



SISTEMA MULTIAGENTE APLICADA A LA GEÓMATICA

TITULACIÓN INTEGRAL TESIS

que para obtener el título de
INGENIERO EN SISTEMAS COMPUTACIONALES

Presenta
JESÚS LEONARDO ZAMBRANO OLGUIN

No. De Control: 141000267

Mazatlán; Sin, diciembre de 2019



1.1 PLANTEAMIENTO DEL PROBLEMA	1
1.2 OBJETIVO.....	2
1.3 OBJETIVOS ESPECÍFICO	3
1.4 JUSTIFICACION	3
2.1 AGENTES	4
2.1.1 TIPOLOGIA	5
2.1.2 MODELOS BASADOS EN AGENTES	11
2.2 SISTEMA MULTIAGENTE	12
2.2.1 SIMULACION MULTIAGENTE.....	13
2.2.2 LAS METODOLOGIAS	14
2.3 METODOLOGIAS PARA SMA.....	16
2.3.1 PASOS ANTES DE CONSTRUIR UNA SMA	17
2.3.2 METODOLOGIA GAIA	17
2.3.3 VENTAJAS Y DESVENTAJAS DE LA METODOLOGIA GAIA	19
2.4 DESCRIPCION GENERAL DE LA PLATAFORMA DE AGENTES QUE SE UTILIZO.	19
2.4.1 PROPIEDADES DE LA PLATAFORMA	20
2.4.2 VISION GENERAL DE LA USABILIDAD	21
2.4.3 CAPACIDAD DE FUNCIONAMIENTO DE CADA PLATAFORMA DE AGENTE	22
2.4.4 VISION GENERAL DE LA PRAGMATICA	23
2.4.5 VISION GENERAL DEL LENGUAJE DE PROGRAMACION.....	23
2.4.6 CUMPLIMIENTO FIPA	24
2.4.7 RESUMEN DEL CUMPLIMIENTO DE FIPA.....	25
3.1 METODOLOGIA BDI.....	26
3.1.1 LOS AGENTES BDI TIENEN 3 BASES DE DATOS.....	27
3.1.2 MODELO MINERO EN BUSQUEDA DE ORO	30
3.1.3 EXPERIMENTOS	33
3.2 METODOLOGIA GAIA	35
3.3 PROCESOS PARA HACER LOS MODELOS	36
3.3.1 EL PROCESO DE ANÁLISIS	36
3.3.2 PROCESO DE DISEÑO	36
SIMULACION DE INFECCION EN UN AREA DE LA CIUDAD DE ZONA CENTRO.	42

4.1 MODELO SI SIMPLE	42
4.1.1 FORMULACIÓN	42
4.1.2 SECCIÓN GLOBAL	43
4.1.3 VARIABLES GLOBALES	43
4.1.4 MODELO DE INICIALIZACIÓN.	43
4.1.5 EXPERIMENTAR.....	44
4.2 Gráficos.....	44
4.2.1 FORMULACIÓN	44
4.2.2 VARIABLES GLOBALES	44
4.2.3 CONDICIÓN DE PARADA	45
4.2.4 MONITOR	46
4.2.5 GRAFICO.....	46
4.3 Integración de datos SIG.....	47
4.3.1 CARGA DE ARCHIVOS SIG.....	47
4.3.2 RESULTADO.....	47
4.4 Movimiento en el gráfico.....	48
4.4.1 FORMULACIÓN	48
4.4.2 VARIABLES GLOBALES	48
4.4.3 AGENTE DE PERSONAS.....	48
4.4.4 RESULTADOS	49
4.5 SIMULACION EVACUACION DE LA CIUDAD	50
4.5.1 CREACION DE CODIGO DE MODELO DE ESCAPE DE LA CIUDAD	50
4.5.2 PARÁMETROS	50
4.5.3 AGENTIFICACIÓN DE DATOS SIG	51
4.5.4 CREACIÓN DE MODELO SIMPLE_BDI.....	52
4.5.5 Aspectos de agentes y catástrofes en la simulación	54
4.6 CONCLUSIONES.....	55
BIBLIOGRAFIA.....	56

TABLA DE CONTENIDOS DE FIGURAS.

Fig 1. La definición y la tipología de agente	8
Fig 2. Metodologías para SMA.....	16
Fig 3. Creación de predicados.....	27
Fig 4. Creación de campo de percepción del agente.	28
Fig 5. Creacion de reglas en predicados.....	29
Fig 6. Creacion de un plan de intención.....	30
Fig 7. Instantánea del modelo minero de oro: los triángulos amarillos son las pepitas de oro, el cuadrado negro es la base, y los círculos son los mineros. El círculo vacío alrededor de cada minero es su rango de percepción. (Figura de color en línea).....	31
Fig 8. Código GAML de las especies y variables mineras	32
Fig 9. Código GAML para la percepción y las reglas del minero.....	33
Fig 10. Código GAML de la letsW ander y obtener planes mineros de oro	34
Fig 11. Código GAML del objetivo elegir oro y volver a los planes mineros base	34
Fig 12. Muestra de agentes sanos e infectados en la simulación.	42
Fig 13. Variables y atributos.....	43
Fig 14. Creación de agentes de especies específicas	44
Fig 15. Creación de experimento	44
Fig 16. Variables para llevar el conteo de personas contagiadas y tiempo transcurrido.....	45
Fig 17. Interfaz de monitoreo de personas infectadas y horas transcurridas.....	45
Fig 18. Código GAML para detener la simulación.	45
Fig 19. Código GAML para crear monitor de resultados de la simulación.....	46
Fig 20. Código GAML para representación de gráficos.....	46
Fig 21. Código GAML para cargar los archivos SIG	47
Fig 22. Edificios y carreteras cargadas en la simulación.	47
Fig 23. Código para movimiento de agentes sobre carreteras y edificios.....	48
Fig 24. Código para saber la ubicación y tiempo que se encuentra dentro del edificio.	49
Fig 25. Agentes en edificios y carreteras.	49
Fig 26. Simulacion de accidente tecnológico en el centro urbano.....	50
Fig 27. Archivos .shp y .png incluidos en la simulación	50
Fig 28. Archivos .shp cargados en el modelo con la extensión file.....	51
Fig 29. Creación de agentes de datos para obtención de forma del modelo.	51
Fig 30. Inicializando simulación creando 200 agentes para realizar el objetivo	51
Fig 31. Otorgándole atributos a los agentes de la simulación.	52
Fig 32. Creación de creencias y deseos de un modelo BDI.....	52
Fig 33. Creación de emociones y percepciones de los agentes en la simulación.....	53
Fig 34. Creación de reglas para resguardo de agentes.	53
Fig 35. Creación de aspecto de agentes y zonas de la catástrofe.	54
Fig 36. Creación del experimento para obtener resultados de la simulación.	54

CONTENIDO DE TABLAS

Tabla 1: características de la plataforma GAMA.....	20
Tabla 2: Visión general de la usabilidad.....	21
Tabla 3: Capacidad de funcionamiento de cada plataforma de agente.	22
Tabla 4: Visión general de la pragmática.	23
Tabla 5: Visión general del lenguaje de programación	23
Tabla 6: Resumen del cumplimiento de fipa	25
Tabla 7: Modelo de roles.....	36
Tabla 8: Definición del protocolo Avisar	38
Tabla 9: Modelo de servicios.	39
Tabla 10: Modelo de roles.....	40
Tabla 11: Modelo de roles.....	41

1.1 PLANTEAMIENTO DEL PROBLEMA

A lo largo de la historia ha sido una inquietud permanente del hombre el querer conocer su futuro y el de sus territorios en la búsqueda de mejores condiciones de vida. El futuro se asume como un espacio de incertidumbre, por lo que se ha hecho necesario recurrir a la planeación estratégica considerando las pautas o patrones de crecimiento, esta disciplina ha cobrado gran importancia debido a la dinámica de la población mundial y a la necesidad cada vez mayor de ordenar el desarrollo urbano y los usos de suelo en equilibrio con el medio ambiente natural que garantice escenarios de sustentabilidad. En este sentido, se ha observado que la planificación tradicional es esencialmente retrospectiva, puesto que analiza fundamentalmente lo que sucede en el presente y ha sucedido en el pasado y bajo este enfoque se exploran posibilidades en la toma de decisiones, sin embargo, esta perspectiva considerando el acelerado desarrollo de la humanidad no permite anticiparse al control integral de situaciones imprevistas.

Uno de los campos de aplicación de los modelos de simulación en el marco del pensamiento prospectivo se refiere a la creación y análisis de escenarios territoriales, entendidos como la descripción de una situación futura para un territorio y la trayectoria de sus eventos que permiten llegar a ella a partir de su situación inicial, para el análisis de estos escenarios es imprescindible considerar los procesos urbanos en equilibrio con las dinámicas urbanas, es decir las condiciones ambientales, culturales y socioeconómicas en las que se circunscribe ese escenario, existen para ello un conjunto de modelos de simulación que permiten analizar

cómo se comporta un sistema conectado a las características de sus componentes individuales y a partir de ello interferir en el proceso de toma de decisiones, estableciendo de esta manera una vinculación entre los cambios en el pasado, la observación de lo presente y la proyección hacia el futuro.

Recientemente se han estado utilizando herramientas matemático-computacionales para modelar fenómenos urbanos, particularmente el desarrollo integrando los elementos espaciotemporales. De los más utilizados son los modelos basados en Autómatas Celulares (AC) R. Rodríguez (2002), aunque también se pueden encontrar modelos de regresión, redes neuronales artificiales (RNA), técnicas de EMC Gómez y J. Barredos (2006) y aún de manera incipiente modelos basados en agentes (MBA) C. Cantergiani, M. Gomez Delgado y C. Vergara (2014).

En esta tesis se plantea el uso de la simulación haciendo uso de agentes para aplicarse en el abordaje de problemas que tiene que ver con aspectos relacionados a la geomática como un área emergente en el estudio de fenómenos que surgen a partir de variables geoespaciales presentes en la dinámica de sistemas naturales y artificiales.

1.2 OBJETIVO

1. Analizar sistemas multiagente aplicables a la geomática utilizando metodologías estándar para su implementación.

1.3 OBJETIVOS ESPECÍFICO

1. Analizar metodología BDI para definir un modelo basado en agentes.
2. Analizar la metodología GAIA para definir el modelo de simulación multiagente.
3. Realizar experimentos de simulación multiagente aplicando metodologías BDI y GAIA

1.4 JUSTIFICACION

Entre una de las características de los Modelos Basados en Agentes más importantes está su capacidad de integrar los actores involucrados a los cambios territoriales, en ello reside su esencia al diferenciarlo de otros modelos más estáticos que se limitan al diseño de escenarios en base a componentes fijos, ello le permite mayor flexibilidad en sus interpretaciones. Carvalho and Colegios(2011) presentan una revisión teórica de los Modelos Basados en Agentes aplicados a simulaciones urbanas, exponiendo su base teórica, componentes, estructura, aplicaciones, así como sus principales potencialidades y deficiencias, destacan en su análisis que los Modelos basados en Agentes constituyen un modelo innovador y prometedor respecto a otros modelos aplicados a simulación de la expansión urbana, debido principalmente a la potencialidad que presenta para incorporar aspectos del comportamiento individual de los agentes en fenómenos emergentes que generan un patrón espacial agregado.

