar gráficas (y manipularlas) con la información resultante de la simulación.
Precondiciones: • La simulación debe ser iniciada.

Post condiciones:

- La simulación continuará.
- La simulación puede ser manipulada.
- La gráfica se actualizará durante la simulación.

Flujo principal:

1. El usuario selecciona el parámetro que desea analizar mediante una gráfica.
2. El usuario genera la gráfica.
3. El usuario puede configurar el aspecto visual de la gráfica.
4. El usuario puede guardar la gráfica generada.

Flujo alternativo:

- El usuario puede crear nuevas gráficas para otros parámetros o inclusive generar una misma gráfica para dos o más parámetros.

Flujo excepción: ninguno

Caso de Uso:
INSPECCIONAR INFORMACIÓN

Descripción:

Este caso de uso indica las características y opciones que el usuario podrá manejar para inspeccionar información individual de los agentes en un momento dado.

Precondiciones: • La simulación debe ser iniciada.
Post condiciones: • La simulación continuará. • La simulación puede ser manipulada.
Flujo principal: 1. El usuario selecciona un agente en el entorno 3D. 2. El usuario accede a las opciones de inspección. 3. El usuario visualiza parámetros del agente en tiempo real en la interfaz de usuario. 4. El usuario cancela la inspección.
Flujo alternativo: ninguno
Flujo excepción: ninguno

2.2.3.3 Diagramas de clases

A continuación analizaremos las clases que intervienen en el sistema, en el diagrama 2.5 podemos analizar la estructura general del sistema, las clases que están anotadas son propias de la librería MASON, aquellas que no están anotadas son definidas específicamente para este proyecto, para un mayor detalle de la arquitectura de la librería MASON, se puede consultar el ANEXO.

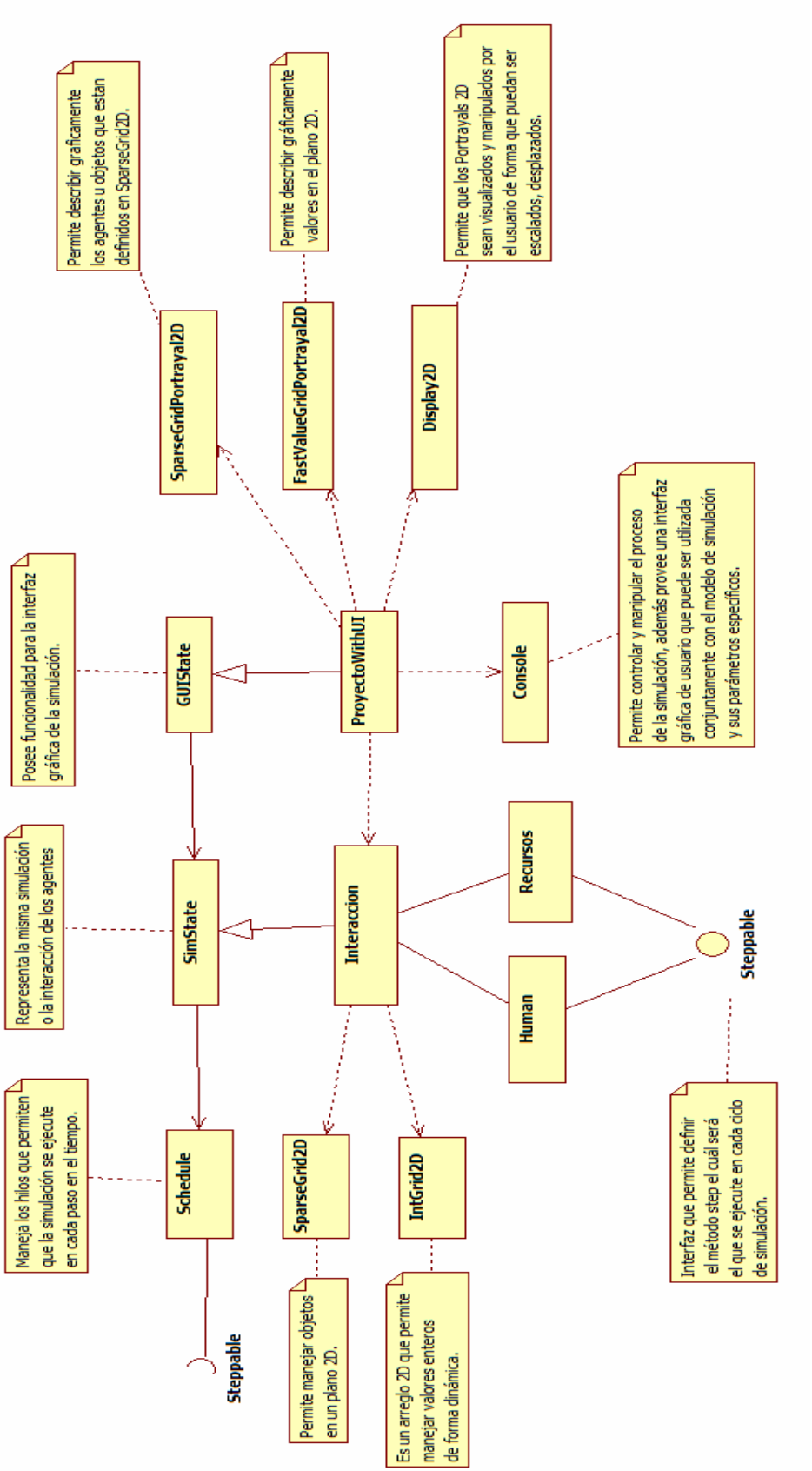

Permite describir graficamente los agentes u objetos que estan definidos en SparseGrid2D.
Permite describir gráficamente valores en el plano 2D.
Permite que los Portrayals 2D sean visualizados y manipulados por el usuario de forma que puedan ser escalados, desplazados.
SparseGridPortrayal2D
FastValueGridPortrayal2D
Display2D
Posee funcionalidad para la interfaz gráfica de la simulación.
GUIState
ProyectoWithUI
Console
Permite controlar y manipular el proceso de la simulación, además provee una interfaz gráfica de usuario que puede ser utilizada conjuntamente con el modelo de simulación y sus parámetros específicos.
Representa la misma simulación o la interacción de los agentes
SimState
Interaccion
Recursos
Human
Steppable
Maneja los hilos que permiten que la simulación se ejecute en cada paso en el tiempo.
Schedule
SparseGrid2D
IntGrid2D
Steppable
Permite manejar objetos en un plano 2D.
Es un arreglo 2D que permite manejar valores enteros de forma dinámica.
Interfaz que permite definir el método step el cuál será el que se ejecute en cada ciclo de simulación.

Diagrama 2.5. Diagrama de clases del software – vista general de la estructura del software.

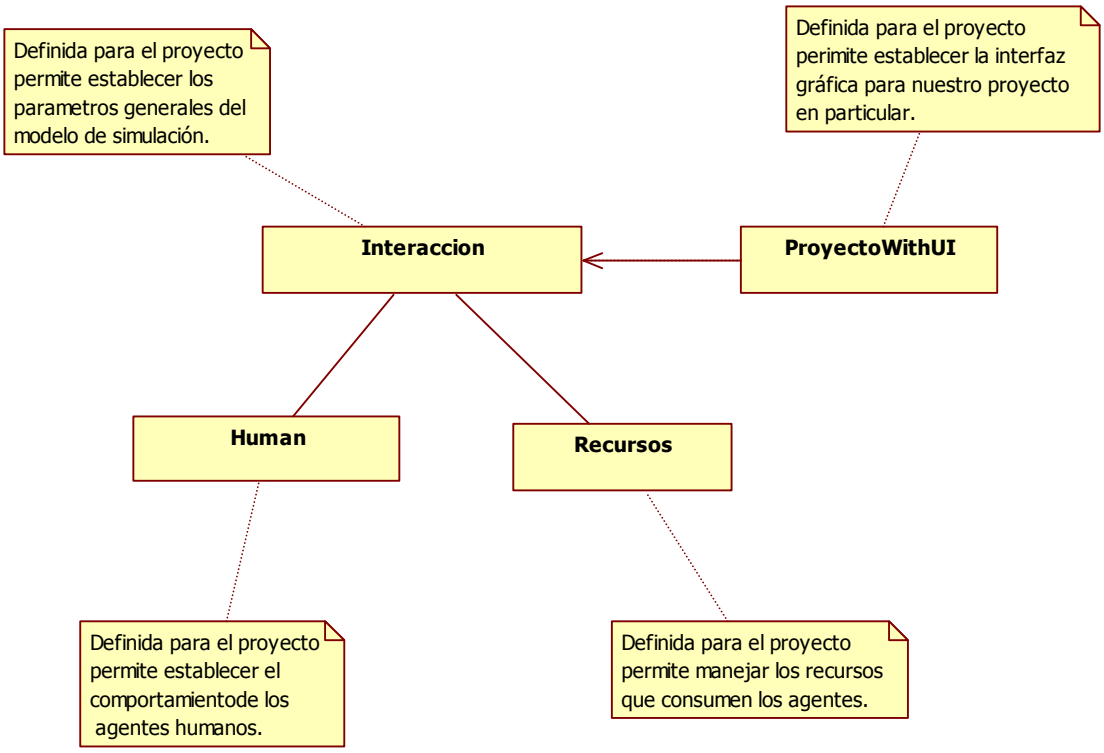


Diagrama 2.6. Diagrama de las clases definidas para el proyecto y sus responsabilidades.

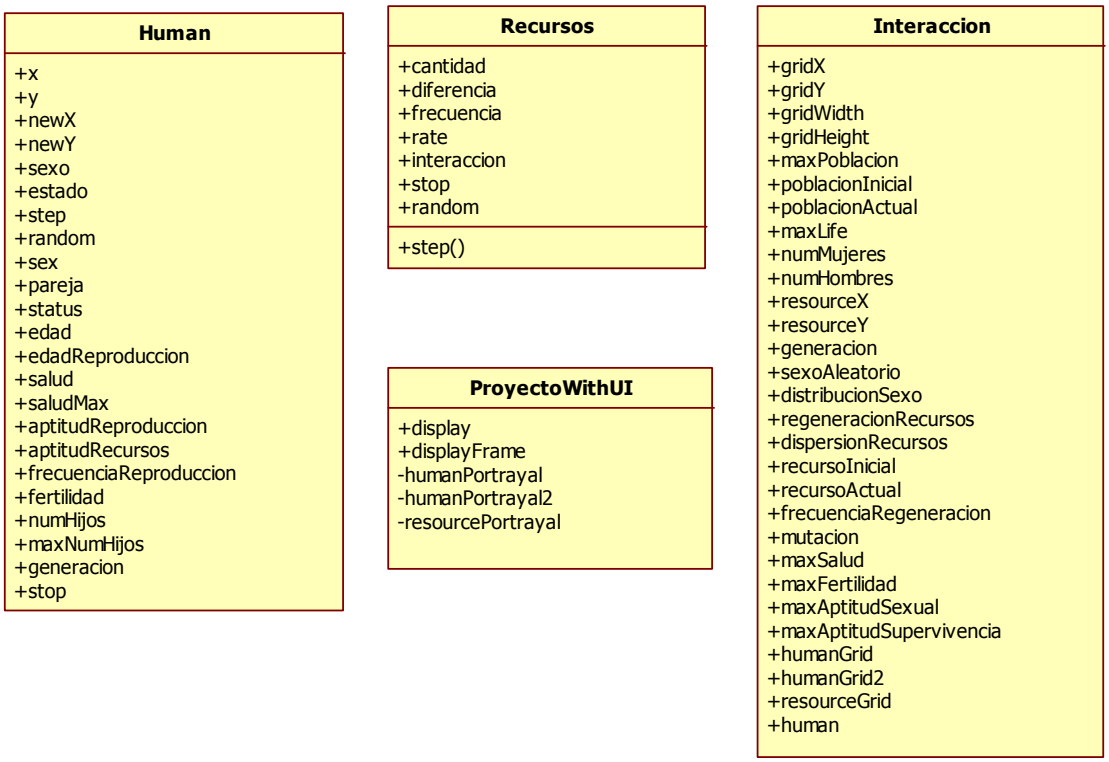


Diagrama 2.7 Clases definidas para el proyecto y sus atributos.

Diagrama 2.8. Clases definidas para el proyecto y sus operaciones.

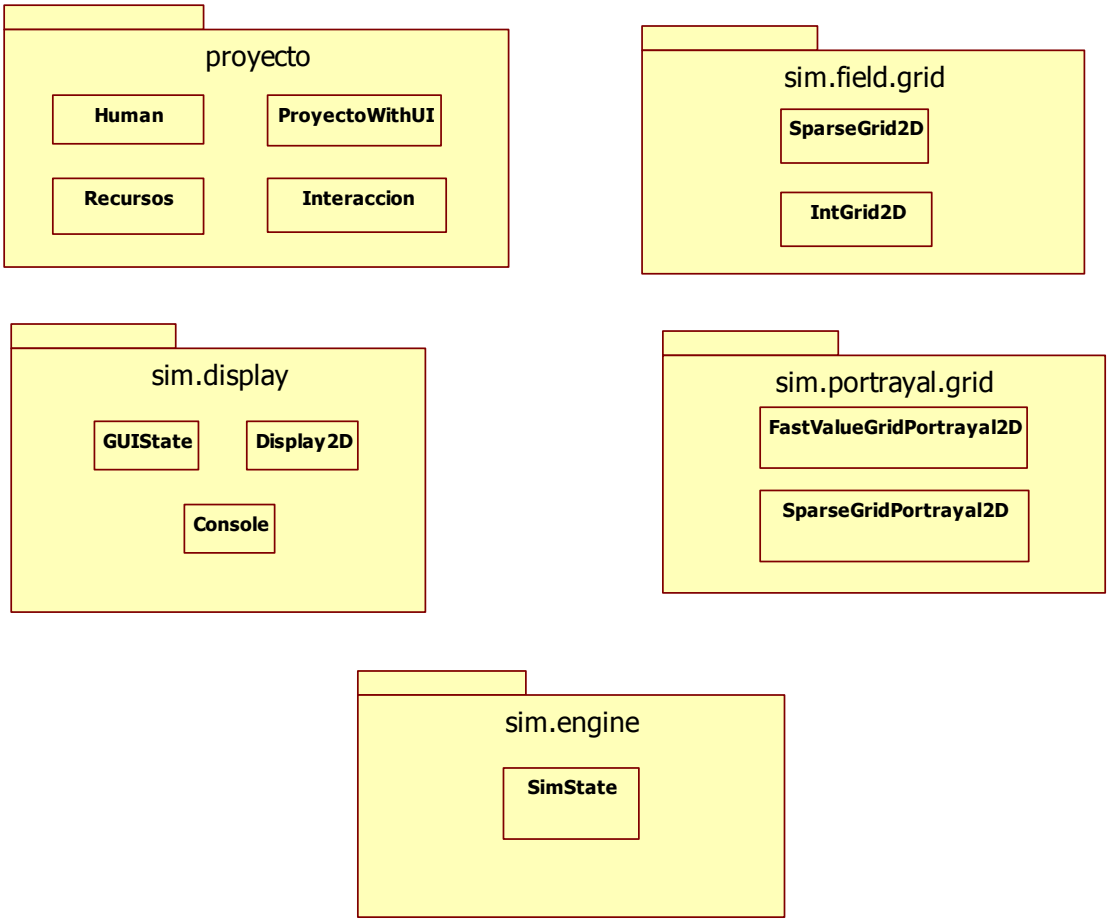


Diagrama 2.9 Paquetes utilizados y clases relacionadas.

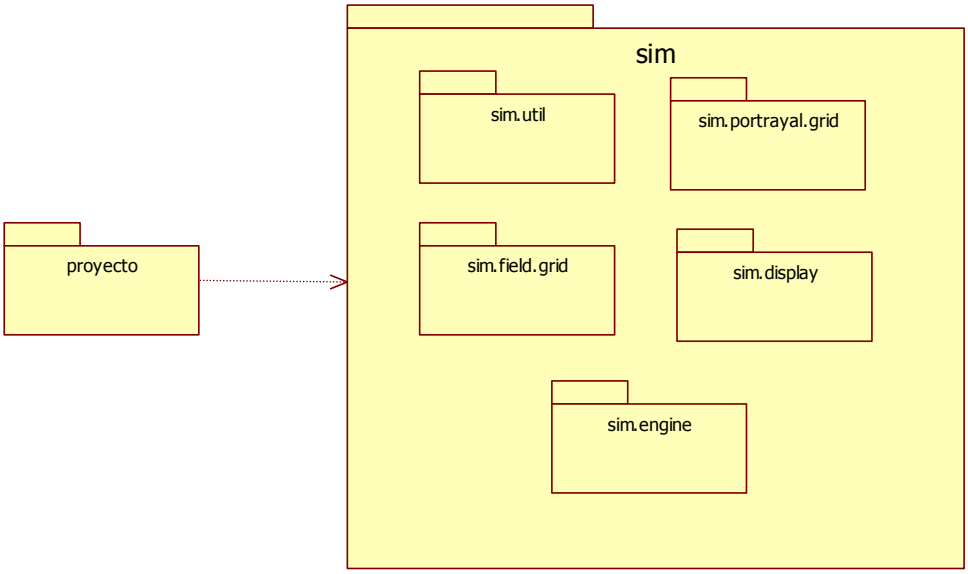


Diagrama 2.10 Dependencia de paquetes en la aplicación.

2.2.3.4 Descripción de clases definidas para el proyecto

Nuevamente debemos recalcar que la librería MASON posee un sinnúmero de clases que pueden ser reutilizadas para aplicaciones particulares, en nuestro proyecto hemos utilizado directamente las clases que se muestran en la tabla 2.4 (las cuales pueden ser analizadas con detalle en los ANEXOS, puesto que se trata de clases propias de MASON)

CLASE	RESPONSABILIDAD	PAQUETE
SimState	Representa la misma simulación o la interacción de los agentes.	sim.engine
GUIState	Posee funcionalidad para la interfaz gráfica de la simulación.	sim.display
SparseGrid2D	Permite manejar objetos en un plano 2D.	sim.field.grid
IntGrid2D	Es un arreglo 2D que permite manejar valores enteros de forma dinámica y con varias funcionalidades.	sim.field.grid
SparseGridPortrayal2D	Permite describir gráficamente los agentes u objetos que están definidos en SparseGrid2D.	sim.portrayal.grid
FastValueGridPortrayal2D	Permite describir gráficamente valores definidos en IntGrid2D.	sim.portrayal.grid
Display2D	Permite que los Portrayals 2D sean visualizados y manipulados por el usuario de forma que puedan ser escalados, desplazados.	sim.display
Console	Permite controlar y manipular el proceso de la simulación, además provee una interfaz gráfica de usuario que puede ser	sim.display

	utilizada conjuntamente con el modelo de simulación y sus parámetros específicos.	

Tabla 2.4. Clases de la librería MASON utilizadas directamente con nuestro proyecto.

Para nuestro proyecto en particular hemos definido 4 clases que se adaptan perfectamente a la estructura definida por las librerías MASON, y permite enfocarse en el modelo de simulación, dando transparencia a otros aspectos de la implementación, las clases que se han definido son:

- Human
- Recursos
- Interaccion
- ProyectoWithUI

NOMBRE DE LA CLASE: Human
DESCRIPCION: Definida específicamente para el proyecto permite establecer el comportamiento de los agentes humanos en la simulación.
IMPLEMENTA: Esta clase implementa la interfaz Steppable.
PAQUETE: Proyecto
ATRIBUTOS:

Identificador: x
Tipo: INTEGER
Descripción: Permite establecer la posición actual del agente en el eje x del plano 2D.

Visibilidad: PUBLICA

Identificador: y
Tipo: INTEGER
Descripción: Permite establecer la posición actual del agente en el eje y del plano 2D.
Visibilidad: PUBLICA

Identificador: newX
Tipo: INTEGER
Descripción: Permite establecer la nueva posición a la que se desplazará el agente en el eje X.
Visibilidad: PUBLICA

Identificador: newY
Tipo: INTEGER
Descripción: Permite establecer la nueva posición a la que se desplazará el agente en el eje Y.
Visibilidad: PUBLICA

Identificador: random
Tipo: MersenneTwisterFast
Descripción: Semilla para generar números aleatorios.
Visibilidad: PUBLICA

Identificador: sexo
Tipo: STRING
Descripción: Descripción nominal del género de un agente.
Visibilidad: PUBLICA

Identificador: estado
Tipo: STRING

Descripción: Descripción nominal del estado de un agente.
Visibilidad: PUBLICA

Identificador: step
Tipo: LONG
Descripción: Indica el paso en que se encuentra la simulación.
Visibilidad: PUBLICA

Identificador: sex
Tipo: INTEGER
Descripción: Sexo del agente.
Visibilidad: PUBLICA

Identificador: pareja
Tipo: BOOLEAN
Descripción: Indica si el agente se encuentra con alguna pareja.
Visibilidad: PUBLICA

Identificador: status
Tipo: INTEGER
Descripción: Estado actual del agente.
Visibilidad: PUBLICA

Identificador: edad
Tipo: INTEGER
Descripción: Edad del agente
Visibilidad: PUBLICA

Identificador: edadReproduccion
Tipo: INTEGER
Descripción: Edad en la que el agente puede empezar a reproducirse.
Visibilidad: PUBLICA

Identificador: saludMax
Tipo: INTEGER
Descripción: Salud máxima que puede tener el agente.
Visibilidad: PUBLICA

Identificador: salud
Tipo: INTEGER
Descripción: Salud del agente en un momento determinado.
Visibilidad: PUBLICA

Identificador: aptitudReproduccion
Tipo: INTEGER
Descripción: Distancia para buscar agentes para reproducirse.
Visibilidad: PUBLICA

Identificador: aptitudRecursos
Tipo: INTEGER
Descripción: Distancia para buscar recursos.
Visibilidad: PUBLICA

Identificador: frecuenciaReproduccion
Tipo: DOUBLE
Descripción: Rapidez con la que el agente puede reproducirse.
Visibilidad: PUBLICA

Identificador: fertilidad
Tipo: INTEGER
Descripción: Número de hijos que podría generar un agente.
Visibilidad: PUBLICA

Identificador: numHijos

Tipo: INTEGER
Descripción: Número de hijos que posee el agente.
Visibilidad: PUBLICA

Identificador: generación
Tipo: INTEGER
Descripción: Número de generación a la que pertenece el agente.
Visibilidad: PUBLICA

Identificador: stop
Tipo: STOPPABLE
Descripción: Permite detener el hilo de control para el agente.
Visibilidad: PUBLICA

OPERACIONES:

Nombre: Human()
Parámetros: Double distribucionSexo, int Edad
Descripción: Constructor con dos parámetros.
Retorna: NADA
Visibilidad: PUBLICA

Nombre: Human()
Parámetros: Double distribucionSexo, int Edad, boolean sex_Aleatorio
Descripción: Constructor con tres parámetros.
Retorna: NADA
Visibilidad: PUBLICA

Nombre: getEdad()
Parámetros:
Descripción: Retorna la edad del agente.
Retorna: INTEGER

Visibilidad: PUBLICA

Nombre: getSexo()
Parámetros:
Descripción: Retorna el sexo del agente
Retorna: STRING
Visibilidad: PUBLICA

Nombre: getEstado()
Parámetros:
Descripción: Retorna el estado del agente.
Retorna: STRING
Visibilidad: PUBLICA

Nombre: getSalud()
Parámetros:
Descripción: Retorna la salud del agente.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getSaludMax()
Parámetros:
Descripción: Retorna la salud máxima del agente.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getFertilidad()
Parámetros:
Descripción: Retorna el valor de fertilidad del agente.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getAptitudReproduccion()
Parámetros:
Descripción: Devuelve el valor de aptitudReproduccion del agente.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getFrecuenciaReproduccion()
Parámetros:
Descripción: Retorna el valor de frecuenciaReproduccion de un agente.
Retorna: DOUBLE
Visibilidad: PUBLICA

Nombre: getAptitudRecursos()
Parámetros:
Descripción: Retorna el valor del atributo AptitudRecursos que posea un agente.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getNumHijos()
Parámetros:
Descripción: Retorna el valor del atributo numHijos que tenga el agente.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getGeneracion()
Parámetros:
Descripción: Retorna la generación del agente.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: generarAptitud()

Parámetros: Human hijo, Human progenitor, Interaccion interaccion
Descripción: Genera los valores de las características de un nuevo agente humano en base a las características de los padres, y de la configuración de la mutación por parte del usuario.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: step()
Parámetros: SimState state
Descripción: Es la implementación del método step declarado en la interfaz Steppable, permite definir la conducta del agente que se realizará en cada ciclo de simulación.
Retorna: VOID
Visibilidad: PUBLICA

NOMBRE DE LA CLASE: Recursos

DESCRIPCIÓN: Esta clase permite que los recursos que se encuentran en el mundo 2D sean incluidos en los ciclos de simulación.

IMPLEMENTA: Esta clase implementa la interfaz Steppable.

PAQUETE: Proyecto

ATRIBUTOS:

Identificador: cantidad
Tipo: INTEGER

Descripción: Es la cantidad de recursos que han sido consumidos por los agentes en el mundo 2D.
Visibilidad: PUBLICA

Identificador: diferencia
Tipo: INTEGER
Descripción: Es la diferencia entre el recurso máximo y el recurso actual existente en el mundo 2D.
Visibilidad: PUBLICA

Identificador: Rate
Tipo: DOUBLE
Descripción: Es el valor de la regeneración de recursos.
Visibilidad: PUBLICA

Identificador: frecuencia
Tipo: DOUBLE
Descripción: Es el valor de la frecuencia de regeneración de los recursos.
Visibilidad: PUBLICA

OPERACIONES:

Nombre: step()
Parámetros: Simstate state
Descripción: Permite establecer el comportamiento que tendrán los recursos, conforme a los parámetros establecidos por el usuario.
Retorna: VOID
Visibilidad: PUBLICA

NOMBRE DE LA CLASE: ProyectoWithUI

DESCRIPCIÓN: Permite establecer la interfaz gráfica de usuario para nuestro proyecto en particular.

HERENCIA: Esta clase hereda atributos y métodos de GUIState.

PAQUETE: Proyecto

ATRIBUTOS:

Identificador: displayFrame
Tipo: JFrame
Descripción: Es el frame o ventana que permite visualizar los componentes en la pantalla del monitor.
Visibilidad: PUBLICA

Identificador: display
Tipo: Display2D
Descripción: Establece una interfaz gráfica para observar y manipular la simulación.
Visibilidad: PUBLICA

Identificador: humanPortrayal
Tipo: SparseGridPortrayal2D
Descripción: Permite describir gráficamente los agentes u objetos que están definidos en SparseGrid2D. Permitirá especificar una capa para visualizar los agentes.
Visibilidad: PRIVADO

Identificador: humanPortrayal2
Tipo: SparseGridPortrayal2D
Descripción: Permite describir gráficamente los agentes u objetos que están definidos en SparseGrid2D. Permitirá especificar una capa para visualizar los agentes.
Visibilidad: PRIVADO

Identificador: resourcePortrayal
Tipo: FastValueGridPortrayal2D
Descripción: Permite describir gráficamente los valores que están definidos en IntGrid2D. Permitirá especificar una capa para visualizar los recursos.
Visibilidad: PRIVADO

OPERACIONES:

Nombre: main()
Parámetros: String[ ] args
Descripción: Permite ejecutar la aplicación.
Retorna: VOID
Visibilidad: PUBLIC STATIC

Nombre: ProyectoWithUI
Parámetros: SimState state
Descripción: constructor con un parámetro de la clase.
Retorna: NADA
Visibilidad: PUBLICA

Nombre: getName()
Parámetros:

Descripción: Establece un nombre para la simulación.
Retorna: STRING
Visibilidad: PUBLICA

Nombre: init()
Parámetros: Controller controller.
Descripción: Inicializa los elementos que serán visibles en la interfaz gráfica.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: setupPortrayals()
Parámetros:
Descripción: Prepara e inicializa los Portrayals para ser utilizados.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getInspector()
Parámetros:
Descripción: Establece un Inspector para analizar la información de los agentes, y del modelo de simulación. Anula el método de la clase padre.
Retorna: Inspector
Visibilidad: PUBLICA

Nombre: start()
Parámetros:
Descripción: Inicializa todos los elementos necesarios para inicializar la aplicación. Anula el método de la clase padre.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: load()
Parámetros: SimState state

Descripción: Inicializa la simulación sin el modo gráfico. Anula el método de la clase padre.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: quit()
Parámetros:
Descripción: Cuando termina la aplicación se asegura de que todos los recursos utilizados sean devueltos al computador. Anula el método de la clase padre.
Retorna: VOID
Visibilidad: PUBLICA

NOMBRE DE LA CLASE: Interaccion
DESCRIPCION: permite establecer los parámetros generales del modelo de simulación, así como manejar los aspectos propios de la simulación.
HERENCIA: Esta clase hereda sus atributos y métodos de la clase SimState.
PAQUETE: Proyecto
ATRIBUTOS:

Identificador: gridX
Tipo: INTEGER
Descripción: Valor de la dimensión x del mundo 2D.
Visibilidad: PUBLICA

Identificador: gridY
Tipo: INTEGER
Descripción: Valor de la dimensión y del mundo 2D.
Visibilidad: PUBLICA

Identificador: gridWidth
Tipo: INTEGER
Descripción: Ancho del plano del mundo 2D.
Visibilidad: PUBLICA

Identificador: gridHeight
Tipo: INTEGER
Descripción: Alto del plano del mundo 2D.
Visibilidad: PUBLICA

Identificador: maxPoblacion
Tipo: INTEGER
Descripción: Valor máximo de la población de agentes.
Visibilidad: PUBLICA

Identificador: poblacionInicial
Tipo: INTEGER
Descripción: Número de habitantes inicial de agentes.
Visibilidad: PUBLICA

Identificador: poblacionActual
Tipo: INTEGER
Descripción: Número de agentes que habitan el mundo 2D en un momento determinado
Visibilidad: PUBLICA

Identificador: maxLife
Tipo: INTEGER
Descripción: Vida máxima que podrían tener los agentes.
Visibilidad: PUBLICA