Es importante analizar diversas metodologías para llevar acabo simulaciones multiagente y poder aplicarlas al estudio de problemas en el área de geomática debido a que esto permite realizar estudios a partir de experimentos a escalas -temporales a través de las cuales se pueden hacer análisis en prospectiva y retrospectiva que resultan útiles para investigadores, planificadores, tomadores de decisiones, etc.

2.1 AGENTES

Las obras reconocidas como precursores en el campo de la investigación de los agentes y el SMA son: i) el modelo de agente para la resolución de problemas mediante el envío de mensajes asíncronos (Hewitt, 1977),) el sistema de blackboard para el intercambio de información entre los agentes (Hayes-Roth, Erman, Lesser, & Reddy, 1980);) el protocolo de contrato neto para la asignación de tareas (Smith, 1980) y,) el Vehículo de control distribuido de banco de pruebas para evaluar alternativas de diseño de las redes de distribución para resolver problemas (Lesser & Corkill, 1983). Actualmente no existe un consenso en la literatura científica sobre la definición de un agente. En este sentido, la definición de Jennings N.R. es usada comúnmente dentro de la comunidad del SMA, la cual expresa que: “un agente es un sistema informático, situado en un ambiente que es capaz de tener acción autónoma flexible para hacer frente a sus Objetivos...” (Jennings, Sycara, & Wooldridge, 1998). Según Wooldridge M. también se puede definir el concepto de un agente de acuerdo a las siguientes propiedades:

- Autonomía: un agente opera sin la intervención y acción directa de humanos o de otros (selección de tareas, toma de decisiones, etc.).
- Reacción: un agente percibe su entorno y reacciona de una manera adecuada a los cambios de este.
- Proactividad: Los agentes son capaces de actuar mediante la adopción de iniciativas impulsadas por sus objetivos.
- Habilidad social: Los agentes son capaces de interactuar con otros agentes a través de lenguajes de comunicación o reglas sociales. La importancia concedida a las características

expresadas anteriormente depende de la aplicación y necesidades que se tengan (Wooldridge & Jennings, 1995). Los agentes tienen la capacidad de percibir la modificación de su entorno y realizar acción sobre este. Entre las posibles acciones, los agentes han de determinar las decisiones más adecuadas que pueden alcanzar sus objetivos. Además del dominio de aplicación, el medio ambiente, la interacción y la organización influenciada por el diseño del agente. Los tres principales enfoques tradicionales definidos como metodología general utilizada para construir los agentes son: las arquitecturas de deliberación o cognitivas, las arquitecturas reactivas y arquitecturas híbridas.

2.1.1 TIPOLOGIA

Agentes deliberativos o cognitivos: Son una representación explícita de sus ambientes y una representación explícita de otros agentes. Acorde a su arquitectura, estos agentes son capaces de tomar decisiones, colaborando y tomando a 33 internos. Las decisiones son hechas por la vía del razonamiento lógico y basadas sobre la percepción que tienen del medio que los rodea. Un modelo particular para construir un agente racional según RAO es el modelo de deseo-creencia-intención (Belief-Desire-Intention Model, BDI) (Rao & Georgeff, 1991).

- Las creencias representan los estados de motivación del agente sobre sí mismo y de otros agentes.
- Los deseos representan los estados de motivación del agente de acuerdo a los objetivos o las situaciones que le gustaría lograr.
- Las Intenciones representan el estado de deliberación a la que el agente optó para hacer un compromiso. Según la información percibida por el medio ambiente, los agentes revisan las creencias, definen las diversas opciones (de sus deseos y sus intenciones actuales) y los filtros

de las posibles opciones y actos basados en las intenciones asociadas. Numerosas implementaciones de arquitecturas para construir agentes BDI son presentadas en la literatura como: IRMA (Intelligence resource - Bounded Machine architecture) y otros trabajos más como PRS (Procedural reasoning system), etc. A fin de crear agentes deliberativos o cognitivos, Shoham Propuso un paradigma de programación orientada a agentes (Agent Oriented Programming, AOP) (shoham, 1993), que declina en tres componentes: i) un lenguaje formal con una sintaxis y una semántica para definir el estado mental de los agentes, ii) un lenguaje de programación interpretado por los agentes de programación, iii) un proceso de identificación del agente etiológico para traducir el programa de agente en un sistema de archivo ejecutable. Cada agente se especifica en términos de: i) un 34 conjunto de capacidades, ii) un conjunto de creencias iniciales, iii) una serie de compromisos, y iv) un conjunto de reglas para estos compromisos.

Agentes reactivos: Estos agentes reaccionan a una base de estímulo-respuesta. Su comportamiento es dirigido por una serie de normas en respuesta a los estímulos del medio ambiente. Estos agentes mantienen un modelo interno de su entorno. Ellos no tienen objetivos explícitos y suelen reducir el uso de comunicación y protocolos. Los agentes reactivos no poseen un comportamiento individual inteligente, pero si poseen un comportamiento inteligente que es desprendido de sus interacciones. Los primeros trabajos realizados acorde a este enfoque están relacionados con la arquitectura de Subsunción. Según esta arquitectura un agente tiene un conjunto de comportamientos que se organizan en una jerarquía de acuerdo a su complejidad. Un comportamiento puede inhibir cualquier acción de un comportamiento de nivel inferior. La metáfora de las sociedades de insectos es ampliamente utilizada para definir sistemas de agentes reactivos. Dentro de estas sociedades de las

actividades de los individuos se coordinan, sin exhibir la comunicación o el razonamiento complejo.

Agentes Híbridos: (del griego ethos "costumbre") es la rama de la biología y de la psicología experimental que estudia el comportamiento de los animales en libertad o en condiciones de laboratorio. Este tipo de agentes combina los dos enfoques de agentes, tanto el cognitivo como el reactivo. Estas arquitecturas para construir este tipo de agentes combinan los dos enfoques anteriormente mencionados y son basadas sobre una descomposición de “capas” del comportamiento para obtener una gestión más eficiente de los recursos de los agentes. El enfoque reactivo se utiliza para la realización de las actividades de reflejo en respuesta a los estímulos del medio ambiente, estas actividades no requieren un razonamiento complejo. El enfoque cognitivo se utiliza para la realización de actividades basadas en la planificación y la deliberación que requieren razonamiento complejo. La arquitectura de IntgeRRaP (integración de la conducta reactiva y la planificación racional) (Muller, 1998) es una arquitectura de capas: La primera capa por encima del sistema de interfaz, es “la capa basada en el comportamiento” dedicada a la ejecución y control de la capacidad reactiva del agente. La segunda capa, que se encuentra por encima de la primera capa, es “la capa base” la cual se dedica a la generación de planes dirigidos por los objetivos del agente. La tercera capa, “la capa de cooperación”, se refiere a la planificación de cooperación para el modelado y la gestión de las interacciones con los otros agentes (por ejemplo: estrategias de negociación). Las capas de reacción, planificación y de modelado se encuentran entre la percepción y la acción. Las arquitecturas híbridas resultan ser más eficaces para estimar la capacidad de las decisiones y acciones de los agentes durante la fase de concepción de los sistemas multiagente.

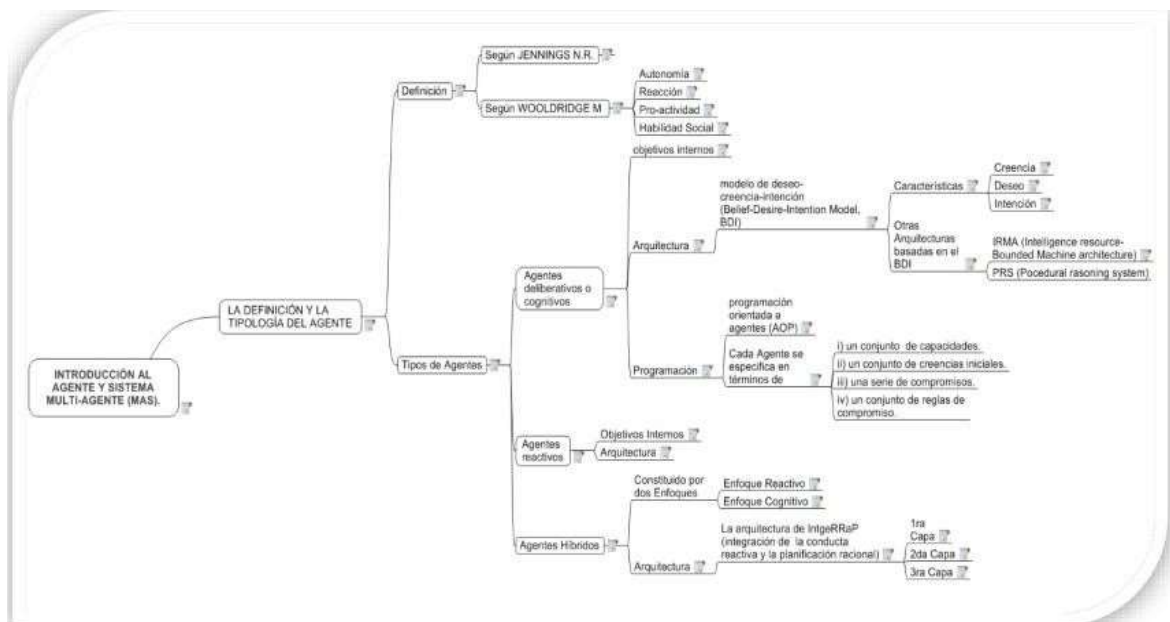


Fig 1. La definición y la tipología de agente

El sistema multiagente (S.M.A.) Sea el que sea el dominio de aplicación o el grado de inteligencia de los agentes, un sistema multiagente requiere la definición de diferentes elementos. Según (Ferber, 1999), un sistema multiagente es un sistema constituido por los siguientes elementos.

- Un medio (Enviroment) E, en general, un espacio con una medida
 - Un conjunto de objetos O, los objetos son posicionados y para cada objeto es posible determinar una posición para su entorno. Los objetos pueden ser percibidos, creados, destruidos y modificados por los agentes.
 - Un conjunto de agentes A, Los agentes son un subconjunto de Objetos (ACO) representando la actividad entre las entidades del sistema.
 - Un conjunto de relaciones o conexiones (Relations) R, Las cuales conectan objetos y también agentes a otros agentes.
 - Un conjunto de Operaciones Óp., propiciando la posibilidad de que los agentes perciban creen, manipulen y destruyan Objetos.
 - Un conjunto de operadores con la tarea de representar la aplicación de las operaciones y las reacciones del mundo para este intento de cambios, el cual ha sido llamado leyes del universo.
- Para desarrollar un sistema multiagente es necesario que haya agentes en diferentes lugares del ambiente y obligatoriamente que interactúen entre sí. Varias dimensiones son identificadas para el análisis y el desarrollo de un SMA: el agente, su entorno, las

interacciones y la organización. La preocupación del modelo de agente más concretamente es la descripción interna de los agentes (arquitecturas, conocimiento, representación, etc.).