Identificador: numMujeres
Tipo: INTEGER
Descripción: Número de habitantes mujeres en un momento dado.
Visibilidad: PUBLICA

Identificador: numHombres
Tipo: INTEGER
Descripción: Número de habitantes del género masculino en un momento dado.
Visibilidad: PUBLICA

Identificador: resourceX
Tipo: INTEGER
Descripción: Posición de un recurso en el eje X.
Visibilidad: PUBLICA

Identificador: resourceY
Tipo: INTEGER
Descripción: Posición de un recurso en el eje Y.
Visibilidad: PUBLICA

Identificador: generacion
Tipo: INTEGER
Descripción: generación de habitantes que se hallan en un momento en el mundo 2D.
Visibilidad: PUBLICA

Identificador: natalidad
Tipo: LONG
Descripción: Número de agentes nacidos en un ciclo.
Visibilidad: PUBLICA

Identificador: mortalidad
Tipo: LONG
Descripción: Número de muertos en un ciclo.
Visibilidad: PUBLICA

Identificador: sexoAleatorio
Tipo: BOOLEAN
Descripción: Bandera que indica si la distribución de sexo se hará de forma aleatoria en medio de la población.
Visibilidad: PUBLICA

Identificador: distribucionSexo
Tipo: DOUBLE
Descripción: Porcentaje que indica la distribución de género en la población.
Visibilidad: PUBLICA

Identificador: regeneracionRecursos
Tipo: DOUBLE
Descripción: Probabilidad de que los recursos se regeneren.
Visibilidad: PUBLICA

Identificador: dispersionRecursos
Tipo: DOUBLE
Descripción: Rango de concentración o dispersión de los recursos en el mundo 2D.
Visibilidad: PUBLICA

Identificador: recursoInicial
Tipo: DOUBLE
Descripción: Es el valor inicial y máximo de recursos.
Visibilidad: PUBLICA

Identificador: recursoActual
Tipo: DOUBLE
Descripción: Valor de los recursos en un momento determinado.
Visibilidad: PUBLICA

Identificador: frecuenciaRegeneracion
Tipo: DOUBLE
Descripción: Rapidez de regeneración de recursos o ciclos necesarios para proceder a regenerar los recursos.
Visibilidad: PUBLICA

Identificador: mutacion
Tipo: DOUBLE
Descripción: Probabilidad de mutación en los caracteres de la población.
Visibilidad: PUBLICA

Identificador: maxSalud
Tipo: INTEGER
Descripción: Salud máxima que podrían tener los agentes.
Visibilidad: PUBLICA

Identificador: maxFertilidad
Tipo: INTEGER
Descripción: Fertilidad máxima que podrían tener los agentes.
Visibilidad: PUBLICA

Identificador: maxAptitudSexual
Tipo: INTEGER
Descripción: Valor máximo que podría poseer un agente en su parámetro aptitudReproduccion.
Visibilidad: PUBLICA

Identificador: maxAptitudSupervivencia
Tipo: INTEGER
Descripción: Valor máximo que podría adquirir la característica aptitudRecursos de los agentes humanos.
Visibilidad: PUBLICA

Identificador: humanGrid
Tipo: SparseGrid2D
Descripción: Arreglo 2D con varias funcionalidades que sirve para manejar los objetos o agentes. En este arreglo 2D manejaremos los agentes de género masculino.
Visibilidad: PUBLICA

Identificador: humanGrid2
Tipo: SparseGrid2D
Descripción: Arreglo 2D con varias funcionalidades que sirve para manejar los objetos o agentes. En este arreglo 2D manejaremos los agentes de género femenino.
Visibilidad: PUBLICA

Identificador: resourceGrid
Tipo: IntGrid2D
Descripción: Arreglo 2D con varias funcionalidades que sirve para manejar valores numéricos. En este arreglo 2D manejaremos los valores de los recursos.
Visibilidad: PUBLICA

OPERACIONES:

Nombre: Interaccion()
Parámetros:
Descripción: Constructor por defecto.
Retorna: NADA
Visibilidad: PUBLICA

Nombre: Interaccion()
Parámetros: long aleatorio, int ancho, int alto, int poblacion
Descripción: Constructor por defecto con tres parámetros, inicializa la semilla aleatoria, el alto y ancho del mundo 2D y la población inicial.
Retorna: NADA
Visibilidad: PUBLICA

Nombre: getPoblacionActual()
Parámetros:
Descripción: Retorna el valor de la población actual del sistema.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: generarRecursos()
Parámetros: int cantidad, double regeneracion
Descripción: permite regenerar los recursos consumidos en base a los parámetros, que son la cantidad de recursos a regenerar, y la probabilidad de regeneración.
Retorna: NADA
Visibilidad: PUBLICA

Nombre: start()
Parámetros:
Descripción: Inicializa los parámetros definidos por el usuario para empezar una nueva simulación.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getGeneracion()
Parámetros:
Descripción: Retorna el valor del atributo generación.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getRecursoActual()
Parámetros:
Descripción: Retorna el valor del atributo recursoActual.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getNatalidad()
Parámetros:
Descripción: Retorna el valor del parámetro natalidad.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getMortalidad()
Parámetros:
Descripción: Retorna el valor del parámetro mortalidad.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getNumHombres()
Parámetros:
Descripción: Retorna el valor del parámetro numHombres.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getNumMujeres()
Parámetros:
Descripción: Retorna el valor del parámetro numMujeres.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getGridX()
Parámetros:
Descripción: Retorna el valor del parámetro gridX, que es el ancho del plano 2D.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: getGridY()
Parámetros:
Descripción: Retorna el valor del parámetro gridY, que es el alto del plano 2D.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setGridX()
Parámetros: int val
Restricción: se podrán asignar valores enteros entre 100 y 200
Descripción: Asigna un valor al parámetro gridX.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: setGridY()
Parámetros: int val
Restricción: se podrán asignar valores enteros entre 100 y 200
Descripción: Asigna un valor al parámetro gridY.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getPoblacionInicial()
Parámetros:
Descripción: Retorna el valor del parámetro poblacionInicial para que pueda ser observado en la interfaz de usuario.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setPoblacionInicial()
Parámetros: int val
Descripción: Asigna un valor al parámetro poblacionInicial para que el usuario pueda establecer el valor desde la interfaz gráfica de usuario.
Restricción: se podrán asignar valores enteros entre 1 y 15000
Retorna: VOID
Visibilidad: PUBLICA

Nombre: setMaxPoblacion()
Parámetros: int val
Descripción: Asigna un valor al parámetro maxPoblacion.
Restricción: se podrán asignar valores enteros entre 1 y 30000
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getMaxPoblacion()
Parámetros:
Descripción: Retorna el valor del parámetro maxPoblacion.

Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setMaxLife()
Parámetros: int val
Descripción: Asigna un valor al parámetro maxLife.
Restricción: se podrán asignar valores enteros entre 1 y 500
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getMaxLife()
Parámetros:
Descripción: Retorna el valor del parámetro maxLife.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setMaxSalud()
Parámetros: int val
Descripción: Asigna un valor al parámetro maxSalud.
Restricción: se podrán asignar valores enteros entre 1 y 500
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getMaxSalud()
Parámetros:
Descripción: Retorna el valor del parámetro maxSalud.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setMaxFertilidad()
Parámetros: int val
Descripción: Asigna un valor al parámetro maxFertilidad.

Restricción: se podrán asignar valores enteros entre 1 y 8.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getMaxFertilidad()
Parámetros:
Descripción: Retorna el valor del parámetro maxFertilidad.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setMaxAptitudSexual()
Parámetros: int val
Descripción: Asigna un valor al parámetro maxAptitudSexual.
Restricción: se podrán asignar valores enteros entre 1 y 100
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getMaxAptitudSexual()
Parámetros:
Descripción: Retorna el valor del parámetro maxAptitudSexual.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setMaxAptitudSupervivencia()
Parámetros: int val
Descripción: Asigna un valor al parámetro maxAptitudSupervivencia.
Restricción: se podrán asignar valores enteros entre 1 y 100
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getMaxAptitudSupervivencia()
Parámetros:

Descripción: Retorna el valor del parámetro maxAptitudSupervivencia.
Retorna: INTEGER
Visibilidad: PUBLICA

Nombre: setMutacion()
Parámetros: double val
Descripción: Asigna un valor al parámetro mutacion.
Restricción: se podrán asignar valores reales entre 0.0 y 1.0
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getMutacion()
Parámetros:
Descripción: Retorna el valor del parámetro mutacion.
Retorna: DOUBLE
Visibilidad: PUBLICA

Nombre: domMutacion()
Parámetros:
Descripción: Retorna un objeto de tipo Interval, es decir retorna un intervalo, que será visible en la interfaz gráfica de usuario como una barra de desplazamiento.
Restricción: El intervalo debe tener relación con los valores que se ha restringido en la función setMutacion(), por lo tanto el intervalo estará ente 0.0 y 1.0
Retorna: OBJECT
Visibilidad: PUBLICA

Nombre: setDistribucionSexo_Aleatoria()
Parámetros: boolean val
Descripción: Asigna un valor al parámetro sexoAleatorio.
Retorna: VOID

Visibilidad: PUBLICA

Nombre: getDistribucionSexo_Aleatoria()
Parámetros:
Descripción: Retorna el valor del parámetro sexoAleatorio.
Retorna: BOOLEAN
Visibilidad: PUBLICA

Nombre: setDistribucionSexo_Aleatoria()
Parámetros: boolean val
Descripción: Asigna un valor al parámetro sexoAleatorio.
Retorna: VOID
Visibilidad: PUBLICA

Nombre: setDistribucionSexo()
Parámetros: double val
Descripción: Asigna un valor al parámetro distribucionSexo.
Restricción: se podrán asignar valores reales entre 0.1 y 0.9
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getDistribucionSexo()
Parámetros:
Descripción: Retorna el valor del parámetro distribucionSexo .
Retorna: DOUBLE
Visibilidad: PUBLICA

Nombre: domDistribucionSexo()
Parámetros:
Descripción: Retorna un intervalo.
Restricción: El intervalo estará entre 0.0 y 1.0
Retorna: OBJECT

Visibilidad: PUBLICA

Nombre: setRecursoMax()
Parámetros: double val
Descripción: Asigna un valor al parámetro recursoInicial.
Restricción: se podrán asignar valores reales entre 0.0 y 10000.0
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getRecursoMax()
Parámetros:
Descripción: Retorna el valor del parámetro recursoInicial .
Retorna: DOUBLE
Visibilidad: PUBLICA

Nombre: domRecursoMax()
Parámetros:
Descripción: Retorna un intervalo.
Restricción: El intervalo devuelto debe estar entre 0.0 y 10000.0
Retorna: OBJECT
Visibilidad: PUBLICA

Nombre: setDispersionRecurso()
Parámetros: double val
Descripción: Asigna un valor al parámetro dispersionRecursos.
Restricción: se podrán asignar valores reales entre 0.1 y 1.0
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getDispersionRecursos()
Parámetros:
Descripción: Retorna el valor del parámetro dispersionRecursos

Retorna: DOUBLE
Visibilidad: PUBLICA

Nombre: domDispersionRecursos()
Parámetros:
Descripción: Retorna un intervalo.
Restricción: El intervalo debe hallarse entre 0.1 y 1.0
Retorna: OBJECT
Visibilidad: PUBLICA

Nombre: setFrecuenciaRegeneracion()
Parámetros: double val
Descripción: Asigna un valor al parámetro frecuenciaRegeneracion.
Restricción: se podrán asignar valores entre 1 y 200
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getFrecuenciaRegeneracion()
Parámetros:
Descripción: Retorna el valor del parámetro frecuenciaRegeneracion.
Retorna: DOUBLE
Visibilidad: PUBLICA

Nombre: domFrecuenciaRegeneracion()
Parámetros:
Descripción: Retorna un intervalo.
Restricción: El intervalo debe encontrarse entre 1 y 200
Retorna: OBJECT
Visibilidad: PUBLICA

Nombre: setRegeneracionRecursos()
Parámetros: double val

Descripción: Asigna un valor al parámetro regeneracionRecursos.
Restricción: se podrán asignar valores reales entre 0.0 y 1.0
Retorna: VOID
Visibilidad: PUBLICA

Nombre: getRegeneracionRecursos()
Parámetros:
Descripción: Retorna el valor del parámetro regeneracionRecursos .
Retorna: DOUBLE
Visibilidad: PUBLICA

Nombre: domRegeneracionRecursos()
Parámetros:
Descripción: Retorna un intervalo.
Restricción: El intervalo debe encontrarse entre 0.0 y 1.0
Retorna: OBJECT
Visibilidad: PUBLICA

2.2.3.5 Diagramas de secuencia

Los diagramas de secuencia nos muestran la parte dinámica del sistema, así como los diagramas de clase nos indican la parte estática; presentamos a continuación los diagramas de secuencia que relacionan las clases de nuestro proyecto conjuntamente con las clases de la librería MASON que hemos utilizado.

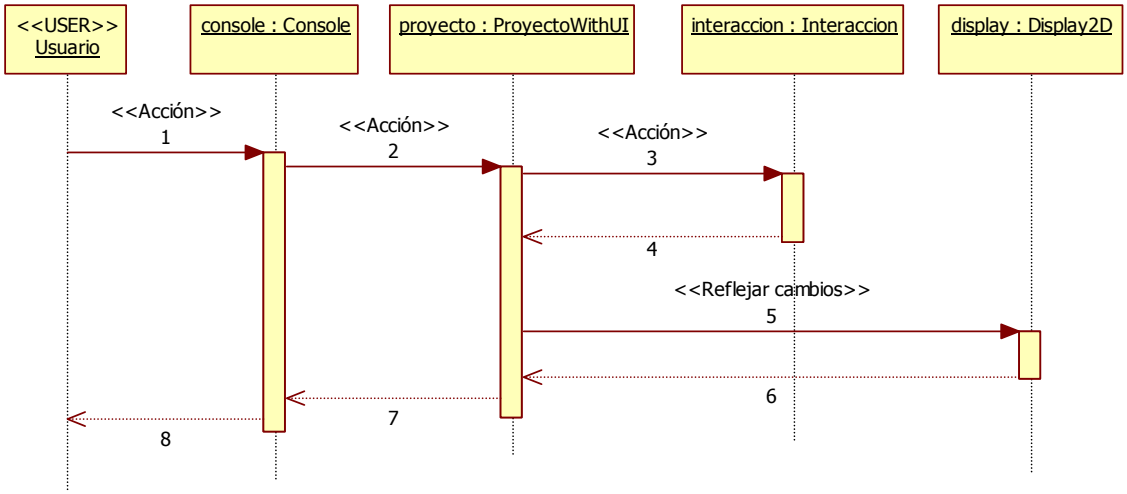


Diagrama 2.11. Diagrama de secuencia que indica como las acciones que el usuario establezca en los parámetros y en la simulación se verán reflejados en la interfaz gráfica de usuario y en el mismo estado de la simulación.

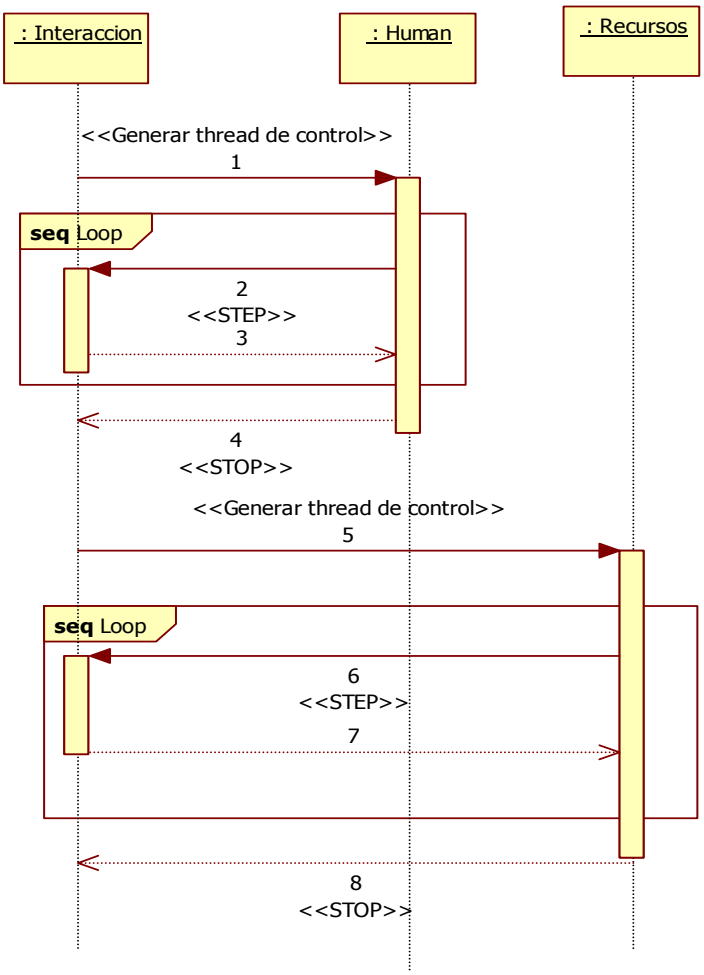


Diagrama 2.12. Generación de los hilos independientes de control.

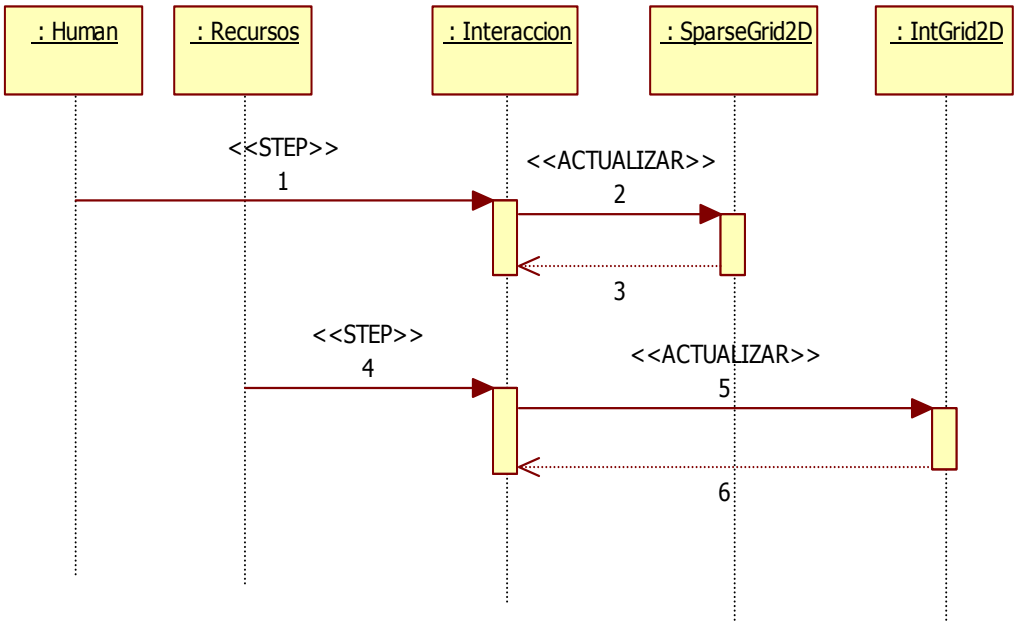


Diagrama 2.13. Actualización de cada ciclo de simulación sin modo gráfico.

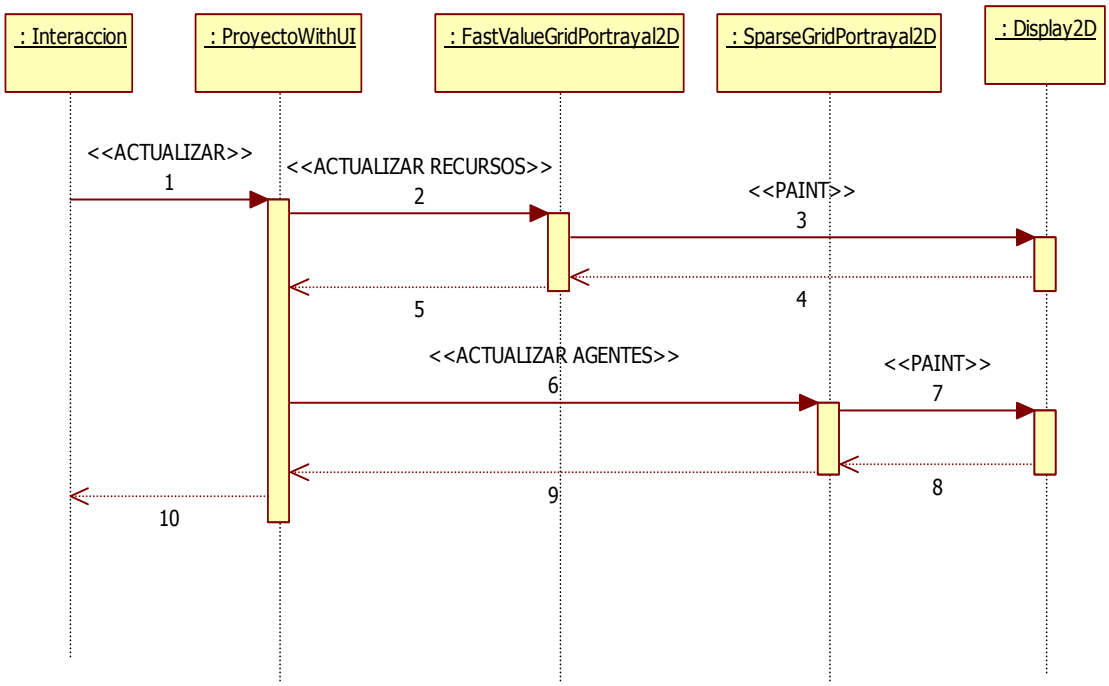


Diagrama 2.14. Diagrama de secuencia que indica cómo se actualiza el estado de simulación en la pantalla de simulación en la interfaz gráfica de usuario.

2.2.3.6 Diagramas de estado.

Los diagramas de estado nos indican los estados en que se puede encontrar nuestro sistema.

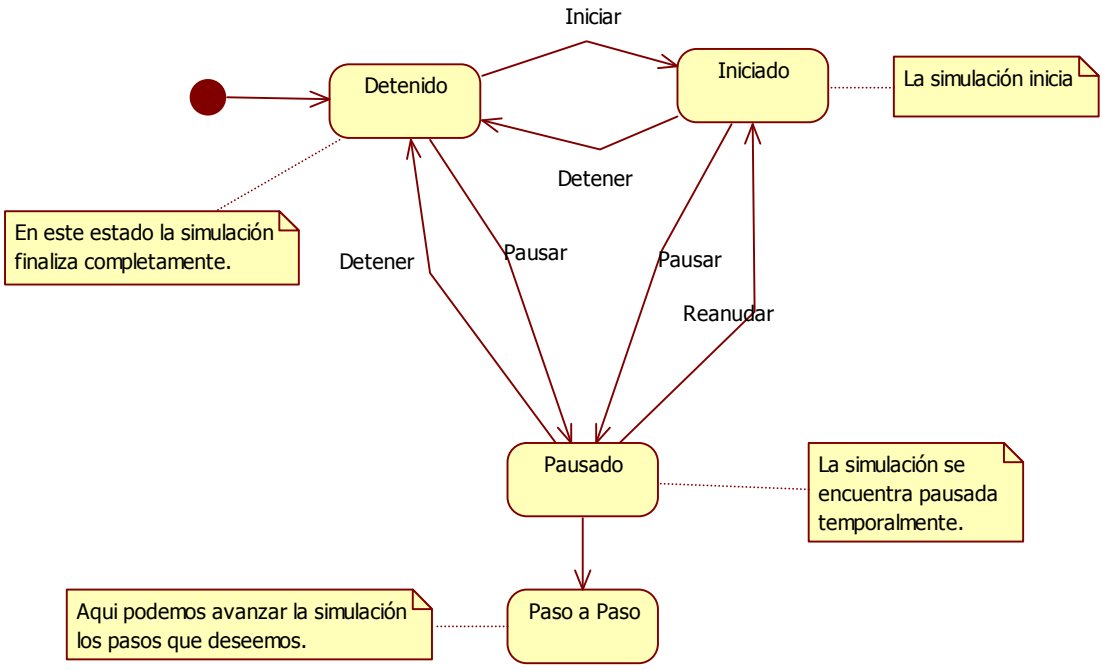


Diagrama 2.15. Diagrama de estado que indica los estados en que se puede encontrar el software de simulación.

2.2.3.7 Interfaces gráficas de usuario

Las librerías MASON, la herramienta que hemos seleccionado para realizar la simulación proveen un conjunto de interfaces gráficas que satisface las necesidades de la simulación que realizamos, por lo tanto a continuación indicamos estas interfaces y su funcionalidad principal.

Los tres principales elementos de la interfaz gráfica son la Consola, la pantalla de simulación o Display y la carta de gráficos.

El completo funcionamiento de la interfaz gráfica MASON se detalla en el manual de usuario en los ANEXOS.

2.2.3.7.1 La consola

La consola permite manejar los parámetros propios de la simulación en sí misma, mostrando los parámetros e indicando la información pertinente en tiempo real, además permite observar la información de los agentes así como del sistema global, permite cambiar de estados de simulación.

Fig. 2.3. Interfaz Gráfica de la Consola de simulación.

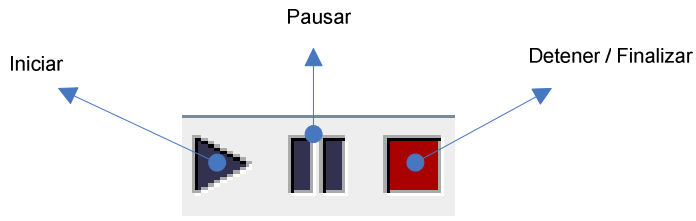


Fig. 2.4. Controles de estado de simulación.

2.2.3.7.2 La pantalla de simulación

Esta permite visualizar los agentes en el mundo 2D, además permite escalar la visualización y provee además utilitarios extras para generar secuencia de video y capturar fotografías, también permite seleccionar los elementos que deseamos observar en la simulación. Puede cerrarse sin que la simulación se detenga, y abrirse nuevamente en otro momento.

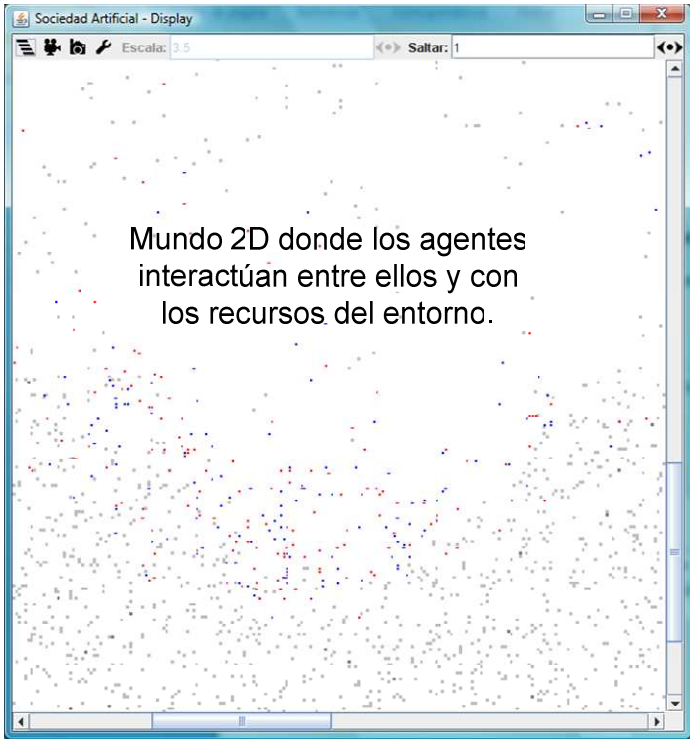


Fig. 2.5. Pantalla de simulación.

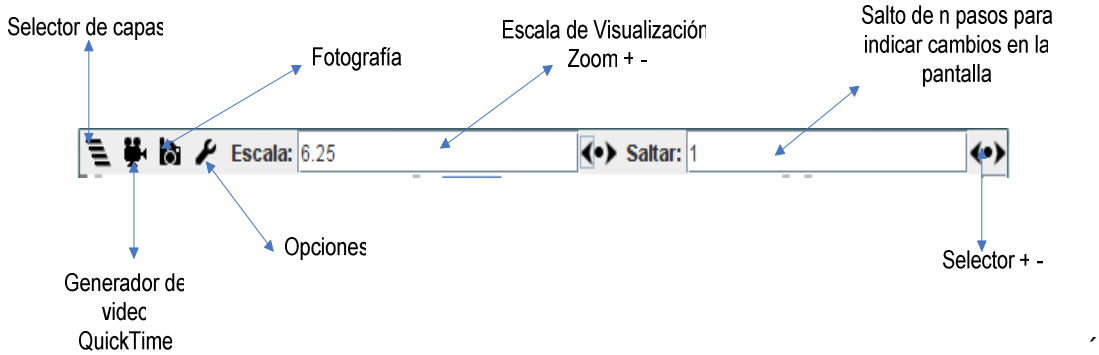


Fig. 2.6. Controles de la pantalla de simulación.

2.2.3.7.3 La carta gráfica

Este elemento permite graficar los valores de los parámetros en un plano cartesiano con ejes X, Y donde X es el tiempo o los ciclos de simulación y en Y puede existir uno o más parámetros, permite configurar y personalizar la visualización de los datos.