Los modelos del medio ambiente corresponden a los campos en los que los agentes 38 deben evolucionar (por ejemplo: representación espacial). La preocupación de los modelos de interacción de la elaboración de un conjunto de normas en adecuación con la granularidad de un agente (Jerarquías, autoorganización, etc.). Para describir una organización modelo AGR (Agente/grupo/rol) se consideran tres conceptos primitivos: un agente es una entidad activa y comunicativa que juega roles entre grupos. Un grupo está compuesto por un grupo de agentes y delimita estructuras organizacionales. El rol es la representación abstracta de la posición funcional de una agente en un grupo. El SMA son sistemas en los cuales demasiadas entidades se desarrollan en un ambiente común. Esta convivencia es realizada acorde a un enfoque cooperativo o competitivo (delegación de tareas, logro de metas comunes y recursos compartidos, etc.). La relación entre agentes puede producir diversas formas de interacción (colaboración, conflicto, competición, negociación, coordinación, dependencia, etc.).

La interacción entre agentes puede ser dirigida por las metas u objetivos compatibles o incompatibles, Los recursos (conflictivos o no) o la capacidad de los agentes (suficiente o insuficiente). Los agentes pueden interactuar incluso directamente de agente a agente o indirectamente por modificación del ambiente.

La comunicación por envío de mensajes hace posible el desarrollo de varios tipos de relación en varios contextos de conflicto o no: desde la coordinación de negociación pasando por la colaboración o la cooperación. Hay dos modos de comunicación diferenciados en el SMA: comunicación indirecta por medio del ambiente y comunicación directa. La comunicación indirecta es por medio de señales puestas en el ambiente que son detectadas e interpretadas

por los agentes. La comunicación permite a los agentes intercambiar datos, información y conocimiento, por eso pueden coordinarse para lograr sus actividades. Este modo de interacción constituye uno de los aspectos fundamentales en la definición de la propiedad de sociabilidad. Una de las características sociales de un agente es su capacidad de interactuar con los otros agentes, el envío, la recepción e interpretación de mensajes requiere el lenguaje de comunicación, estos lenguajes apoyan la comunicación entre los agentes. Hay dos lenguajes principales para la comunicación.

- KQML (Knowledge Query Manipulation Language): desarrollado en 1993 por el consorcio DARPA-KSE
- ACL (Agent Communication language): Propuesto en 1997 por la FIPA Uribe Vásquez, J.

2.1.2 MODELOS BASADOS EN AGENTES

Un modelo basado en agentes (MBA) es un tipo de modelo computacional que permite la simulación de acciones e interacciones de individuos autónomos dentro de un entorno, y permite determinar qué efectos producen en el conjunto del sistema. Combina elementos de teoría de juegos, sistemas complejos, emergencia, sociología computacional, sistemas multiagente, y programación evolutiva. Los modelos simulan las operaciones simultáneas de entidades múltiples (agentes) en un intento de recrear y predecir las acciones de fenómenos complejos. Es un proceso de emergencia desde el nivel más elemental (*micro*) al más elevado (*macro*).

Generalmente los agentes individuales actúan según lo que perciben como sus intereses propios, tales como reproducción, beneficio económico o status social, y su conocimiento es limitado. Los agentes MBA pueden experimentar “aprendizaje”, adaptación y reproducción.

2.2 SISTEMA MULTIAGENTE

Un sistema multiagente es un sistema distribuido en el cual los nodos o elementos son sistemas de inteligencia artificial, o bien un sistema distribuido donde la conducta combinada de dichos elementos produce un resultado en conjunto inteligente.

Hay que notar que los agentes no son necesariamente inteligentes. Existen como en todo el resto del dominio de la inteligencia artificial, dos enfoques para construir sistemas multiagente:

- El enfoque formal o clásico, que consiste en dotar los agentes de la mayor inteligencia posible utilizando descripciones formales del problema que resolver y de hacer reposar el funcionamiento del sistema en tales capacidades cognitivas. Usualmente la inteligencia es definida utilizando un sistema formal (por ejemplo, sistemas de inferencia lógica) para la descripción, raciocinio, inferencia de nuevo conocimiento y planificación de acciones a realizar en el medio ambiente.
- El enfoque constructivista, que persigue la idea de brindarle inteligencia al conjunto de todos los agentes, para que, a través de mecanismos ingeniosamente elaborados de interacción, el sistema mismo genere comportamiento inteligente que no necesariamente estaba planeado desde un principio o definido dentro de los agentes mismos (que pueden ser realmente simples). Este tipo de conducta es habitualmente llamado comportamiento emergente. Uribe Vásquez, J. A. (2015).

2.2.1 SIMULACION MULTIAGENTE

Por un lado, los MAS se refieren a un dominio de investigación en informática que se ocupa de sistemas que están compuestos a nivel micro por entidades (agentes) que tienen un comportamiento autónomo y proactivo e interactúan con el medio ambiente, resultando en su funcionamiento en un comportamiento del sistema global que se observa a nivel macro. Actualmente se utilizan en numerosos dominios de aplicación e investigación, y pueden ser una forma muy interesante y conveniente para la comprensión, modelado, diseño e implementación de diferentes tipos de sistemas distribuidos. Por ejemplo, pueden ser utilizados como un paradigma de programación para desarrollar sistemas de software distribuidos que se ejecutan en contextos en los que un control global es difícil o imposible de lograr. Y también representan una alternativa muy interesante de modelado, en comparación con el modelado basado en ecuaciones, para la representación y simulación de sistemas del mundo real o virtual que pueden ser descompuestos en unidades menores que interactúan.

Por otro lado, la simulación por computadora es una forma de diseñar, probar y estudiar tanto teorías como sistemas reales para diversos fines. Por ejemplo, de acuerdo a Shoham, (1983) la simulación se define como: *"El proceso de diseñar un modelo de un sistema real junto con la realización de experimentos con este modelo con el fin de mejorar la comprensión del comportamiento del sistema y/o de la evaluación de diferentes estrategias para el funcionamiento del sistema"*.

Por lo tanto, las oportunidades de utilizar ambos, MAS y simulación, son numerosas, precisamente porque se pueden aplicar conjuntamente en una amplia gama de dominios de

aplicación, y para fines muy diferentes. De hecho, el número de trabajos de investigación y aplicaciones de software que pertenecen a la intersección entre el MAS y de simulación es enorme. Esta intersección la analizaremos según dos perspectivas principales:

1. **Modelado y Simulación (M&S) para el MAS:** se refiere a los proyectos en los que la simulación por ordenador se utiliza como un medio para el diseño, la experimentación, el estudio y/o la ejecución de arquitectura de MAS, sean cuales sean los objetivos.
2. **MAS para M&S:** se relaciona con experimentos de simulación que utilizan MAS como paradigma de modelado para construir laboratorios artificiales. Hoy en día, el uso de laboratorios artificiales representa una forma de poner a prueba teorías y modelos para muchos campos de aplicación. Uribe Vásquez, J. A. (2015).

2.2.2 LAS METODOLOGIAS

La metodología es una pieza esencial de toda investigación (método científico) que sigue a la propedéutica ya que permite sistematizar los procedimientos y técnicas que se requieren para concretar el desafío. Cabe aclarar que la propedéutica da nombre a la acumulación de conocimientos y disciplinas que son necesarios para **abordar y entender** cualquier materia. El término proviene del griego *pró* (“**antes**”) y *paideutikós* (“**referente a la enseñanza**”)

En otras palabras, la metodología es un recurso concreto que deriva de una **posición teórica y epistemológica**, para la selección de técnicas específicas de investigación. La metodología, entonces, depende de los postulados que el investigador crea que son válidos, ya que la acción metodológica será su herramienta para analizar la realidad estudiada. La metodología para ser eficiente debe ser disciplinada y sistemática y permitir un enfoque que permite analizar un problema en su totalidad.

Dentro de una investigación pueden desarrollarse muchas metodologías, pero todas ellas pueden encasillarse en dos grandes grupos, la metodología de investigación **cualitativa y cuantitativa**. La primera es la que permite acceder a la información a través de la recolección de datos sobre variables, llegando a determinadas conclusiones al comparar estadísticas; la segunda, realiza registros narrativos sobre fenómenos investigados, dejando a un lado la cuantificación de datos y obteniéndolos a través de entrevistas o técnicas no-numéricas, estudiando la relación entre las variables que se obtuvieron a partir de la observación, teniendo en cuenta por sobre todo los contextos y las situaciones que giran en torno al problema estudiado.

Por otro lado, la metodología también puede ser comparativa (analiza), descriptiva (expone) o normativa (valora). Para saber si conviene utilizar un tipo de metodología u otro, el científico o investigador tiene que tener en cuenta un conjunto de aspectos importantes. Algunas de las preguntas que debe hacerse son: ¿qué resultados espera conseguir? ¿quiénes son los interesados en conocer los resultados? ¿cuál es la naturaleza del proyecto?

Es imprescindible que el método empleado y la teoría, que ofrece el marco donde se insertan los conocimientos, estén unidos por la coherencia (el cómo y el qué deben ser coherentes entre sí); esto significa que la metodología debe ser utilizada dentro de un **marco ideológico**, un sistema de ideas coherentes que sean las encargadas de explicar el para qué de la investigación.

Como ya lo hemos explicado, método y metodología son cosas diferentes. El término método, también conocido como **técnicas de investigación**, puede definirse como el camino para alcanzar a un fin; en relación con la metodología consiste en los procedimientos que deben llevarse a cabo para cumplir con lo estipulado por ella y obtener conclusiones verídicas

sobre el fenómeno o problema que se analiza. En otras palabras, mientras que la metodología es lo que une al sujeto al objeto de conocimiento y es imprescindible para conseguir el conocimiento científico, el método es el camino o instrumento que nos lleva a él.

2.3 METODOLOGIAS PARA SMA

- 1 AAI (Australian Artificial Intelligence Institute): Basada en el modelo BDI
- 2 Gaia: SMA como conjunto de entidades que interactúan
- 3 MaSE: OO con conversaciones entre objetos
- 4 Tropos: Énfasis en la gestión de requisitos
- 5 Zeus: Entorno visual de desarrollo de agentes y Prototipos de agentes
- 6 MAS-CommonKADS: CommonKADS extendido con OO, SDL y MSC
- 7 MESSAGE, INGENIAS: Meta-modelado, agentes que siguen el principio de racionalidad de Newell



Fig 2. Metodologías para SMA

Fuente: <http://www.fdi.ucm.es/profesor/jpavon/doctorado/desarrolloSMA.pdf>

2.3.1 PASOS ANTES DE CONSTRUIR UNA SMA

Para construir una SMA habrá que tomar en consideración lo siguiente:

- Los resultados que va a producir
- Documentación.
- Código, prototipos, pruebas.
- Visuales: data Flow diagrams, entity-Relationship diagrams message sequence charts, UML
- Formales: Z, redes Petri
- Qué actividades para producir los resultados
- De análisis, diseño, implementación, validación, entre otros
- Con que herramienta
- Desarrollo
- Entorno de ejecución.

Para esto existen varios tipos de metodologías y hay que elegir la que se adapta a las intenciones y resultados esperados. (Pavon,2004)

2.3.2 METODOLOGIA GAIA

Es una metodología para el diseño de sistemas basados en agentes cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global. GAIA pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que, según los autores, esté lo suficientemente detallado como para ser implementado directamente. En GAIA se entiende que el objetivo del análisis es conseguir comprender el sistema y su estructura sin

referenciar ningún aspecto de implementación. Esto se consigue a través de la idea de organización. Una organización en GAIA es una colección de roles, los cuales mantienen ciertas relaciones con otros y toman parte en patrones institucionalizados de interacción con otros roles. Los roles agrupan cuatro aspectos: responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas e interacciones. GAIA propone trabajar inicialmente con un análisis a alto nivel. En este análisis se usan dos modelos, el modelo de roles para identificar los roles clave en el sistema junto con sus propiedades definitorias y el modelo de interacciones que define las interacciones mediante una referencia a un modelo institucionalizado de intercambio de mensajes, como el FIPA-Request [FIPA 03]. Tras esta etapa, se entraría en lo que GAIA considera diseño a alto nivel. El objetivo de este diseño es generar tres modelos: el modelo de agentes que define los tipos de agente que existen, cuántas instancias de cada tipo y qué papeles juega cada agente, el modelo de servicios que identifica los servicios (funciones del agente) asociados a cada rol, y un Modelo de conocidos, que define los enlaces de comunicaciones que existen entre los agentes. A partir de aquí, los autores de GAIA proponen aplicar técnicas clásicas de diseño orientado a objetos. Sin embargo, GAIA declara que queda fuera de su ámbito. Esta metodología sólo busca especificar cómo una sociedad de agentes colabora para alcanzar los objetivos del sistema, y qué se requiere de cada uno para lograr esto último. Sanz, J. J. G. (2003).

2.3.3 VENTAJAS Y DESVENTAJAS DE LA METODOLOGIA GAIA

➤ VENTAJAS

- En el análisis añade:
- Modelo del entorno: representación computacional abstracta del entorno del SMA, como una lista de recursos abstractos.
- Reglas organizacionales: restricciones para las actividades de los roles.
- En el diseño considera la definición de la estructura organizacional.
- Definida a partir de un conjunto de patrones organizacionales.

➤ DESVENTAJAS

- Sigue sin considerar una implementación.

Sanz, J. J. G. (2003).

2.4 DESCRIPCION GENERAL DE LA PLATAFORMA DE AGENTES QUE SE UTILIZO.

GAMA (Grignard et al. 2013) es una plataforma de simulación, cuyo objetivo es proporcionar a los expertos de campo, modeladores y científicos informáticos un entorno completo de modelado y desarrollo de simulación para la construcción de simulaciones multiagente espacialmente explícitas. Proporciona capacidades relativas a la estrecha combinación de visualización 3D, gestión de datos SIG y modelado multinivel. G A M A admite capacidades para la construcción de modelos grandes escritos en el lenguaje orientado al agente GAML, con una herramienta de modelado gráfico opcional para admitir el diseño rápido y la creación de prototipos. Los usuarios también pueden crear instancias de agentes a partir de datos SIG, bases de datos o archivos y ejecutar simulaciones a gran escala (hasta

millones de agentes) y diseñar interfaces de usuario ricas que admiten inspecciones profundas en agentes, acciones y paneles controlados por el usuario, y múltiples pantallas y aspectos 2D/3D multicapa.

2.4.1 PROPIEDADES DE LA PLATAFORMA

Tabla 1: características de la plataforma GAMA

Nombre	Desarrollador/Organización	Dominio principal	Licencia de última	Versión	Código abierto
GAMA	Unidad Internacional de Investigación IRD/UPMC UMMISCO	Simulaciones basadas en agentes espacialmente explícitas distribuidas a gran escala	RANGO 1.6.1 (06/10/2013)	GPL	Si

2.4.2 VISION GENERAL DE LA USABILIDAD

Tabla 2: Visión general de la usabilidad

Nombre	Simplicidad	Aprendabilidad	Escalabilidad	Compatibilidad estándar	Comunicación
GAMA	Simple/Algunos útiles para las funciones de simulación de GUI	Fácil	Buena	(GIS, capacidades 3D)	ACL (mensajes GAML que representan los mensajes ACL FIPA)

2.4.3 CAPACIDAD DE FUNCIONAMIENTO DE CADA PLATAFORMA DE AGENTE.

Tabla 3: Capacidad de funcionamiento de cada plataforma de agente.

Nombre	Rendimiento	Estabilidad	Robustez	Idiomas de programación	Sistemas operativos
GAMA	Buena	Buena	Buena	GAML	Mac OS X, Windows, Linux

2.4.4 VISION GENERAL DE LA PRAGMATICA

Tabla 4: Visión general de la pragmática.

Nombre	Instalación	Soporte al usuario	Popularidad	Madurez	Costo
GAMA	Instalacion	Bueno (foro, correo electrónico, documentos, wiki, redes sociales)	Bajo	Liberación estable baja, estado de desarrollo (activo)	Gratis

2.4.5 VISION GENERAL DEL LENGUAJE DE PROGRAMACION

Tabla 5: Visión general del lenguaje de programación.

GAML

GAMA

2.4.6 CUMPLIMIENTO FIPA

La Fundación para Agentes Físicos Inteligentes (FIPA) es una organización internacional que se dedica a promover la industria de los agentes inteligentes mediante el desarrollo abierto de especificaciones que apoyan la interoperabilidad entre agentes y Aplicaciones. El cumplimiento de FIPA es un punto de referencia para muchos investigadores. Algunas plataformas son totalmente compatibles, mientras que otras adoptan solo algunas de las especificaciones disponibles.

El cumplimiento de las especificaciones FIPA proporciona algunas ventajas específicas que podrían demostrarse útiles o incluso cruciales para algunos casos. Por ejemplo, ser compatible con FIPA garantiza la arquitectura y el rendimiento del sistema debido a los protocolos potentes y bien probados que permiten la cooperación y la interoperabilidad de los agentes. Por lo tanto, los estándares FIPA son necesarios siempre que se estudien casos dinámicos, flexibles y reconfigurables. Además, las especificaciones FIPA son populares en la investigación basada en agentes y, por lo tanto, hay un montón de conocimiento y apoyo en la comunidad de investigación

2.4.7 RESUMEN DEL CUMPLIMIENTO DE FIPA

Tabla 6: Resumen del cumplimiento de fipa.

Cumplimiento total	JADE, Jadex, JACK, EMERALD
Cumplimiento parcial	JAS, Jason, AGLOBE, Agent Factory, SeSAm, GAMA
Sin cumplimiento	Cougaar, Swarm, MASON, INGENIAS Development Kit, Cormas, Repast, MaDKit, CybelePro, JIAC, AgentScape, AnyLogic, NetLogo, JAMES I

3.1 METODOLOGIA BDI

Las arquitecturas BDI se han introducido en varias plataformas de modelado y simulación basadas en agentes (ABMS). Por ejemplo, han propuesto una extensión orientada a la educación a Netlogo para permitir a los modeladores manipular conceptos BDI en un lenguaje sencillo. Esta arquitectura muy simple está inspirada en PRS: los agentes tienen un conjunto de creencias (información obtenida por percepción o comunicación), un conjunto de intenciones (lo que quieren ejecutar) y formas de administrar estos dos conjuntos.

El modelador sólo es necesario para definir simple BDI como la arquitectura del agente, y para definir al menos un plan y un deseo inicial de hacerlo operativo. A continuación, el modelador define principalmente los planes (lo que el agente puede hacer), los deseos (lo que el agente quiere) y (normalmente una) percepción (s) (cómo el agente percibe su entorno). Muchas palabras clave y funciones se definen para ayudar al usuario a actualizar y administrar bases de creencias, deseos e intenciones, y crear y administrar predicados.

Las creencias, deseos e intenciones se describen mediante predicados. Un predicado tiene un nombre, también puede tener valores (un conjunto de pares nombre-valor) y puede ser true (de forma predeterminada) o false. Los predicados se pueden definir fácilmente a través del lenguaje GAML. Por ejemplo, en Fig. 3, definimos una nueva ubicación predicado de oro, que representa el hecho de que una pepita de oro está presente en la ubicación (20, 50). Tenga en cuenta que, al comparar dos predicados, si uno de ellos no tiene ningún valor para una clave específica, no se tienen en cuenta los valores correspondientes a esta clave. Por ejemplo, la prueba de si el predicado ("ubicación de oro") está presente en la base de

creencias devolverá true si el predicado ("location gold", ["location value": 20, 50]) está en la base (o el mismo predicado con cualquier otro valor).

```
new_predicate("location_gold", ["location_value":: {20, 50}]);
```

Fig 3. Creación de predicados.

3.1.1 LOS AGENTES BDI TIENEN 3 BASES DE DATOS:

1. Base de creencias: (lo que sabe): el conocimiento interno que el agente tiene sobre el mundo o sobre su estado interno, actualizado durante la simulación. Una creencia puede referirse a cualquier tipo de información (una cantidad, una ubicación, un valor booleano, etc.) y se describe mediante un predicado. Por ejemplo, la ubicación del predicado dorado (Fig. 1)

se agrega como una creencia cuando el agente percibe una pepita de oro en la posición (20,50).

2. Base de deseo: (lo que quiere): objetivos que el agente desea a completar, también actualizado durante la simulación. Un deseo se cumple cuando está presente en la base de la Creencia (por ejemplo, si deseo que la pepita de oro en un cierto lugar ya no existe en el mundo, y también creo que esto es verdad, entonces mi deseo se cumple y se elimina de mi base de deseos) o cuando es eliminado por el agente. Los deseos pueden tener vínculos jerárquicos (sub/super deseos) cuando se crea un deseo

como un objetivo intermediario. Los deseos tienen un valor de prioridad dinámico utilizado para seleccionar una nueva intención entre los deseos cuando sea necesario.

3. Base de intención: (lo que está haciendo): lo que el agente ha elegido hacer. La intención actual determinará el plan seleccionado. Las intenciones pueden ponerse en espera (por ejemplo, cuando requieren que se logre una su intención). Para esta razón, hay una pila de intenciones: la superior es la intención actual, todas las demás están en espera.

Comportamiento: Además de las estructuras básicas de comportamiento disponibles para todos los agentes de GAMA, nuestra arquitectura simple BDI tiene tres tipos de estructuras

- Percepción: Una percepción es una función ejecutada en cada iteración para actualizar la base de creencias del agente, para observar los cambios en su entorno (el mundo, los otros agentes y sí mismo).

El agente puede percibir otros agentes hasta una distancia fija o dentro de una geometría específica. Por ejemplo, un minero puede percibir todas las pepitas de oro a una distancia de 10 m y para cada una de ellas añadir una nueva creencia con respecto a la ubicación del oro (véase la Fig. 4). Al igual que para muchas otras declaraciones gama, se pueden utilizar varias facetas (la mayoría de ellas opcionales) para ajustar las percepciones:

```
perceive target:gold in:10 {  
  focus var:location;  
}
```

Fig 4. Creación de campo de percepción del agente.