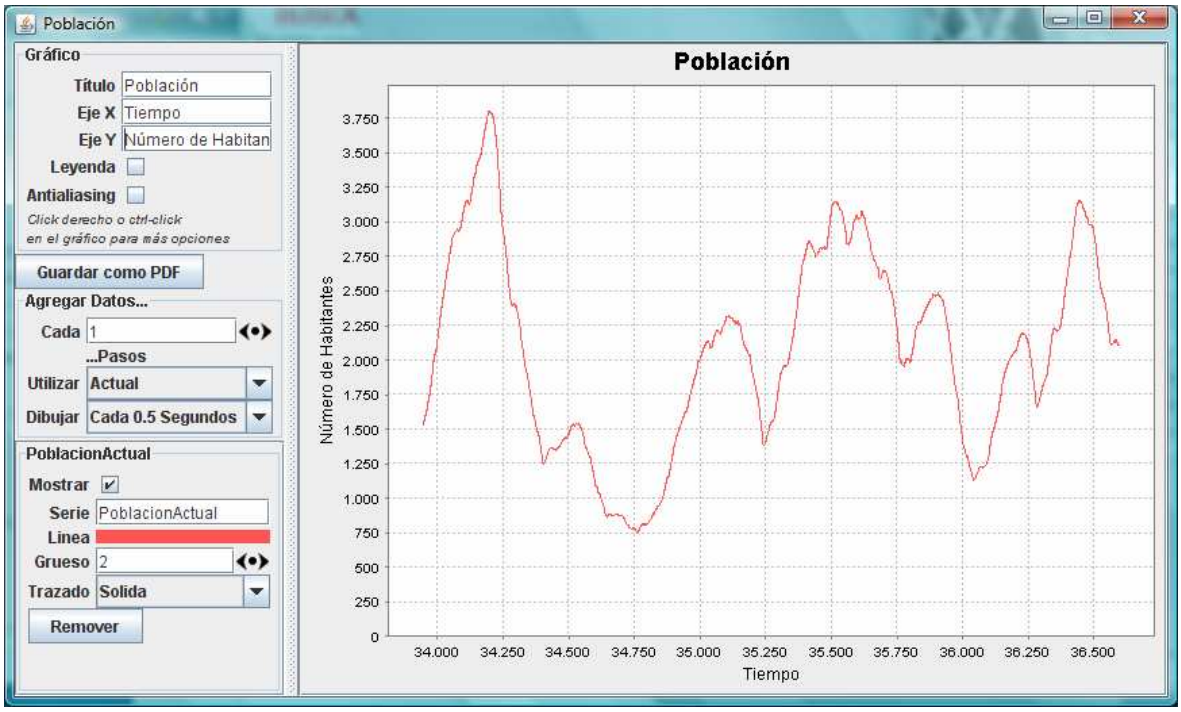


Fig. 2.7. La carta gráfica

Estas interfaces “prefabricadas” permiten optimizar el tiempo de desarrollo de la aplicación.

2.3 PRUEBAS

Durante la implementación del software se ha realizado depuración del código y pruebas experimentales para evitar errores y corregir posibles “bugs”, por esta razón realizaremos a continuación pruebas funcionales y pruebas de rendimiento para poder determinar si el software cumple efectivamente con sus características, y por tratarse de una simulación multi-agentes debemos analizar especialmente el rendimiento del sistema.

2.3.1 PRUEBAS FUNCIONALES

CASO DE PRUEBA	RESULTADO ESPERADO	RESULTADO OBTENIDO
Iniciar la simulación	Simulación iniciada	Simulación iniciada
Detener simulación	La simulación finaliza	Simulación finalizada
Pausar la simulación	Simulación pausada	Simulación pausada
Reanudar la simulación	Simulación empieza desde el punto en que fue pausada sin pérdida de información	Se reanuda normalmente la simulación.
Avanzar la simulación un paso.	Cuando la simulación se encuentra pausada el modo de paso a paso debe activarse y se puede avanzar la simulación un paso.	Simulación paso a paso normalmente.
Guardar el estado de una simulación.	Debe guardarse un archivo con el estado de la simulación.	Se guarda un archivo con la extensión “checkpoint”.
Cargar el estado de una simulación.	Desde un archivo (con la extensión “checkpoint”) que contenga un estado de simulación anterior debe cargarse dicho estado en el sistema.	Se carga normalmente la simulación anterior.
Cargar el estado de una simulación desde un archivo incorrecto	Debe mostrarse una ventana de error.	Se muestra la ventana de error y la aplicación continúa normalmente.
Cambio de parámetros	Se debe permitir el cambio de parámetros y	Los parámetros pueden ser cambiados

	debe reflejarse dicho cambio en la simulación actual.	normalmente y se reflejan los cambios en la simulación en tiempo real.
Activación del inspector de agentes	Al hacer doble click sobre un agente visible del mundo 2D debemos poder observar la información de dicho agente en tiempo real.	El inspector se activa normalmente permitiendo observar el estado interno del agente.
Esconder y mostrar la pantalla de simulación.	Se puede cerrar la pantalla de simulación, y la misma debe continuar, pudiendo reactivar la pantalla posteriormente.	La pantalla de simulación se muestra o se esconde normalmente sin detener o alterar la simulación.
Aumento y disminución de la visión en la pantalla de simulación.	En estado de pausa se puede ampliar y disminuir la visión (zoom) de la pantalla de simulación.	Se puede escalar normalmente la visibilidad de la simulación.
Cambios en los saltos de pasos en la simulación.	Se puede establecer el número de pasos en los que se quiere que se actualice la simulación en la pantalla, alterando la visualización conforme a esto.	El cambio de pasos en la visualización de la simulación se efectúo normalmente sin alterar la simulación.
Selección de las capas de visualización en la simulación.	Se puede seleccionar que capas se desea observar y cuáles no, produciéndose el efecto deseado en la pantalla de simulación, sin que esto afecte a la simulación.	Al seleccionar las capas se puede visualizar el efecto deseado sin que influya esto en el estado real de la simulación.

Registro de video QuickTime (Utilidad de la librería MASON)	Se puede guardar un archivo QuickTime de video que registre la secuencia de simulación visible en la pantalla de simulación.	En ciertas ocasiones se realizó perfectamente el proceso, en otras ocasiones la aplicación colapsó, resultando ser un bug propio de la librería MASON.
Captura de imágenes durante la simulación.	Se puede guardar un archivo de imagen en un momento determinado de la simulación.	Se realizó la captura de imagen sin ningún inconveniente, aunque el proceso puede demorar un poco de tiempo si existen muchos agentes en un momento determinado.
Despliegue de gráficos.	Una carta de gráficos debe desplegarse para la propiedad que se determine.	La carta de gráficos se despliega normalmente, sin alterar la simulación.
Personalización de gráficos.	Los gráficos pueden ser personalizados por el usuario.	Se puede personalizar normalmente las gráficas.
Actualización de datos coherente con las gráficas obtenidas.	Los datos obtenidos de la simulación deben mostrarse adecuadamente en la gráfica en tiempo real.	Los datos se actualizan correctamente reflejando los cambios en las gráficas.

Tabla 2.5. Pruebas funcionales.

2.3.2 PRUEBAS DE RENDIMIENTO

2.3.2.1 Ambiente de pruebas

Puesto que las características propias del ambiente donde se realizan las pruebas de rendimiento son imprescindibles para inferir las características del software que se analiza, describiremos las características del computador donde se realizaron las pruebas:

- PROCESADOR: Core 2 Duo 2.0 GHz
- RAM: 2GB
- SISTEMA OPERATIVO: Windows Vista Home Premium.

2.3.2.2 Casos de prueba

2.3.2.2.1 Caso 1: Rendimiento y número de agentes

Principalmente debemos determinar cómo se comporta el sistema en relación con el número de agentes que pueden existir en la simulación, el número mínimo de agentes es 1 y el máximo 30000. Ya que cada agente representa por sí mismo un hilo independiente de control, consumirá recursos en tiempo de procesador y memoria, siendo el primero el más importante.

Número de agentes promedio	Pasos realizados/ minuto	Consumo de memoria promedio	Utilización de CPU promedio	Pasos entre los que se tomaron los datos
3000	850	56 MB	52%	5400 – 6250
10000	320	58 MB	52%	14650 – 14970
20000	155	60 MB	52%	16360 – 16515
30000	95	63 MB	52%	16930 – 17025

Tabla 2.6. Relación entre el número de agentes y el rendimiento del software

Como podemos observar el rendimiento se degrada en relación al número de agentes, la figura 2.3 nos indica claramente este hecho, al disminuir el número de

pasos/minuto la simulación se vuelve más lenta, esto se debe principalmente a la cantidad de cálculos que debe realizar el sistema. Como podemos observar en la tabla 2.6 la cantidad de agentes influye también en la cantidad de memoria que ocupará teniendo un tope máximo de 63 MB. La utilización del CPU tiene un promedio de 52% en todos los casos pero recordemos que las pruebas fueron realizadas en un computador con doble núcleo lo cual implicaría que un procesador fue utilizado específicamente para el procesamiento de la información.

Las pruebas anteriores se realizaron con la pantalla de simulación activada, pero si desactivamos la misma veremos que existe un ligero aumento del rendimiento del sistema, puesto que para el caso en el que tenemos 30000 agentes los pasos por minuto aumentan de un promedio de 95 a un promedio de 113, es decir un 18% aproximado de incremento del rendimiento, esto es debido principalmente porque el sistema ya no tiene que efectuar operaciones gráficas.

Número de agentes promedio	Pasos realizados/ minuto	Consumo de memoria promedio	Utilización de CPU promedio	Pasos entre los que se tomaron los datos
30000	113	63 MB	53%	22548 – 22661

Tabla 2.7. Ligero aumento del rendimiento cuando se cierra la pantalla de simulación.

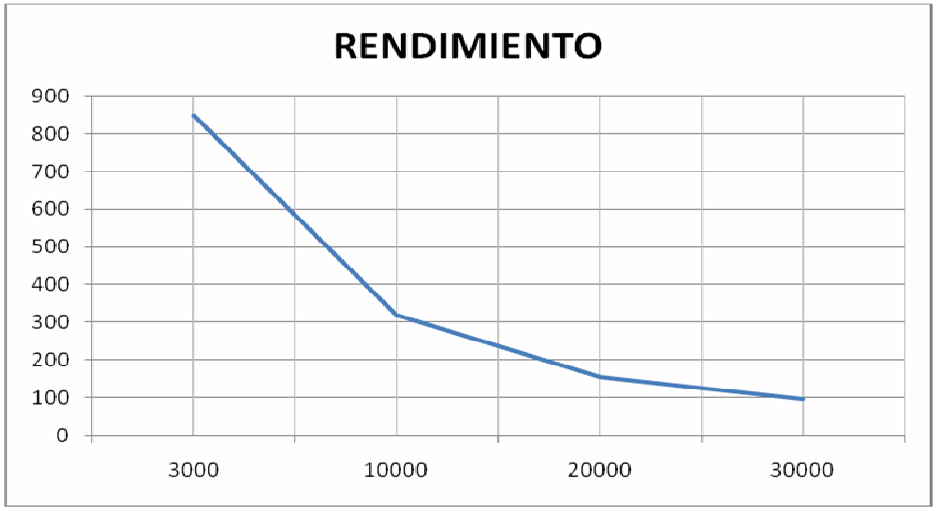


Fig. 2.8. Pasos/minuto (Y) vs número de agentes (X).

Por lo tanto hemos restringido la cantidad máxima de agentes que el usuario puede establecer, para que de esa manera haya un punto tolerable máximo que el sistema pueda soportar, además hemos permitido que el usuario asigne el valor máximo de agentes que pueden interactuar, así que según sus intereses el usuario puede permitir un mayor o menor rendimiento en sus indagaciones o experimentos.

2.3.2.2.2 Caso 2: Rendimiento y la carta de gráficos

El problema de la reutilización de software: componentes, código, librerías, etc. es que si existe una carencia de documentación, el mantenimiento y cambio del sistema se hace sumamente complejo, quitando tiempo y recursos a las actividades principales de un proyecto de software, en el caso de nuestro proyecto MASON provee la librería JFreeChart la cual permite generar gráficas estadísticas como las que utilizamos en nuestro software, sin embargo por las pruebas que hemos realizado existe un deterioro en el rendimiento de la aplicación cuando la carta de gráficos ha estado activado durante algunos momentos, este problema es propio de las librerías y su corrección se hace sumamente compleja por la falta de documentación.

Realizando la simulación con una carta de gráfico que genere gráficas en tiempo real por más de 15 minutos dió como resultado un descenso en el rendimiento del sistema, puesto que la carta de gráficos va acumulando un conjunto de información gráfica abundante que degrada el funcionamiento del sistema.

Número de agentes promedio	**Pasos realizados/ minuto**	**Consumo de memoria promedio**	**Utilización de CPU promedio**	**Minutos después que se activo la carta de gráficos.**
3000	596	71 MB	56%	16
3000	550	71MB	57%	24
3000	482	74MB	57%	40

Tabla 2.8. Decrecimiento en el rendimiento cuando esta activado y visible la carta para gráficos.

Ahora bien, los datos de la tabla 2.8 son válidos cuando la ventana de la carta de gráficos, así como la pantalla de simulación y la consola están visibles, si minimizamos la ventana de la carta gráfica por ejemplo para el último dato de la tabla 2.8 (482 pasos/minuto) obtuvimos un valor de 679 pasos/minuto, lo cual implica un mejoramiento aproximado del 40% en el rendimiento del sistema, estas variaciones son debidas a los cálculos que realiza el sistema para poder graficar.

Si cerramos la ventana de la carta gráfica, el rendimiento vuelve a incrementarse como podemos observar en la tabla 2.9 al valor de 693 pasos/minuto, sin embargo podemos observar que el consumo de la memoria ha aumentado considerablemente y el del CPU también de forma que estos valores no han decrecido al cerrar la ventana de cartas de gráficos, por lo cual podemos inferir que las librerías JFreeChart no liberan los recursos que han consumido mientras se ejecute la simulación lo cual es un defecto de las librerías de software utilizadas.

Número de agentes promedio	**Pasos realizados/ minuto**	**Consumo de memoria promedio**	**Utilización de CPU promedio**	**Pasos entre los que se tomaron los datos**
3000	693	79 MB	60%	74900 – 75593

Tabla 2.9. Datos de rendimiento después de haber cerrado la ventana de cartas de gráficos.

Además podemos apreciar que el nuevo valor de 693 pasos por minuto no está cerca del valor promedio de 850 pasos por minuto que obtuvimos en el primer caso de prueba para una población promedio de 3000 agentes en el mundo 2D.

De esta forma hemos podido analizar el rendimiento y funcionalidad de nuestro sistema después del diseño y la implementación que hemos realizado del mismo. Para mejorar el rendimiento del sistema se debería modificar el motor de simulación MASON, así como algunas librerías internas y además se recomendaría migrar el código a un lenguaje de programación más optimo como puede ser C/C++.

2.3.3 CASOS DE ESTUDIO

A continuación indicaremos unos casos de estudio prácticos que muestran el potencial de la herramienta, así también se vislumbrará la capacidad de aplicar la herramienta en estudios poblacionales, demográficos y sociales.

2.3.3.1 Población y Recursos

Para las sociedades animales, y entre ellas las sociedades humanas la relación que existe entre la disponibilidad de recursos y el bienestar de la población es un factor imprescindible que determinará el crecimiento o disminución de los habitantes del conglomerado social, influyendo así en la natalidad o en la mortalidad.

Ahora bien, nuestra aplicación muestra un modelo de simulación realmente genérico, sin embargo puede indicar aspectos generales de todas las sociedades modernas y antiguas, sin embargo si se desea estudiar un aspecto de una sociedad específica sería más recomendable crear los detalles del modelo específico que se quiere analizar.