- objective (obligatorio): la lista de agentes (o especies de agentes) a percibir.
- when (opcional): condición de percepción - puede ser una expresión dinámica.
- in (opcional): puede ser un radio (flotante) o una geometría (por ejemplo, la definición de un cono de percepción o algo más complejo).

Regla: Una regla es una función ejecutada en cada iteración para inferir nuevos deseos o creencias de las creencias y deseos actuales del agente, es decir, un nuevo deseo o creencia pueden surgir de los existentes. Por ejemplo, cuando el minero tiene una creencia acerca de algunas pepitas de oro en un lugar específico, la regla crea el deseo de extraer oro (véase la Fig. 5).

```
rule belief: new_predicate("location_gold") new_desire: get_gold ;
```

Fig 5. Creacion de reglas en predicados

Por supuesto, si el nuevo deseo (o creencia) ya está en la base, no se añade nada. Las facetas de las reglas son las siguientes:

- creencia (opcional): creencia necesaria para activar la regla.
- deseo (opcional): deseo necesario de activar la regla.
- cuando (opcional): condición necesaria para activar la regla.
- nueva creencia (opcional): la nueva creencia que se añadirá a la base del agente.
- nuevo deseo (opcional): el nuevo deseo de ser añadido a la base del agente.

Plan: El agente tiene un conjunto de planes, que son comportamientos definidos para lograr intenciones específicas. Los planes pueden ser instantáneos y/o persistentes, y pueden tener un valor prioritario (que puede ser dinámico), utilizado para seleccionar un plan cuando hay varios planes posibles disponibles para lograr la misma intención. Por ejemplo, cuando el minero tiene la intención de find oro, puede activar su plan letsWander (ver Fig. 6) que hace que se mueva al azar.

```
plan letsWander intention:find_gold {  
  do wander;  
}
```

Fig 6. Creacion de un plan de intención

En cuanto a las facetas de los planes:

- intención (opcional): intención de que el plan intente cumplir.
- cuando (opcional): condición de activación.
- terminado cuando (opcional): condición

3.1.2 MODELO MINERO EN BUSQUEDA DE ORO

Para ilustrar nuestra arquitectura con un ejemplo más detallado, presentamos aquí una implementación del modelo minero en búsqueda de oro

En la Fig. 7 se muestra una instantánea del modelo. Definimos 3 especies de agentes: minero (los mineros), oro (las pepitas de oro) y base (la base donde los mineros almacenan sus pepitas de oro, que es única). Debido a las limitaciones de espacio, sólo detallamos aquí el comportamiento de los agentes mineros. Los agentes mineros utilizan la arquitectura simpleBDI.

Se definen 8 variables para esta especie (Fig. 7):

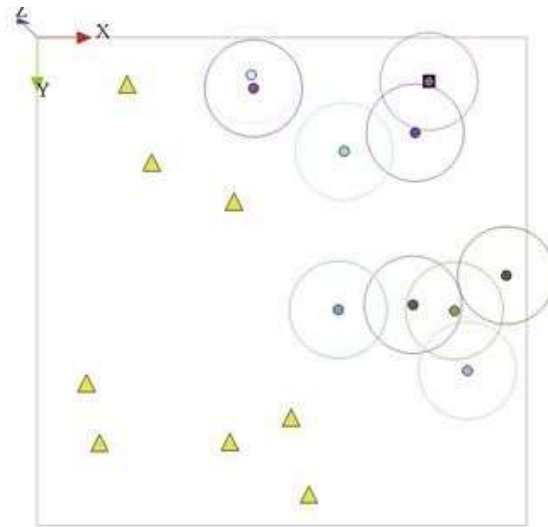


Fig 7. Instantánea del modelo minero de oro: los triángulos amarillos son las pepitas de oro, el cuadrado negro es la base, y los círculos son los mineros. El círculo vacío alrededor de cada minero es su rango de percepción. (Figura de color en línea)

- velocidad: la velocidad de movimiento del agente (3 m/s)
- mycolor: un color aleatorio que se utilizará para mostrar el agente
- objetivo: la ubicación donde el agente quiere ir (sólo se utiliza cuando el agente no vaga por).
- tiene oro: un predicado utilizado como la creencia que indica que el agente tiene una pepita de oro.
- definir el objetivo de oro: un predicado utilizado como el deseo de definir un nuevo objetivo (con un valor de prioridad de 3).
- obtener oro: un predicado utilizado como el deseo de tomar una pepita de oro (con un valor de prioridad de 2).
- wander: un predicado utilizado como el deseo de vagar.
- return base: un predicado utilizado como deseo de volver a la base (con un valor de prioridad de 3). La especie utiliza una percepción y dos reglas (Fig. 8):

- percepción: la percepción permite percibir las pepitas de oro en un radio de 20 m. Cada vez que se percibe una pepita de oro, se añade una nueva creencia sobre la ubicación de la pepita de oro a la base de creencias y la intención de vagar se elimina.
- regla: la primera regla añade un nuevo deseo de tomar una pepita de oro si el agente cree que hay una pepita de oro en alguna parte. La segunda regla añade un nuevo deseo de volver a la base si el agente cree que tiene una pepita de oro.

```
species miner skills: [moving] control:simple_bdi {
  float speed <- 3.0;
  rgb mycolor<-rnd_color(255);
  point target;

  predicate has_gold <- new_predicate("has_gold");
  predicate define_gold_target <- new_predicate("define_gold_target") with_priority 3;
  predicate get_gold <- new_predicate("get_gold") with_priority 2;
  predicate wander <- new_predicate("wander");
  predicate return_base <- new_predicate("return_base") with_priority 4;
```

Fig 8. Código GAML de las especies y variables mineras

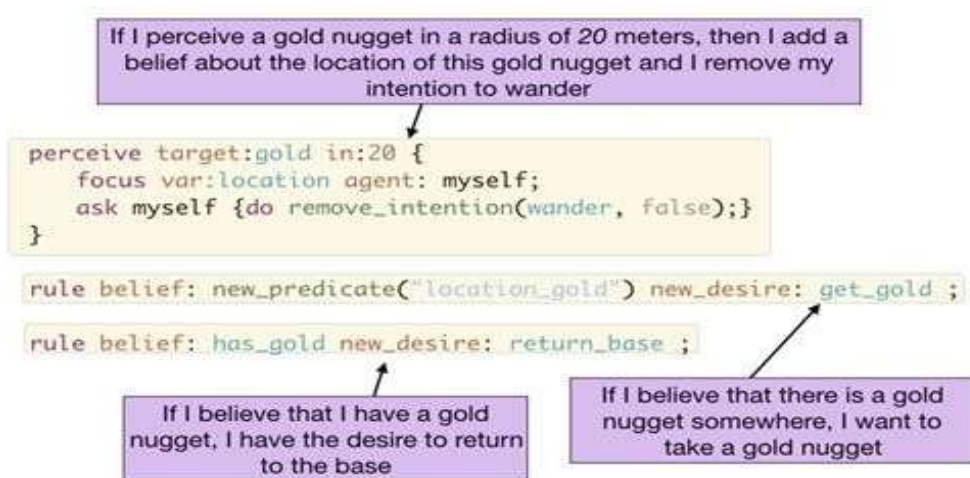


Fig 9. Código GAML para la percepción y las reglas del minero

La especie utiliza cuatro planos (Figs. 9 y 10):

- letsWander: para lograr la intención de vagar, el agente se mueve al azar.
- getGold: para lograr la intención de obtener oro, el agente se mueve hacia una de las pepitas de oro que conoce y tomarla.
- Elegir objetivo de oro: para lograr la intención del objetivo de oro def ine, el agente elige sus pepitas de oro más cercanas conocidas como objetivo.
- retorno a la base: para lograr la intención de la base de retorno, el agente vuelve a su base.

3.1.3 EXPERIMENTOS

Análisis de comentarios de los modeladores Contexto del experimento. Con el fin de validar nuestra afirmación de que nuestra arquitectura es fácil de usar y se adapta a varios tipos de usuarios, llevamos a cabo una.

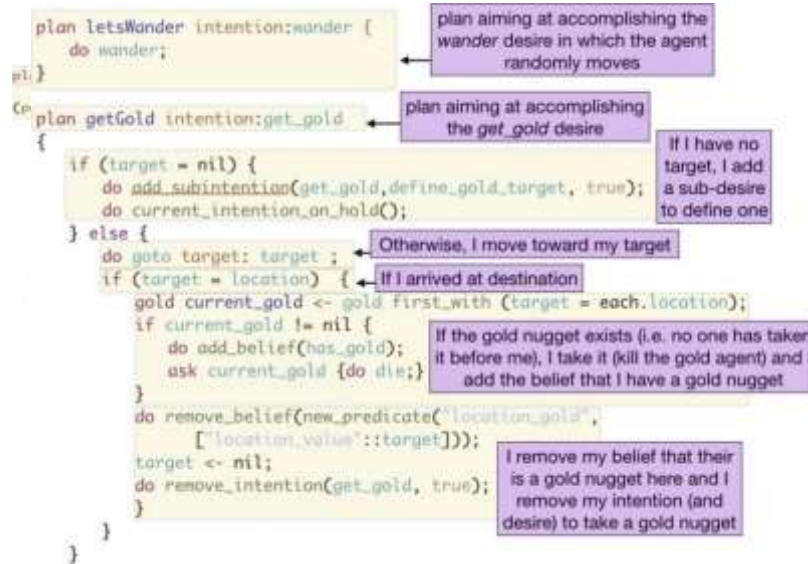


Fig 10. Código GAML de la letsWander y obtener planes mineros de oro

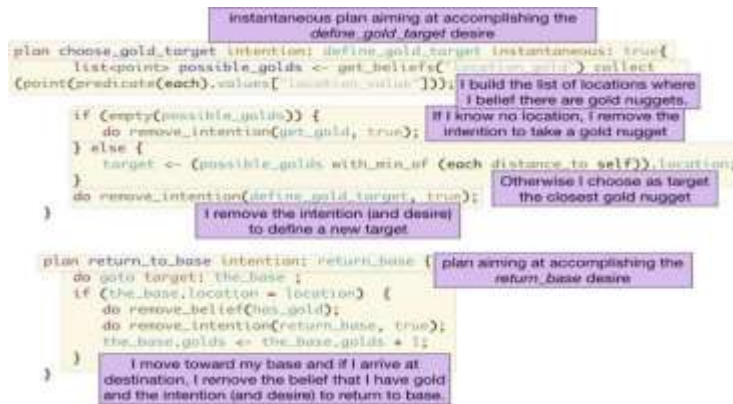


Fig 11. Código GAML del objetivo elegir oro y volver a los planes mineros base

3.2 METODOLOGIA GAIA

En GAIA se habla de organización que es una colección de roles, los cuales mantienen ciertas relaciones con otros e interacciones con otros roles. Los roles agrupan cuatro aspectos: responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas e interacciones.

GAIA propone trabajar inicialmente con un análisis a alto nivel donde se usan dos modelos, el modelo de roles para identificar los roles clave en el sistema junto con sus propiedades definitorias y el modelo de interacciones. Luego vendría el diseño a alto nivel. El objetivo de este diseño es generar tres modelos: el modelo de agentes(tipos), modelo de servicios(funciones), y un modelo de conocidos(enlaces).

Los principales conceptos que aparecen en la metodología se dividen en dos: abstractos y concretos.

Las entidades abstractas son aquellas que son empleadas durante el análisis para la conceptualización del sistema.

Las entidades concretas son empleadas en el proceso de diseño.

La entidad más abstracta de un sistema en la jerarquía de conceptos que se presenta es el sistema relacionado con la idea de “sociedad” u “organización”.

3.3 PROCESOS PARA HACER LOS MODELOS

3.3.1 EL PROCESO DE ANÁLISIS

- a) Identificar los roles del sistema
- b) Para cada papel identificar y documentar los protocolos asociados

Empleando el modelo de protocolo como base, elaborar con más detalle el modelo de roles.

3.3.2 PROCESO DE DISEÑO

- a) Creación de un modelo de agente, agregando roles a los tipos de agentes y documentando las instancias de cada tipo de agente.
- b) Desarrollo de un modelo de servicios, examinando protocolos y propiedades de viveza y seguridad.
- c) Desarrollo de un modelo de conocimiento, por medio del modelo de interacción y el modelo de agente

Tabla 7: Modelo de roles.

Modelo de Roles	
Esquema del Rol:	Minero
Descripción:	Este rol involucra al agente minero en buscar por el territorio asignado pepitas de oro, para lograr la extracción de oro y guardar esa pepita de oro en una base de almacenamiento.
Actividades:	Buscar: Consiste en que el agente buscara dentro del territorio pepitas de oro esparcidas sobre este. Extraer: Esta actividad implica en extraer un fragmento de oro o la mayor cantidad que pueda llevar dicho agente minero. Almacenar: Cuando el minero obtiene una pepita de oro, este tiene la obligación de llevarlo a la base de almacenamiento.
Protocolos:	Avisar: El agente minero dará información sobre la locación de pepitas de oro dentro del territorio.
Permisos Change supplied: Terreno	//Terreno donde se encuentra el minero.

Responsabilidades	
Liveness:	Minero=([Buscar]. [Extraer]. [Almacenar]. [Avisar].)
Safety:	Obtención >35

Tabla 8: Definición del protocolo Avisar.

Definición del protocolo Avisar
AVISAR
Cuando una pepita es encontrada en el terreno y los agentes en un diámetro de 0.1m se encuentran este le da información sobre la ubicación del oro que encontró.

Tabla 9: Modelo de servicios.

Modelo de Servicios				
Servicio	Entradas	Salidas	Precondiciones	Postcondiciones
Buscar	Locación	Actividad o protocolo	Camino=0	La condición es encontrar oro
Extraer	Pepitas de oro	El agente encuentra el oro	Valor oro >1	Tiene que extraer todo el oro
Tiene que extraer todo el oro	oro	El agente almacena el oro	Valor oro>1	Almacenar todo el oro

Tabla 10: Modelo de roles.

Modelo de roles	
Esquema del Rol:	Pepitas de oro
Descripción:	Será un agente estático con propios atributos, los cuales se irán descontando cada que un agente minero encuentre su ubicación y extraiga oro.
Actividades:	Aparecer: Esta actividad se dará al momento de inicializar la simulación se crearán pepitas de oro al azar en el territorio para que los agentes las extraigan.
Protocolos:	Disminuir: Al momento de que el agente encuentre su ubicación, la ubicación de oro disminuirá dependiendo la capacidad del agente minero para sustraer oro.
Permisos Change supplied: Terreno	//Terreno donde se encuentra el oro.
Responsabilidades	
Liveness:	Pepitas de oro = ([Aparecer].[disminuir].)
Safety:	Disminuir > Obtención

Tabla 11: Modelo de roles.

Modelo de roles	
Esquema del Rol:	Almacenamiento de oro
Descripción:	El agente almacenamiento de oro aparecerá al momento de arrancar la simulación colocando un agente estático dentro del territorio para almacenar el oro encontrado por los agentes mineros.
Actividades:	Aparecer2: Esta actividad se dará al momento de inicializar la simulación se creará un almacenamiento de oro al azar en el territorio para que los agentes resguarden el oro extraído de las pepitas.
Protocolos:	Contador: se tendrá un registro del oro almacenado
Permisos Change supplied: Terreno	//Terreno donde se encuentra el oro.
Responsabilidades	
Liveness:	Pepitas de oro = ([Aparecer2]. [avisar 2]. [Contador])
Safety:	Contador > obtención

SIMULACION DE INFECCION EN UN AREA DE LA CIUDAD DE ZONA CENTRO.

Descripción: Calcular tiempo en que tarda la infección en esparcirse por tiempo indeterminado o contagio total de personas.

- Paso de simulación: 1 minuto.
- La gente se mueve en los caminos de un edificio a otro.
- La gente usa el camino más corto para moverse entre edificios.
- Todas las personas tienen la misma velocidad y se mueven a una velocidad constante.
- Cada vez, las personas llegan a un edificio donde se quedan un tiempo determinado.
- El tiempo de permanencia depende de la hora actual (menor a las 9h - ir a trabajar - a las 12h ir a almorzar - a las 18h - volver a casa).
- Las personas infectadas nunca se curan.

4.1 MODELO SI SIMPLE

4.1.1 FORMULACIÓN

- Definición de la **people** especie con una variable (is_infected), un aspecto (base) y dos comportamientos (move y infect).
- Definición de nb_infected_init, distance_infection and parámetros proba_infection`.
- Creación de **500** people agentes ubicados aleatoriamente en el entorno (tamaño: 500x500).
- Definición de una pantalla para visualizar los agentes de personas.

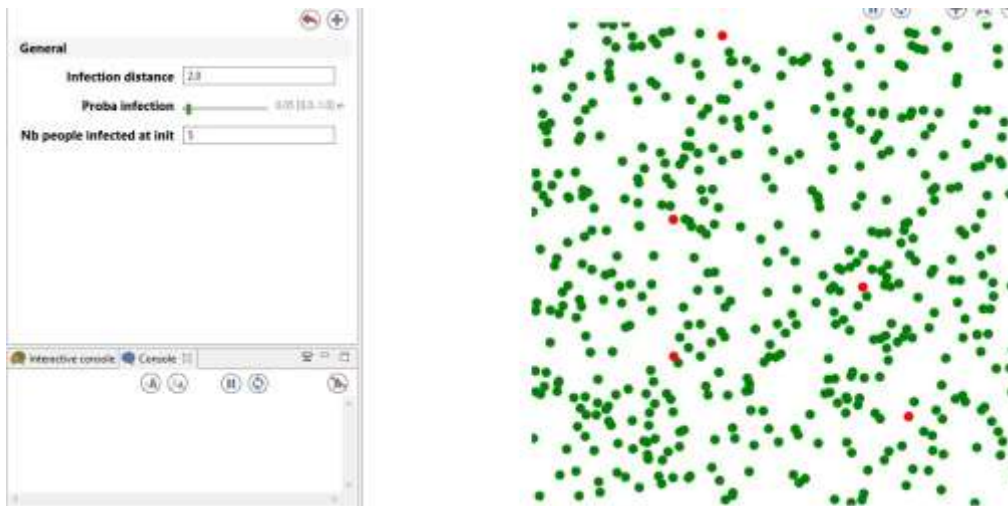


Fig 12. Muestra de agentes sanos e infectados en la simulación.

4.1.2 SECCIÓN GLOBAL

La global sección representa la definición de la especie de un agente específico (llamado world). El agente mundial representa todo lo que es global para el modelo: dinámica, variables ... Permite iniciar simulaciones (init bloque): el mundo siempre se crea e inicializa primero cuando se inicia una simulación. La geometría (forma) del agente mundial es por defecto un cuadrado con 100 m para el tamaño del lado, pero se puede redefinir si es necesario. Del mismo modo, el modelador puede redefinir el step variable que representa la duración de un paso de simulación y que, por defecto, es 1 segundo.

4.1.3 VARIABLES GLOBALES

Para nuestro modelo, definimos 5 variables globales: **nb_people** (int, valor inicial: 500), **infection_distance** (valor flotante, valor inicial: 2 metros), **proba_infection** (flotante, valor inicial: 0.05), **nb_infected_init** (int, valor inicial: 5) y **agent_speed** (flotante, valor inicial: 5 # km / # h). Además, redefinimos la geometría del mundo por un cuadrado de 500 metros de tamaño y un paso de simulación de 1 minuto.

```
global {  
  int nb_people <- 500;  
  float agent_speed <- 5.0 #km/#h;  
  float infection_distance <- 2.0 #m;  
  float proba_infection <- 0.05;  
  int nb_infected_init <- 5;  
  float step <- 1 #minutes;  
  geometry shape<-square(500 #m);  
}
```

Fig 13. Variables y atributos.

4.1.4 MODELO DE INICIALIZACIÓN.

La sección init del bloque global permite inicializar el modelo. La declaración create permite crear agentes de una especie específica: create species_name+:

- number: número de agentes para crear (int, 1 por defecto)
- from: Archivo SIG que se utilizará para crear los agentes (cadena o archivo)
- returns: lista de agentes creados (lista)

Para nuestro modelo, la definición del init bloque para crear nb_people agentes de personas. Establecemos el valor inicial de la **speed** variable (dada por la moving habilidad) al valor de la variable global agent_speed (es decir, 5 km / h).

Además, pedimos a las nb_infected_init personas que se infecten (uso del **nb among list** para dibujar aleatoriamente elementos nb de la lista).

```

global{
  ...
  init{
    create people number:nb_people {
      speed <- agent_speed;
    }
    ask nb_infected_init among people {
      is_infected <- true;
    }
  }
}

```

Fig 14. Creación de agentes de especies específicas.

4.1.5 EXPERIMENTAR

Un **experiment** bloque define cómo se puede simular (ejecutar) un modelo. Se pueden definir varios experimentos para un modelo. Se definen usando: `experiment exp\_name type: gui/batch {[entrada] [salida]}`.

- `gui`: experimenta con una interfaz gráfica, que muestra sus parámetros de entrada y salidas.
- `lote`: permite configurar una serie de simulaciones (sin interfaz gráfica).

```

experiment main_experiment type: gui {
}

```

Fig 15. Creación de experimento

4.2 Gráficos

4.2.1 FORMULACIÓN

- Definición de nuevas variables globales: `hora actual`, `nb_people_infected`, `nb_people_not_infected`, `infected_rate`.
- Definición de un monitor para seguir la hora actual y el número de personas infectadas.
- Definición de un gráfico de serie para seguir el número de personas infectadas y no infectadas.
- Definición de una condición de detención (cuando la tasa de infección = 1).

4.2.2 VARIABLES GLOBALES

- **`nb_people_infected(int)`**: nb de personas con `is_infected` is true (uso del **list count condition** operador que cuenta el número de elementos de la lista para los que la condición es verdadera)
- **`nb_people_not_infected`** (En t): `nb_people - nb_people_infected`
- **`infected_rate`** (flotador): `nb_people_infected / nb_people`

```

global{
  ...
  int nb_people_infected <- nb_infected_init update: people count (each.is_infected);
  int nb_people_not_infected <- nb_people - nb_infected_init update: nb_people - nb_people_
  float infected_rate update: nb_people_infected/nb_people;
  ...
}

```

Fig 16. Variables para llevar el conteo de personas contagiadas y tiempo transcurrido.