Las posibilidades de estudios son varias, sin embargo para este caso de estudio podemos suponer que alguien quiere averiguar los parámetros necesarios para que una población inicial de 1000 habitantes que puede ser una tribu o una población que ha sufrido una gran mortandad –por ejemplo- alcance una población de 10000 habitantes, es decir diez veces la población original, además se determinaría la cantidad de años que se requiere para que se de tal fenómeno, de forma que la población se estabilice alrededor de los 10000 habitantes.

A continuación mediante prueba y error observamos que los siguientes parámetros producen una población estable alrededor de 10000 habitantes:

- Recurso Máximo: 1500
- Dispersión de recursos: 1.0
- Regeneración de recursos: 1.0

- Frecuencia de regeneración: 4

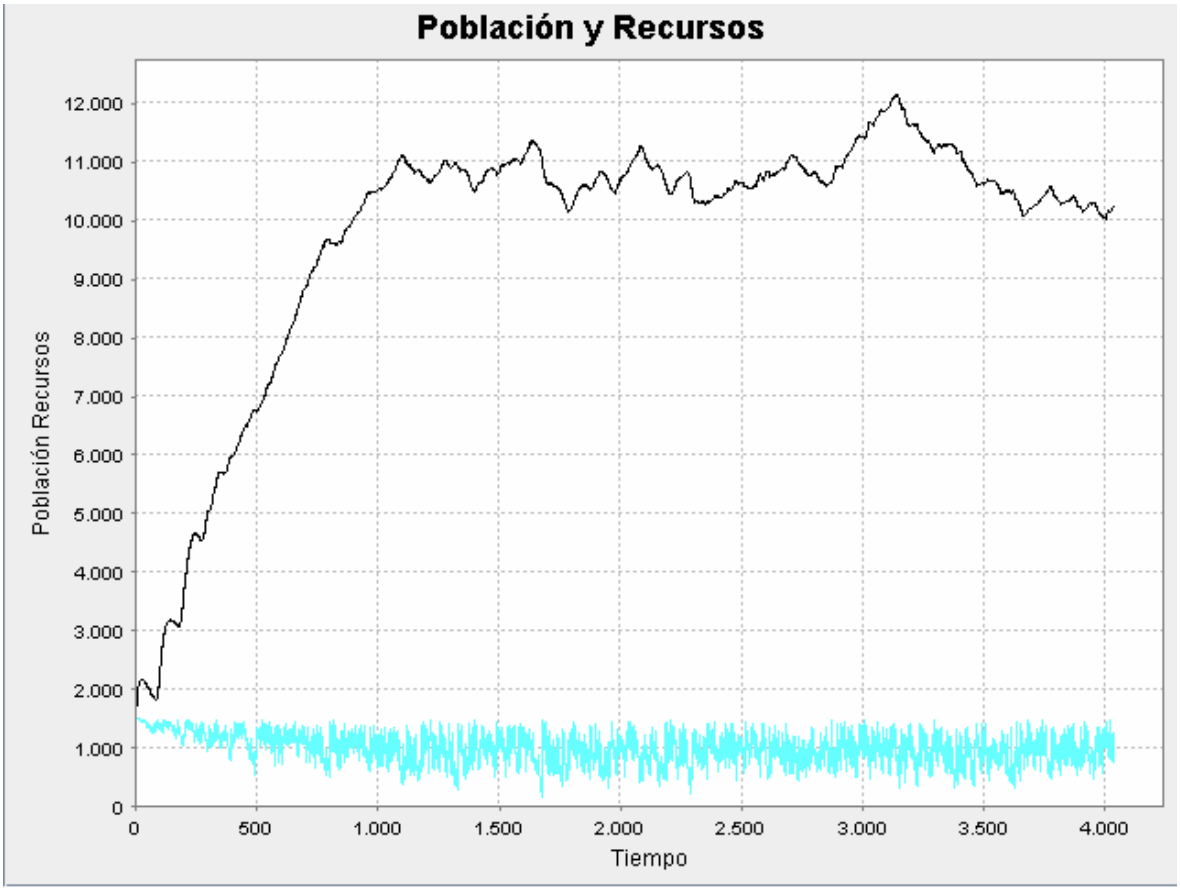


Fig. 2.9. Gráfica de la relación entre recursos (verde) y población (negro) alrededor de 4000 ciclos.

Ahora bien la complejidad de determinar la relación entre los valores específicos de la simulación y los valores de parámetros de un caso de estudio particular depende principalmente de éste y de la asociación que realice el investigador, sin embargo para este caso de estudio podemos determinar que para obtener los resultados deseados se debería aplicar sobre la población las siguientes características:

- Se debe garantizar que los recursos sean asequibles para toda la población, sin que existan concentración de recursos en pocas manos, o preferencia para algunos (dispersión de recursos = 1.0).
- Los recursos consumidos por la población deben ser totalmente regenerados o producidos (Regeneración de recursos =1.0)

- Debe garantizarse que los recursos estén disponibles en los centros de abastecimiento de una forma continua y sin demora, de tal manera que la población pueda acceder a los recursos según sus necesidades (Frecuencia de regeneración = 4)
- En cuanto a los recursos, se puede determinar que bajo las mismas condiciones anteriores para mantener una población alrededor de los 1000 habitantes se requiere un valor aproximado de *Recurso Máximo* de 250, y para los 10000 habitantes un valor de *Recurso Máximo* de 1500, esto implicaría un aumento relativo de 6 veces la capacidad de producción inicial de recursos, esto podría aplicarse a cualquier medio de producción o actividad que determine la obtención de recursos por parte de la población.

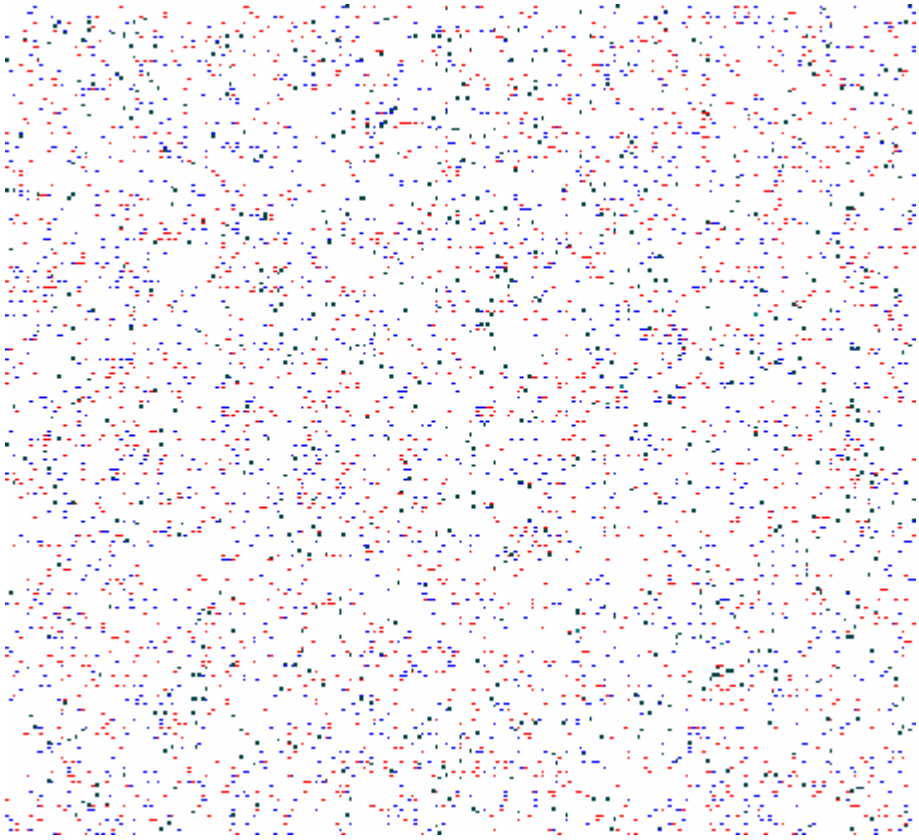


Fig. 2.10. Dispersión de la población en el mundo 2D

Se puede observar también en el plano que forma el mundo 2D, que la población no está concentrada en una misma área, sino dispersa alrededor del plano, lo cual nos indica una adecuada distribución de los bienes a la población a través del tiempo y del espacio.

Podemos observar mediante la Figura 2.9 que se requieren alrededor de 1100 ciclos para que la población alcance el número de habitantes deseados, para esto es necesario conocer los valores de los parámetros que se utilizaron para el caso de estudio:

- Ancho del mundo 2D: 350
- Alto del mundo 2D: 350
- Población Inicial: 1000
- Vida Máxima: 300
- Salud Máxima: 150
- Fertilidad Máxima: 6
- Aptitud Reproductiva máxima: 60
- Aptitud Supervivencia máxima: 50
- Mutación: 0.45

Estos valores de los parámetros son valores que permiten simular un crecimiento muy semejante a las variaciones poblacionales que se presentan en la sociedad humana, la cual generalmente en condiciones normales es de forma exponencial. Todos los valores son los que presenta el sistema por defecto, con excepción del ancho y alto del plano 2D y la población inicial.

Como podemos observar el valor de la vida Máxima es de 300 pasos, este representa la esperanza máxima de vida que tiene un agente lo cuál puede ser comparado con la esperanza de vida que tienen los seres humanos para una sociedad determinada, ya que este es un factor que varía según el tiempo y el espacio, para nuestro caso podemos considerar la esperanza de vida de la población que queremos estudiar en 75 años, por lo tanto si 300 ciclos equivalen a 75 años, entonces 4 ciclos equivaldrían a 1 año humano relativamente.

Entonces 1100 ciclos equivaldrían a 275 años, es decir que se necesitaría esta cantidad de tiempo para que la población alcance dicho número de habitantes, y

habría que sostener la producción de recursos anteriormente mencionada para que la población de esa sociedad se estabilice alrededor de los 10000 habitantes. Durante este periodo de tiempo han aparecido en el simulador alrededor de 12 generaciones, es decir deberán existir y sobrevivir 12 generaciones de seres humanos en condiciones semejantes a las anteriormente mencionadas para alcanzar el objetivo propuesto.

Como vemos la simulación multiagentes con técnicas de ALife permiten el estudio del sistema complejo de variación poblacional de las sociedades humanas, esto puede ser utilizado como una herramienta de predicción y planificación social y demográfica.

2.3.3.2 Población y distribución sexual de los habitantes

A continuación realizaremos un experimento en el que pondremos de relieve la importancia de una distribución de género adecuada en los habitantes de una población, es decir que el número de mujeres sea apropiado para el número de hombres que existen en la población, puesto que en ciertos procesos sociales, como pueden ser las migraciones, las guerras, o las enfermedades endémicas, la distribución sexual de la población se puede ver grandemente afectada, y esto determinará el futuro comportamiento poblacional.

Consideremos que unos investigadores para el caso anterior desean saber qué pasará si por consecuencia de la guerra la distribución de hombres y mujeres de la población total varía en un 30% para los hombres y un 70% para las mujeres, de esta forma desean predecir lo que acontecería si esta tendencia continuaría.

Para esto establecemos los parámetros de simulación con los valores anteriores, para que la población alcance 10000 habitantes y después de un tiempo de estabilidad procederemos a alterar la distribución sexual de la población para ver como se comportaría el sistema global.

Los valores iniciales de los parámetros son:

- Ancho del mundo 2D: 350
- Alto del mundo 2D: 350
- Población Inicial: 1000
- Vida Máxima: 300
- Salud Máxima: 150
- Fertilidad Máxima: 6
- Aptitud Reproductiva máxima: 60
- Aptitud Supervivencia máxima: 50
- Mutación: 0.45
- Distribución sexual: aleatoria
- Recurso Máximo: 1500
- Dispersión de recursos: 1.0
- Regeneración de recursos: 1.0
- Frecuencia de regeneración: 4

Con estos valores iniciamos nuestra simulación, lo cual nos indica que es una población que tiene una capacidad para producir recursos aproximadamente para 10000 habitantes, y que los recursos están distribuidos uniformemente en la población; además que la capacidad de producción o regeneración de los recursos es continua y constante. Estos fueron los valores de los parámetros que utilizamos en el caso de estudio anterior.

Como podemos ver en la figura 2.11, la variación entre la población femenina (color azul) y la población masculina (color rojo) durante los primeros 1500 ciclos es pequeña y permite que la población se desarrolle normalmente. A partir de los 1500 ciclos variamos la distribución sexual de la población al valor de 0.3, de forma que los nacimientos femeninos se van a incrementar y los masculinos van a disminuir, esto no es exactamente lo que pasa en una guerra donde los hombres mueren y las mujeres quedan viudas, sin embargo sirve para representar el fenómeno de variación del género en la población, y lo que acontece en el sistema social si la tendencia perdura.

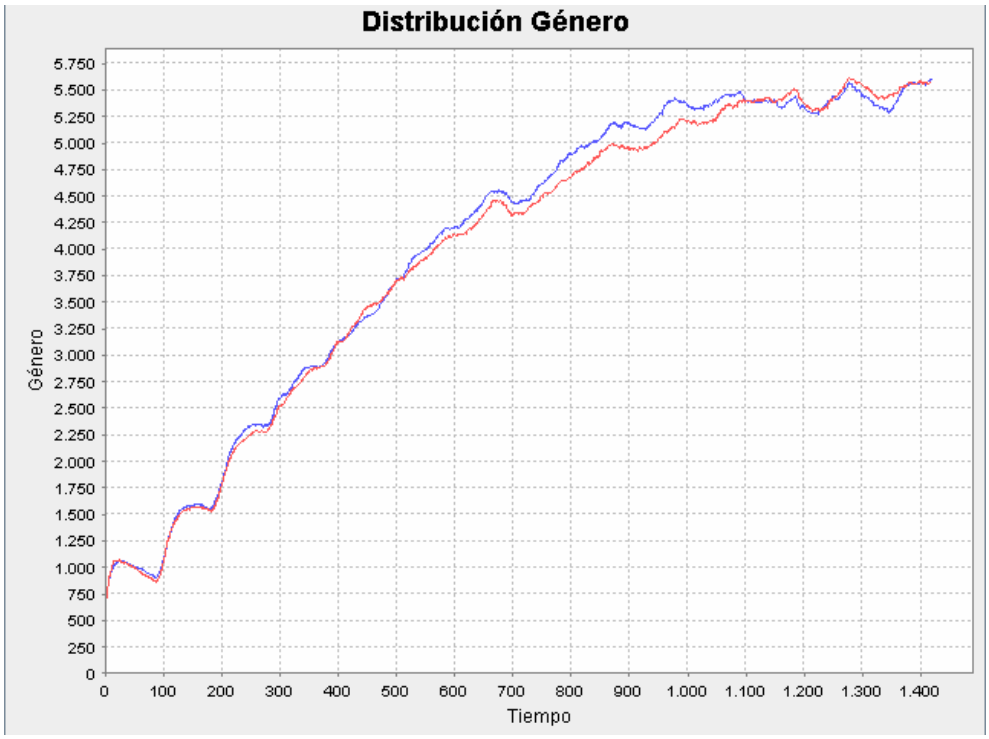


Fig. 2.11 Variación del género sexual (hombres – rojo, mujeres - azul) en la población durante los primeros 1500 ciclos.

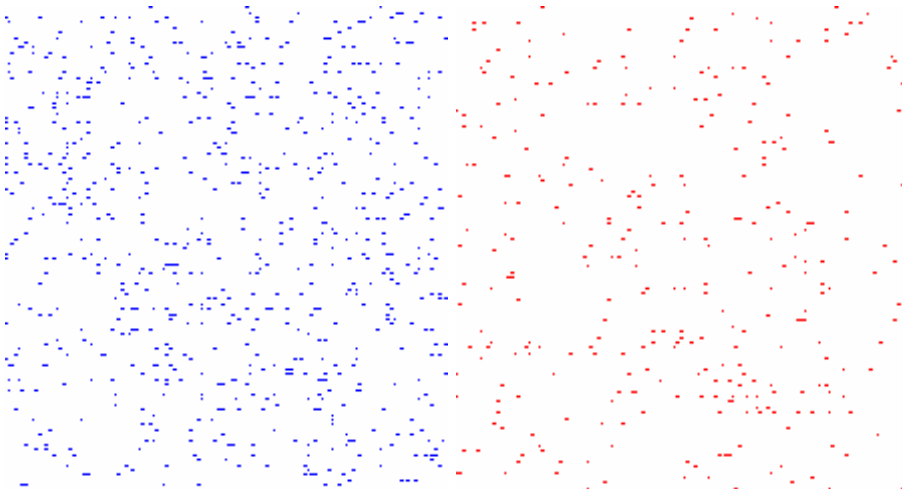


Fig. 2.12. Densidad de la población femenina (izquierda – agentes azules) y población masculina (derecha – agentes rojos) para una misma zona del mundo 2D

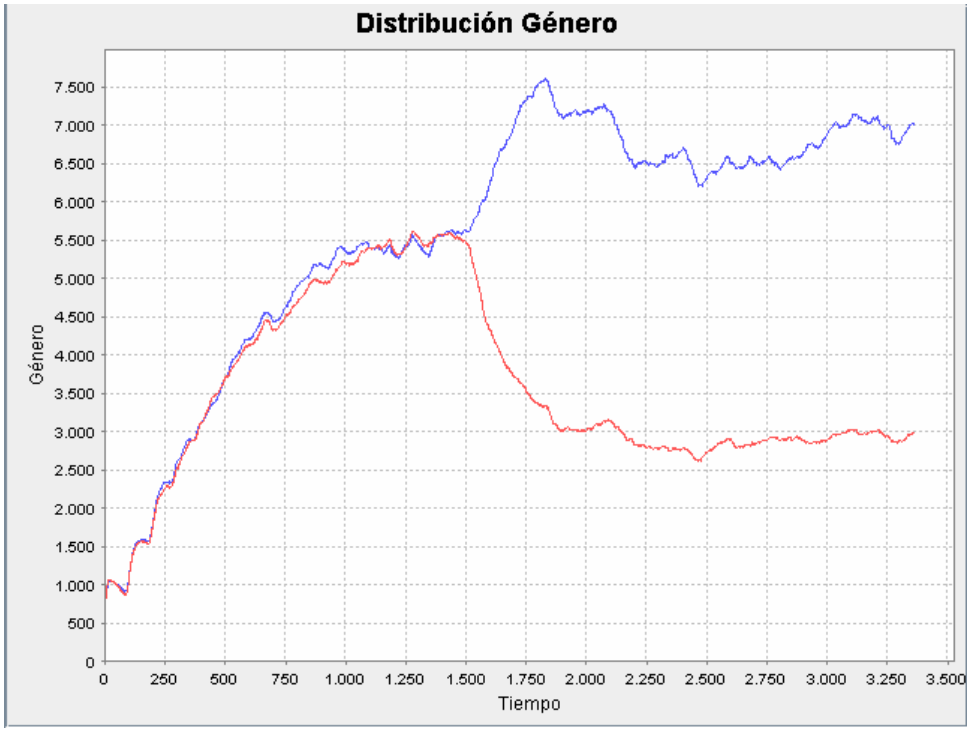


Fig. 2.13. Abrupta variación en la población femenina y masculina alrededor del ciclo 1500 hasta aproximadamente el ciclo 1800, en el que relativamente se estabiliza el sistema.

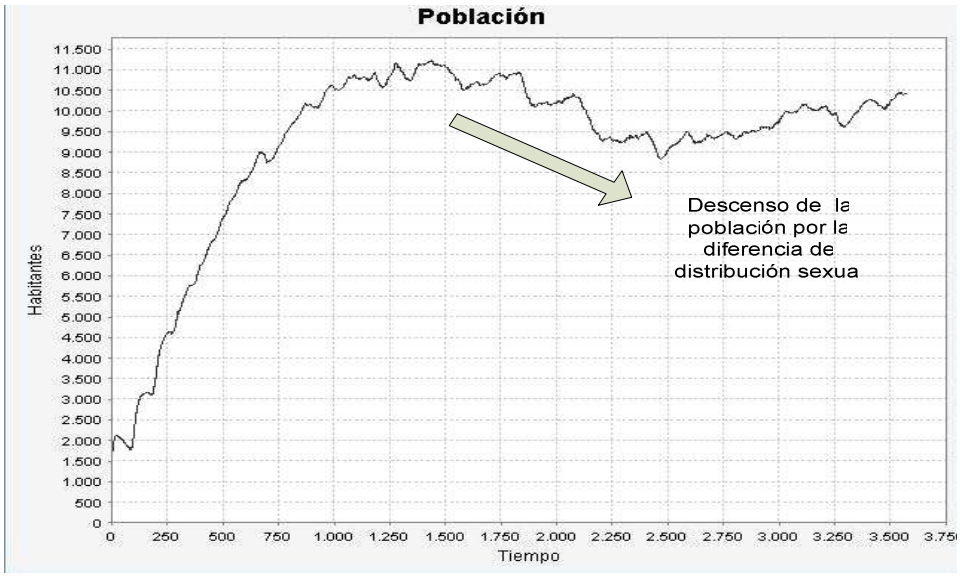


Fig. 2.14 Imagen que muestra la variación general de la población

De los resultados obtenidos, que son visibles en las gráficas que hemos obtenido podemos determinar qué:

- A partir del ciclo 1500 hasta el ciclo 1800 existe una abrupta variación de la población masculina y femenina, según las consideraciones del caso de estudio anterior, este lapso equivaldría a 75 años, después de los cuáles el sistema empieza a estabilizarse, manteniendo la tendencia de población en el 70% de mujeres y el 30% de hombres.

- En la gráfica general de la población (figura 2.14) podemos observar que existe un lento descenso de la población desde el ciclo 1500 hasta aproximadamente el ciclo 2480 donde empieza a existir un ligero crecimiento poblacional. Es decir podemos establecer que a partir de la variación sexual de la población aproximadamente por 245 años existirá un descenso poblacional desde los 11000 habitantes hasta los 8000.

- Hay que mencionar que esta tendencia se presenta cuando los seres humanos se aparean normalmente sin inhibiciones culturales, como por ejemplo el matrimonio, lo que sin duda en casos como estos repercutiría en un mayor decrecimiento poblacional puesto que muchas mujeres quedarían solteras y no podrían engendrar.

- Hay que mencionar que durante la simulación no se cambio las características económicas de los recursos, lo cual en caso de guerras o de pestes no siempre se mantiene estable.

Estas tendencias nos indican una probable predicción de tendencias cuando existen variaciones en el género sexual de la población.

2.3.3.3 Población, natalidad y mortalidad

A continuación realizaremos otro caso de estudio donde podremos analizar la relación que existe entre la mortalidad y la natalidad en la población. Para esto consideremos que para la población de estudio del primer caso, la productividad desciende por hambruna y sequía de forma que los Recursos Máximos descienden a 1/3 de lo necesario para que subsistan 10000 habitantes, la probabilidad de regeneración de recursos se mantendrá igual, pero la dispersión

de recursos va a variar a 0.5 de forma que la distribución de recursos no sea tan equitativa, y haya una mayor dificultad para conseguir recursos, la frecuencia de regeneración también va a ser igual. La aptitud de los agentes para conseguir recursos, y sus características se mantendrán igual. Por lo tanto los parámetros con que vamos a afectar la simulación serán:

- Recursos Máximo: 500
- Dispersión de recursos: 0.5
- Regeneración de recursos: 1.0
- Frecuencia de regeneración: 4

Los otros parámetros se mantienen igual, de esta forma vamos a estudiar cómo se verá afectada una población prospera, por medio de una baja en los recursos que consume a 1/3 de la capacidad inicial.

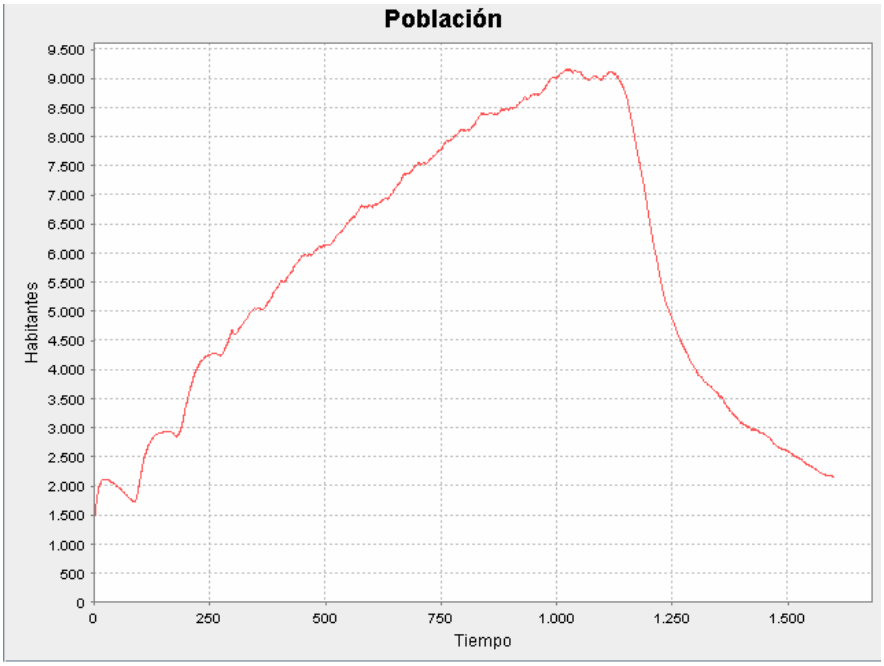


Fig. 2.15 Variación Poblacional por carestía de recursos

Durante los primeros 1100 ciclos dejamos que la simulación tenga un crecimiento normal, bajo las condiciones del primer caso de estudio, y a partir de dicho ciclo cambiamos los parámetros descritos para ver como se ve afectada la población.

Podemos observar que existe un decrecimiento acelerado de la población, ya que existe una hambruna generalizada y no todos los agentes logran conseguir recursos para subsistir, esto produce un gran incremento en la mortalidad, y un descenso en la natalidad, consecuencia de la disminución de habitantes.

Analizamos los datos hasta el ciclo 1600, las conclusiones que podemos inferir de la simulación son las siguientes:

- Del ciclo 1100 al 1600, la población decrece aproximadamente de 9000 habitantes a 2100, esto equivale a que en 125 años la población se ha reducido en un 76.67%.

- Según la gráfica de mortalidad podemos observar que durante los ciclos 1100 y 1250 la mortalidad alcanza un incremento considerable, esto implicaría que durante los primeros 37.5 años de la recesión existiría un incremento de la mortalidad media de casi el 50% por ciento. Posteriormente este índice decrece y se estabiliza.

- Según la gráfica de natalidad observamos que desde el ciclo 1100 existe un decrecimiento continuo de la natalidad, y se puede observar que en los 37.5 años de la hambruna donde existe la mayor mortalidad, el descenso de la natalidad es alrededor de un 50%, sin embargo este índice ya no se recupera y va en descenso continuamente.

Como hemos podido ver se podría predecir el efecto que tendría la carencia de recursos en una población, para poder tomar las precauciones necesarias para evitar el aniquilamiento de poblaciones enteras.

Por los casos analizados y propuestos, la herramienta se muestra como un poderoso utilitario en las ciencias que se encargan del estudio y planificación de las sociedades humanas, sin embargo, creemos que éste es un paso importante y genérico en soluciones más específicas y concretas en el ámbito de la predicción

planificación social. Cabe mencionar también la gran gama de posibilidades que se da con la configuración de los distintos parámetros que posee el simulador.