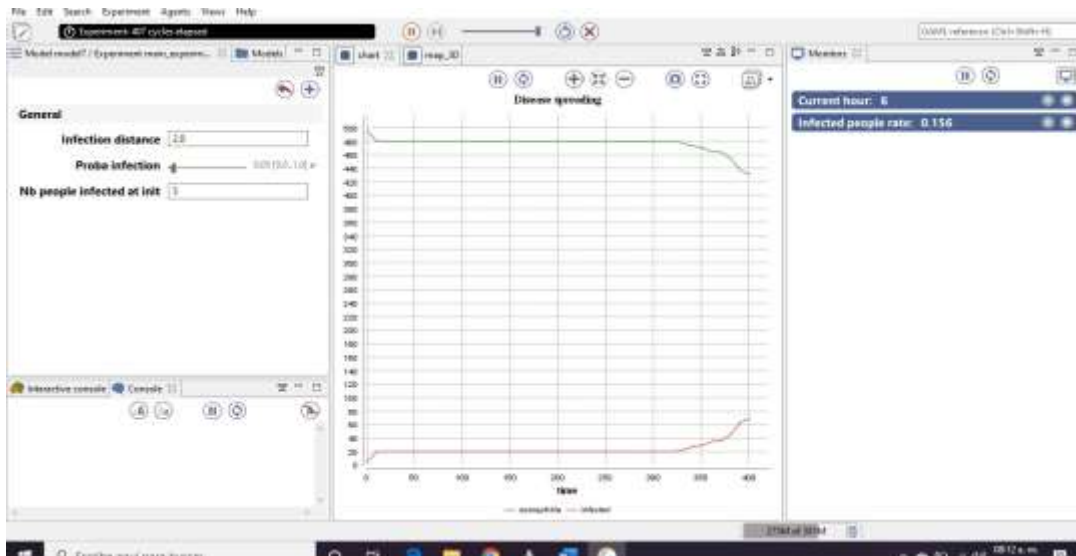


Fig 17. Interfaz de monitoreo de personas infectadas y horas transcurridas.

4.2.3 CONDICIÓN DE PARADA.

Agregamos un nuevo reflejo que detiene la simulación cuando todos los agentes de personas están infectados (es decir, si `infected_rate` es igual a 1). Para detener la simulación, usamos la **pause** acción global.

```

global {
  ...
  reflex end_simulation when: infected_rate = 1.0 {
    do pause;
  }
}

```

Fig 18. Código GAML para detener la simulación.

4.2.4 MONITOR

Un monitor permite al modelador seguir el valor de una expresión arbitraria en GAML. Tiene que definirse en una sección de salida. A **monitor** se define de la siguiente manera:

```
monitor monitor_name value: an_expression refresh:every(nb_steps);

experiment main_experiment type: gui {
  ...
  output {
    monitor "Current hour" value: current_date.hour;
    monitor "Infected people rate" value: infected_rate;
    ...
  }
}
```

Fig 19. Código GAML para crear monitor de resultados de la simulación.

4.2.5 GRAFICO

Agregamos una nueva pantalla llamada "gráfico" y la actualizamos cada 10 pasos de simulación. Dentro de esta pantalla, definimos un gráfico de tipo series:

- data1: susceptible; color verde
- data2: infectado; color rojo

```
experiment main_experiment type: gui{
  ...
  output {
    ...
    display chart refresh: every(10#cycles) {
      chart "Disease spreading" type: series style: spline {
        data "susceptible" value: nb_people_not_infected color: #green;
        data "infected" value: nb_people_infected color: #red;
      }
    }
  }
}
```

Fig 20. Código GAML para representación de gráficos.

4.3 Integración de datos SIG.

4.3.1 CARGA DE ARCHIVOS SIG

Uno permite al usuario elegir el archivo de forma de la carretera, otro que le permite elegir el archivo de forma del edificio.

```
global {  
  file shape_file_buildings <- file("../includes/building.shp");  
  file shape_file_roads <- file("../includes/road.shp");  
}
```

Fig 21.Código GAML para cargar los archivos SIG.

4.3.2 RESULTADO

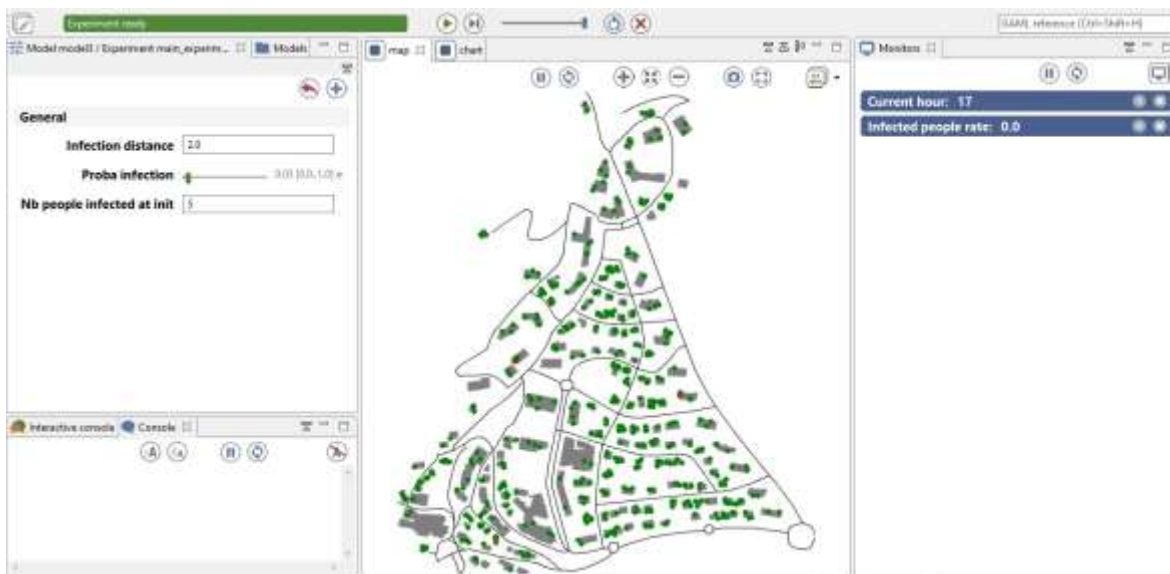


Fig 22. Edificios y carreteras cargadas en la simulación.

4.4 Movimiento en el gráfico.

4.4.1 FORMULACIÓN

- Definición de un gráfico global para representar la red de carreteras.
- Definición de una nueva variable global: **staying_coeff** para representar el hecho de que las personas se mueven más cerca de las 9h, 12h y 18h.
- Definición de dos nuevas variables para los agentes de personas: **target** y **staying_counter** (para gestionar su movilidad).
- Definición de un nuevo reflejo de los agentes de la gente: **stay**.
- Modificación del move reflejo de las personas agentes.

4.4.2 VARIABLES GLOBALES

- **road_network** (gráfico): representa el gráfico construido a partir de la red de carreteras.
- **staying_coeff** (flotación): representa el hecho de que las personas tienen más posibilidades de mudarse de su edificio actual cerca de las 9h (ir a trabajar), 12h (hora del almuerzo) y 18h (irse a casa). Esta variable se actualiza en cada paso de simulación (usando la update faceta).

```
20 graph road_network;  
21 float staying_coeff update: 10.0 ^ (1 + min([abs(current_hour - 9), abs(current_hour - 12), abs(current_hour - 18)]))
```

Fig 23. Código para movimiento de agentes sobre carreteras y edificios.

4.4.3 AGENTE DE PERSONAS.

- **target** (punto): la ubicación objetivo a la que las personas quieren llegar (un punto dentro de un edificio).
- **staying_counter** (int): el número de ciclos desde que el agente llegó a su edificio. Definimos un nuevo reflejo llamado **stay** que se activa cuando el agente no tiene objetivo (**target** = nil), es decir, cuando el agente está dentro de un edificio. Este reflejo incrementa el **staying_counter**, luego prueba la probabilidad de salir que se calcula a partir de **staying_counter** (cuanto más tiempo el agente esté dentro del edificio, más posibilidades tiene de irse) y el **staying_coeff** (más cerca de las 9h, 12h y 18h, más el agente tiene una oportunidad de irse). Si los agentes deciden irse, calcula un nuevo objetivo como un punto aleatorio dentro de uno de los edificios (elegido aleatoriamente).

```

43 species people skills:[moving]{
44   bool is_infected <- false;
45   point target;
46   int staying_counter;
47   reflex staying when: target = nil {
48     staying_counter <- staying_counter + 1;
49     if flip(staying_counter / staying_coeff) {
50       target <- any_location_in (one_of(building));
51     }
52   }
53 }

```

Fig 24. Código para saber la ubicación y tiempo que se encuentra dentro del edificio.

4.4.4 RESULTADOS

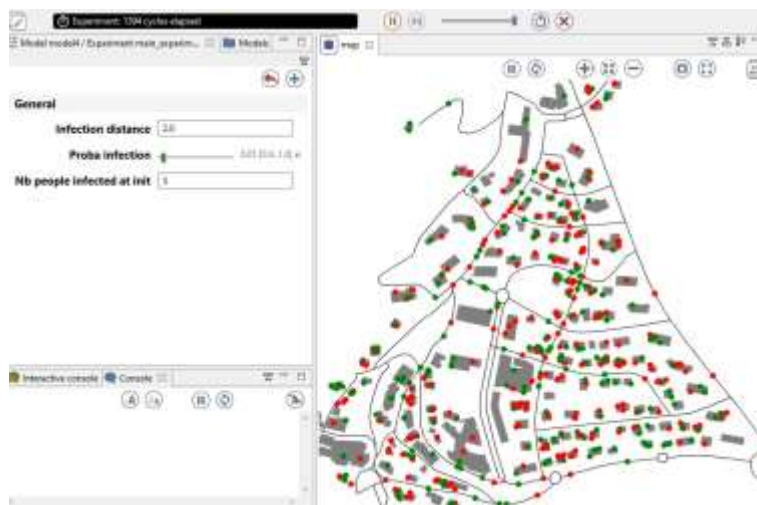


Fig 25. Agentes en edificios y carreteras.

4.5 SIMULACION EVACUACION DE LA CIUDAD

DESCRIPCION: En este modelo se simula un accidente tecnológico en un edificio de la zona centro.