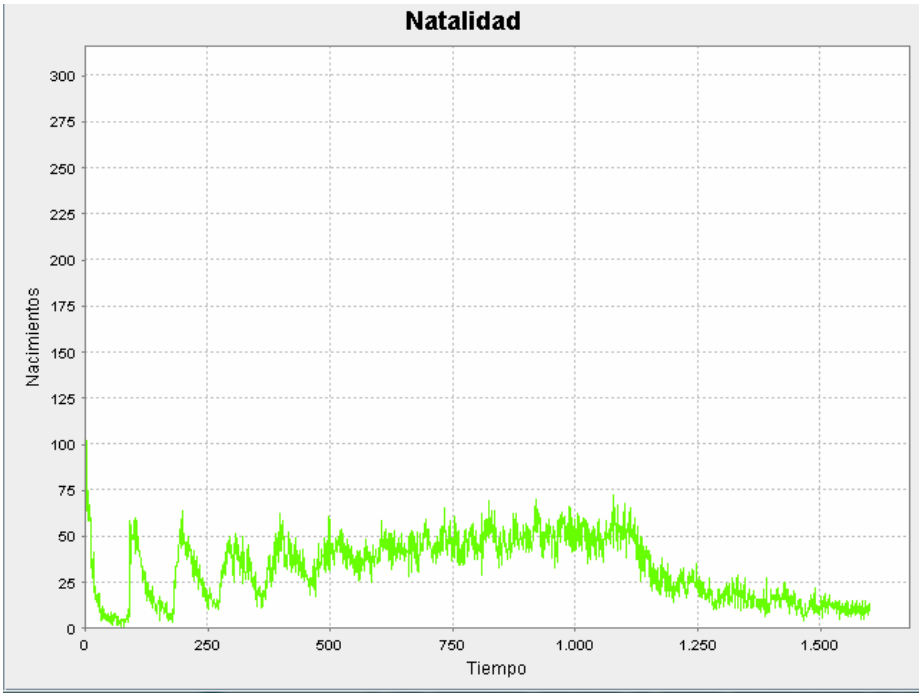


Fig. 2.16. La Natalidad a través del tiempo

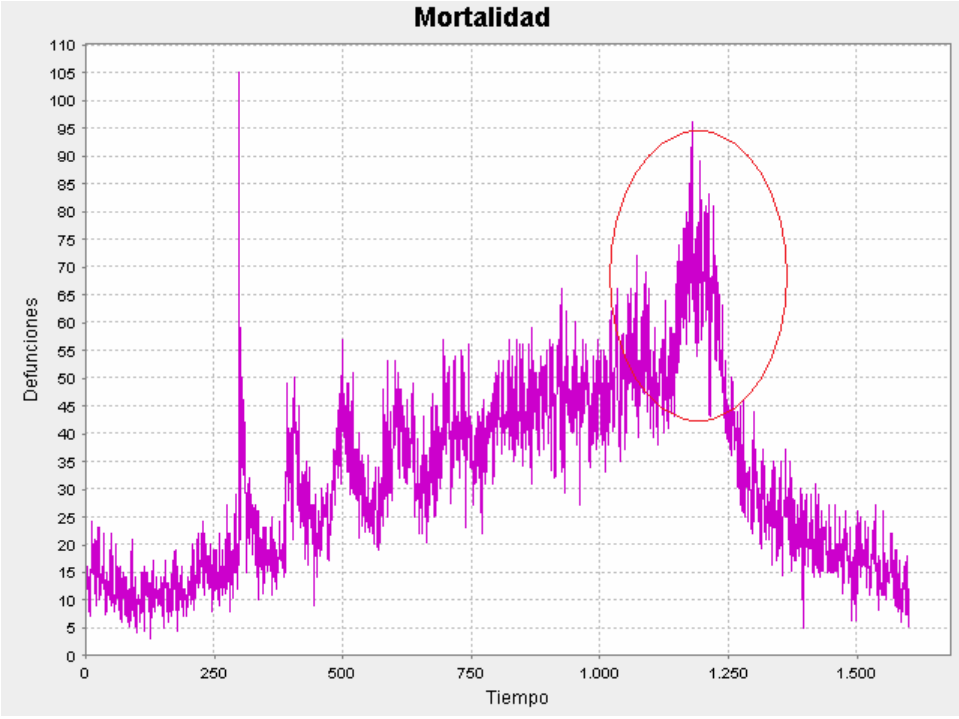


Fig. 2.17. La mortalidad a través del tiempo

CAPITULO III: CONCLUSIONES Y RECOMENDACIONES

3.1 CONCLUSIONES

- Las posibilidades de análisis y experimentación con el modelo de simulación que hemos generado son amplias, dando lugar a un pequeño “laboratorio social”.
- Es importante recalcar que sin un factor de mutación, la población converge a ciertas características comunes que pueden ser útiles en ciertas etapas del entorno cambiante, pero en otras puede producir la extinción de la población, por lo tanto la mutación de los caracteres hereditarios es una característica imprescindible de simulaciones basadas en ALife.
- Las relaciones entre población y recursos ciertamente muestra la importancia de la planificación productiva en las sociedades modernas, juntamente con su repercusión en los índices de natalidad y mortalidad.
- ALife o Vida Artificial es una nueva rama del conocimiento que permite el desarrollo de nuevas técnicas para arte, entretenimiento, informática y ciencias de la computación, así mismo amplía el horizonte de las simulaciones computarizadas en ramas como la medicina, la biología, la ciencia social, la robótica y otras ramas del saber humano donde el ingenio pueda aplicar ALife.
- Las Sociedades Artificiales ocupan un rol cada vez más importante en las ciencias sociales, biológicas y psicológicas donde se estudia y analiza el comportamiento e interacción de los individuos que pertenecen a una colectividad.
- Las Sociedades Artificiales se simulan específicamente en modelos computacionales basados en agentes, donde la acción de cada agente es independiente de la de otros agentes.

- Las simulaciones basadas en agentes permiten el aparecimiento de comportamientos emergentes, es decir que no estaban previamente programados.
- La simulación a nivel de agentes, es un nuevo paradigma en el campo de la simulación, distinto a la simulación a nivel de sistema, los agentes permiten una mayor aproximación a un sistema real que está compuesto de varios elementos autónomos entre sí, pero que interactúan entre ellos para formar el sistema global.
- Se puede observar que un conjunto de reglas sencillas que rigen el comportamiento de los agentes, produce comportamientos emergentes complejos a nivel social.
- Los investigadores podrían formar modelos de simulación específicos para probar teorías y predecir fenómenos de acuerdo a ciertos datos y parámetros, utilizando ALife o Sociedades Artificiales.
- La dificultad de simular una Sociedad Artificial humana radica en la complejidad del fenómeno cultural humano, lo cual implica que la complejidad cognitiva (inteligencia artificial) de los agentes debe ser mayor.
- Al surgimiento de un mayor o menor grado de cultura en una Sociedad Artificial se le puede denominar Cultura Artificial, y es un fenómeno emergente.
- Ya que la dificultad de integrar todos los fenómenos sociales humanos en una Sociedad Artificial es elevada, se pueden formar simulaciones específicas para analizar un problema o una cultura específica, lo cual sin duda será de ayuda a sociólogos, antropólogos, sociobiólogos y psicólogos sociales, los cuales podrían disponer, de esta forma, de un laboratorio de escenarios posibles “que tal si…” .
- El rendimiento de las simulaciones basadas en agentes se ve limitado por la capacidad de memoria y procesamiento que tenga un ordenador, ya que cada nuevo agente que se integre a la simulación consumirá recursos, por eso en simulaciones a gran escala se podría optar por el procesamiento distribuido.

- Existen muchas herramientas y librerías de software que agilitan el proceso de implementación de una Sociedad Artificial o de una simulación multiagente.
- Con el avance de técnicas de Inteligencia artificial, con el aumento de la capacidad de procesamiento de los computadores, y con un mayor conocimiento de la forma como los humanos adquieren conocimiento y crean cultura, podrá en el futuro hacerse simulaciones más prácticas y aproximadas a las sociedades humanas en varios de sus aspectos.
- En fin ALife y las Sociedades Artificiales son un nuevo campo de investigación en el área de las ciencias de la computación y la ingeniería de software.

3.2 RECOMENDACIONES

- Se recomienda seleccionar adecuadamente la herramienta o librerías de software, según las necesidades de nuestro proyecto, puesto que si bien estas herramientas podrían agilitar grandemente el proceso de desarrollo, en cambio, podrían afectar el rendimiento o las características propias de la simulación.
- Se recomienda establecer un límite al número de agentes máximos que soportará una simulación multiagentes, para evitar fallas en el rendimiento de la simulación.
- En el caso de simulaciones a gran escala que requieran de una gran capacidad de procesamiento se recomienda el procesamiento distribuido en red, lo cual agilitará el proceso y los resultados.
- Se recomienda estructurar adecuadamente el código fuente y la implementación de la simulación, de forma que las acciones que realicen los agentes sean lo menos costosas en cuanto a rendimiento se refiere.
- Realizar una mayor integración entre las ciencias sociales y la informática puede ser un forma de ampliar los horizontes cognitivos y tecnológicos en nuestra nación.

- Las sociedades artificiales y ALife son dos temáticas que abren muchos posibles proyectos de titulación o tesis de grado en amplios campos de la ingeniería de sistemas, por eso se recomienda tomarlos en cuenta para futuros proyectos.
- Se recomienda la investigación en temas basados en tecnologías de agentes, lo que sin duda beneficiará a la ingeniería de sistemas en nuestro país.
- Se sugiere que se incentive la investigación en temas novedosos, en el pensum académico de la facultad de sistemas, de forma que los estudiantes puedan estar más capacitados para lidiar en el cambiante mundo tecnológico de hoy día, y formar así generadores y creadores de tecnología y no meros autómatas que repitan soluciones anteriormente establecidas y que no satisfacen problemas propios de nuestro medio.

REFERENCIAS BIBLIOGRÁFICAS

Libros y Manuales

- ADAMATZKY, Andrew; KOMOSINSKI, Maciej. Artificial Life models in software. Springer. USA. 2005.
- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. The Unified Modeling Language Reference Manual. Segunda edición. Addison-Wesley. USA. 2004.
- HALES, David; EDMONDS, Bruce; NORLING, Emma; ROUCHIER, Juliette. Multi-Agent-Based Simulation III. Springer. USA. 2005.
- LANGTON, Christopher; Varios autores. Artificial Life an overview. Cuarta edición. MIT Press. Inglaterra.1998.
- LINDEMANN, Gabriela; MOLDT, Daniel; PAOLUCCI, Mario. Regulated Agent-Based Social Systems. Springer. USA. 2005.
- LIU, Jiming; JIN, XiaLong; TSUI, Kwok Ching. Autonomy Oriented Computing from problem solving to complex systems modeling. Kluwer Academic Publishers. USA. 2005.
- OGBURN, William; NIMKOFF, Meyer; Sociología. Decimo segunda edición. Aguilar. Madrid, 2001.
- PRESSMAN, Roger. Software Engineering a practitioner's approach. Quinta edición. McGraw-Hill. USA. 2001.
- SIGGRAPH 98 Course 22 Notes. Artificial Life for Graphics, Animation, Multimedia, and Virtual Reality.USA.1998.
- WINSTON, Patrick Henry. Inteligencia Artificial. Addison-Wesley. Cuarta Edición. USA. 2000.

Artículos

- BONABEAU, Eric; ICOSYSTEM CORPORATION. Agent-Based modeling: Methods and techniques for simulating human systems. Cambridge. 2002.

- CRAWFORD, Robert. What´s it all about ALife?. Technology Review Abril 1996.
- FURUTA, Hitoshi; YASUI, Masahiro; DEPARTAMENTO DE INFORMÁTICA DE LA UNIVERSIDAD DE KANSAI. Evacuation simulation in underground mall by artificial life technology. Japón. 2003.
- GESSLER, Nicholas; UCLA HUMAN COMPLEX SYSTEMS CENTER. Evolving artificial cultural things-that-think and work by dynamical hierarchical synthesis. USA. 2003.
- LA GACETA ELECTRÓNICA DE EL COLEGIO DE SAN LUIS. Grupo de modelado de organizaciones del PEPI, Primer Congreso sobre Simulación Social. México. 2006.
- LENSKI, Richard; OFRIA, Charles; PENNOCK, Robert; ADAMI, Christopher; MICHIGAN STATE UNIVERSITY. The evolutionary origin of complex features. Nature vol. 423. May 2003.
- LENSKI, Richard; OFRIA, Charles; WANG, Jia Lan; WILKE, Claus; ADAMI, Christopher. Evolution of digital organism at high mutation rates leads to the survival of the flattest. Nature vol. 412. July 2001.
- MICELI, Jorge; GUERRERO, Sergio; QUINTEROS, Ramón; DIAZ, Diego; UNIVERSIDAD DE BUENOS AIRES. Teorías de la complejidad y el caos en ciencias sociales, modelos basados en agentes y sociedades artificiales. Buenos Aires. Sin año.
- SARKAR, Palash; INDIAN STATISTICAL INSTITUTE. A brief history of cellular automata. India 2003.
- SIMS, Karl; THINKING MACHINES CORPORATION. Evolving virtual Creatures. Cambridge. 1994.

Direcciones electrónicas

- AGENT TECHNOLOGY GROUP. A-Globe. http://agents.felk.cvut.cz/aglobe, 2007.
- ASCAPE. http://ascape.sourceforge.net/, 2007.
- AMBLER, Scott; AMBYSOFT. The Agile Unified Process (AUP). http://www.ambysoft.com/unifiedprocess/agileUP.html, 2006.

- ANONIMO. Artificial Societies Links. http://www.geocities.com/goldenziby/p.html .
- ANUNZIATO, Mauro. Artificial Societies. http://www.plancton.com/artsoc/asociety.htm, 2005.
- BREVE. The Breve Simulation Environment. http://www.spiderland.org/breve, 2008.
- CASES, Blanca; DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS DE LA UPV/EHU. Kerman 3D - Vida Artificial. http://members.tripod.com/~MoisesRBB/vida.html, 2008.
- UNIVERSITY OF MICHIGAN. Center for the study of complex systems. http://www.cscs.umich.edu/, 2008.
- COUGAAR. Cougaar Agent Architecture. http://www.cougaar.org/, 2007.
- CRESS. Center for Research in Social Simulation. http://cress.soc.surrey.ac.uk/, 2007.
- ECOLAB. http://ecolab.sourceforge.net/.
- GEORGE MASON UNIVERSITY. MASON Multiagent Simulation Toolkit. http://cs.gmu.edu/~eclab/projects/mason/, 2008.
- GERSHENSON, Carlos. Artificial Societies of Intelligent Agents. http://cogprints.org/1477/, 2007.
- GESSLER, Nicholas. Artificial Culture - A Simulation Laboratory for the Human Complex Systems Program. http://gessler.bol.ucla.edu/, 2008.
- GLT LIFE. GltLife – Conways Game of Llife for OpenGL. http://www.nigels.com/glt/gltlife/, 2002.
- JADE. Java Agent DEvelopment Framework. http://jade.tilab.com/, 2007.
- JAS. Java Agent-Based Simulation Library. http://jaslibrary.sourceforge.net/, 2006.
- JASSS. Journal of Artificial Societies and Social Simulation. http://jasss.soc.surrey.ac.uk/, 2008.
- IBM. alphaWorks: Agent Building and Learning Environment. http://www.alphaworks.ibm.com/tech/able, 2005.
- ISA. ISA - Asociación Internacional de sociología. http://www.isa-sociology.org/sp/, 2008.

- ISAL. International Society of Artificial Life. http://www.alife.org/, 2008.
- KOMOSINSKI, Maciej; ULATOWSKI Szymon. Framsticks – Artificial Life – 3D Evolution and Simulation. http://www.frams.alife.pl/, 2007.
- LENSKI, Richrad; MICHIGAN STATE UNIVERSITY. Lenski Lab Web Page. http://myxo.css.msu.edu/, 2008.
- LA FLECHA DIARIO DE CIENCIA Y TECNOLOGIA. Desarrollan la primera simulación informática de una sociedad compleja. http://www.laflecha.net/canales/ciencia/200504254/, 2006.
- MARTINEZ, Juan Antonio. Teoría de Juegos y el Dilema Del Prisionero. http://oasis.dit.upm.es/~jantonio/documentos/revistas/teoriajuegos/teoriajuegos.html.
- NETLOGO. NetLogo Home Page. http://ccl.northwestern.edu/netlogo/, 2007.
- NEW TIES. New Ties Portal - New and Emergent World models Through Individual, Evolutionary, and Social Learning. https://www.new-ties.org, 2007.
- REPAST. Recursive Porous Agent Simulation Toolkit. http://repast.sourceforge.net/, 2007
- RED CIENTÍFICA. Gaia – Vida Artificial. http://www.redcientifica.com/gaia/va/va_c.htm.
- SANTA FE INSTITUTE. Artificial Societies and the Social Sciences. http://www.santafe.edu/research/publications/wpabstract/200203011, 2002.
- SUGARSCAPE. Growing Agent-Based Artificial Societies. http://sugarscape.sourceforge.net/ .
- SWARM. Main Page Swarm Wiki. http://www.swarm.org/wiki/Main_Page, 2008.
- VALDIOSERA, Cuauhtémoc. La Vida Artificial y las ciencias computacionales. http://www.cimac.org.mx/especiales/mujerytecnologia/lavidaartificial.html.
- VENTRELLA, Jeffrey. Artificial Life Software and research. http://www.ventrella.com/, 2008.

- WIKIPEDIA. Vida Artificial – Wikipedia la enciclopedia libre. http://es.wikipedia.org/wiki/Vida_artificial#Naturaleza_del_.C3.A1rea, 2008.
- WIKIPEDIA. Artificial Society. http://en.wikipedia.org/wiki/Artificial_Society, 2008.