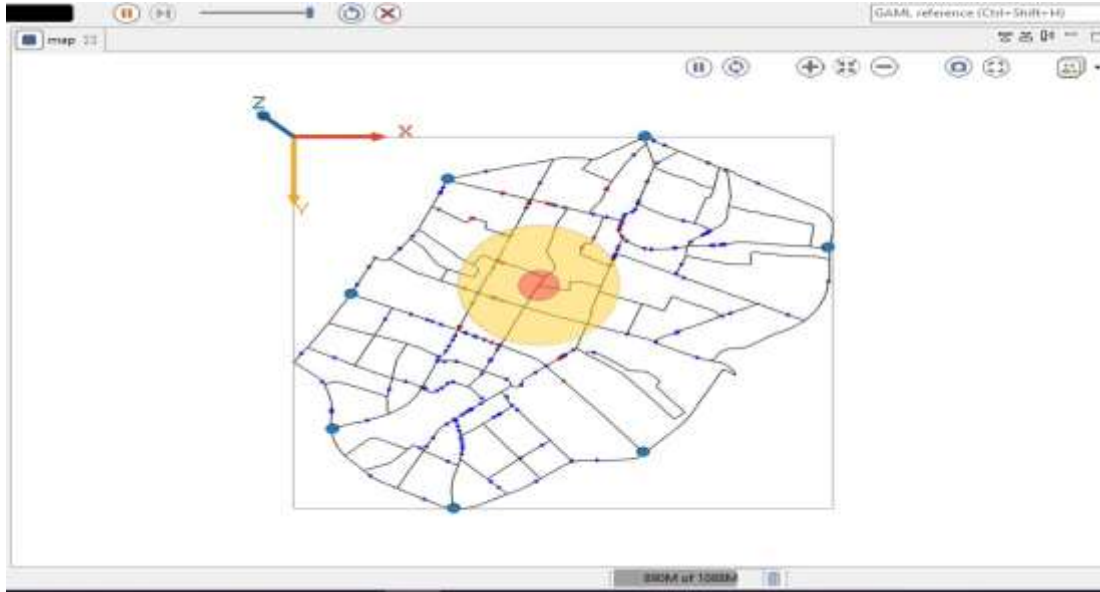


Fig 26. Simulacion de accidente tecnológico en el centro urbano.

4.5.1 CREACION DE CODIGO DE MODELO DE ESCAPE DE LA CIUDAD.

Para obtener esta simulación fue necesario el uso de archivos .shp y archivos .PNG incluidos en una carpeta llamada (includes).

- ▼ includes
 - > Escapes.shp (7 objects | Lambert_Conformal_Conic | 6151m x 4956m)
 - > Fire.png (626x854 | PNG)
 - > Rouen roads.shp (154 objects | Lambert_Conformal_Conic | 6516m x 5144m)
 - > Technological hazard.shp (1 object | Lambert_Conformal_Conic | 0m x 0m)

Fig 27. Archivos .shp y .png incluidos en la simulación

4.5.2 PARÁMETROS

GAMA nos permite cargar automáticamente datos GIS como .shp y PNG que tengamos incluidos en la carpeta (includes). Para que podamos elegir los archivos de forma más rápida, en este caso definimos 3 parámetros, uno que nos permite cargar la carretera, otro las formas

de los edificios y un último que nos permite obtener información sobre límites. Estos archivos los mandamos llamar con la extensión file como se muestra en la(fig2).

```
global {
  file shapefile_roads <- file("../includes/Rouen roads.shp");
  file shapefile_hazard <- file("../includes/Technological hazard.shp");
  file shapefile_shelters <- file("../includes/Escapes.shp");
```

Fig 28. Archivos .shp cargados en el modelo con la extensión file.

4.5.3 AGENTIFICACIÓN DE DATOS SIG

La agentificación de los datos GIS, usamos el comando “create” y “from” para pasar el archivo de forma. Cada objeto del archivo de forma se utilizará directamente para crear una instancia de un agente de la especie especificada.

```
init {
  create road from: shapefile_roads;
  create hazard from: shapefile_hazard;
  create catastrophe;
  create shelter from: shapefile_shelters;
```

Fig 29. Creación de agentes de datos para obtención de forma del modelo.

Inicializamos la simulación creando agente con el comando “people” seguido de “number” e indicamos el número de agentes con los que se empezara la simulación.

```
// al comienzo de la simulación, agregamos los agentes de personas el deseo de ir a su objetivo.
create people number: 200{
  location <- any_location_in(one_of(road));
  do add_desire(at_target);

  // el agente también tiene el deseo de que no haya catástrofe (establecemos la prioridad de este deseo en 0.
  do add_desire(nonCatastrophe ,0.0);

  // damos el agente de un carisma al azar y la receptividad
  charisma<-rnd(1.0);
  receptivity<-rnd(1.0);

  if(flip(0.9)){
    fearful<-true;
  }else{
    fearful <- false;
  }
}
road_network <- as_edge_graph(road);
current_weights <- road as_map (each::each.shape.perimeter);
```

Fig 30. Inicializando simulación creando 200 agentes para realizar el objetivo.

Aquí le damos los atributos a los agentes que creamos al inicializar la simulación con el comando “species” + el nombre de con el que creamos los agentes este caso “people” como se muestra en la

```
species people skills: [moving] control: simple_bdi{  
  point target;  
  float speed <- 30 #km/#h;  
  rgb color <- #blue;  
  bool escape_mode <- false;  
  bool fearful;
```

Fig 31. Otorgándole atributos a los agentes de la simulación.

4.5.4 CREACIÓN DE MODELO SIMPLE_BDI

Las creencias, deseos e intenciones las creamos mediante comando “predicate”, en este modelo construimos cuatro deseos como variables, al igual que 2 creencias, las variables deseos nos dan los objetivos que desean lograr los agentes en la simulación dentro de las 2 variables de creencias el entorno que creamos para que se moviera ese agente, código en

```
// con el fin de simplificar el modelo definimos 4 deseos como variables  
predicate at_target <- new_predicate("at_target");  
predicate in_shelter <- new_predicate("shelter");  
predicate has_target <- new_predicate("has target");  
predicate has_shelter <- new_predicate("has shelter");  
  
// les damos también 2 creencias como variables  
predicate catastropheP <- new_predicate("catastrophe");  
predicate nonCatastrophe <- new_predicate("catastrophe", false);
```

Fig 32. Creación de creencias y deseos de un modelo BDI.

Las intenciones de un agente pueden estar acompañadas de emociones y percepciones, para esta simulación las intenciones son que los agentes detecten emocionalmente cuando se encuentran en peligro por una catástrofe ocurrida dentro de la simulación y a su vez avisen a otros agentes sobre dicha catástrofe.

```

// al final definimos 2 emociones ligadas al conocimiento de la catástrofe
emotion fearConfirmed <- new_emotion("fear_confirmed",catastrophe);
emotion fear <- new_emotion("fear",catastrophe);

bool noTarget<-true;

// establecemos esta variable incorporada en true para utilizar el proceso emocional
bool use_emotions_architecture <- true;

// si el agente percibe que es algo que no es normal (un peligro), tiene una probabilidad proba_detect_hazard de suponer (agregar a su base de incertidumbre) que está ocurriendo
perceive target:hazard in: hazard_distance when: not escape_mode and flip(proba_detect_hazard){
  focus id:"catastrophe" is_uncertain: true;
  ask myself{
    if(fearful){
      do to_escape_mode;
    }else{
      color<-#green;
    }
  }
}

// si el agente percibe la catástrofe, se añade una creencia acerca de ella y pasar en el modo de escape
perceive target:catastrophe in:catastrophe_distance{
  focus id:"catastrophe";
  ask myself{
    if(not escape_mode){
      do to_escape_mode;
    }
  }
}

```

Fig 33. Creación de emociones y percepciones de los agentes en la simulación.

La creación de reglas va ligadas a las creencias y deseos que tienen los agentes en la simulación, si el agente detecta que hay peligro en su entorno la regla es ponerse en resguardo lo más pronto posible.

```

// si el agente tiene un miedo confirmado, tiene el deseo de ir a un refugio
rule emotion:fearConfirmed remove_intention: at_target new_desire:in_shelter strength:5.0;

// si el agente tiene la creencia de que hay una catástrofe, que tiene el deseo de ir a un refugio
rule belief:new_predicate("catastrophe") remove_intention:at_target new_desire:in_shelter strength:5.0;

rule emotion:new_emotion("fear",new_predicate("catastrophe")) new_desire:in_shelter remove_intention:at_target when: fearful strength:5.0;

// plan de movimiento normal
plan normal_move intention: at_target {
  if (target = nil) {
    target <- any_location_in(one_of(road));
  } else {
    do goto target: target on: road_network move_weights: current_weights recompute_path: false;
    if (target = location) {
      target <- nil;
      noTarget<-true;
    }
  }
}

```

Fig 34. Creación de reglas para resguardo de agentes.

4.5.5 Aspectos de agentes y catástrofes en la simulación.

```
= aspect default {  
  draw triangle(30) rotate: heading + 90 color: color;  
}  
}  
= species road {  
  float capacity <- 1 + shape.perimeter/50;  
  int nb_people <- 0 update: length(people at_distance 1);  
  float speed_coeff <- 1.0 update: exp(-nb_people/capacity) min: 0.1;  
  aspect default {  
    draw shape color: #black;  
  }  
}  
= species shelter {  
  aspect default {  
    draw circle(30) color: rgb(#gamablue,0.8) border: #gamablue depth:10;  
  }  
}  
= species hazard {  
  aspect default {  
    draw circle(hazard_distance) color: rgb(#gamaorange,0.3) border:#gamaorange depth:5;  
  }  
}  
= species catastrophe{  
  init{  
    location <- first(hazard).location;  
  }  
}
```

Fig 35. Creación de aspecto de agentes y zonas de la catástrofe.

Al final agregamos un experimento para que nos diera el resultado de en cuanto tiempo los agentes detectan el peligro, avisan a los demás agentes y logran ponerse a salvos en los puntos de salvación

```
3 experiment main type: gui {  
4   float minimum_cycle_duration <- 0.02;  
5   output {  
6     display map type: opengl{  
7       species shelter refresh: false;  
8       species road refresh: false;  
9       species people;  
0       species catastrophe;  
1       species hazard;  
2     }  
3   }  
4 }  
5
```

Fig 36. Creación del experimento para obtener resultados de la simulación.

4.6 CONCLUSIONES

Durante esta investigación se analizaron diversas metodologías tanto para la definición de modelos basados en agentes como para definir modelos de simulación multiagente considerando ventajas y desventajas de cada una de ellas con respecto a su aplicación para llevar a cabo experimentación que pueda ser aplicable a problemas con características que en un determinado momento puedan servir de referencia para estudios y/o investigación en el campo de la geomática. En este sentido, se identificó la metodología BDI para definir la estructura de los agentes de acuerdo a sus tres elementos básicos: creencias, deseos e intenciones. Con esta metodología se pueden definir diversos tipos de agentes (reactivos, proactivos, híbridos, etc.) en función del tipo de problema a modelar, en esta investigación se trabajó con modelos BDI de propagación y de evacuación.

Para la definición del modelo de simulación se utilizó la metodología GAIA considerando aspectos como: la definición del entorno, la definición de roles, y la definición del esquema de comunicación, considerando que esta metodología se alinea con el estándar FIPA para establecer mecanismos de comunicación entre agentes.

Como resultado de esta tesis se pudieron realizar experimentos con simulaciones implementadas en GAMA, un entorno computacional que se identificó también durante esta investigación y que tiene la característica de implementar con cierto grado de transparencia la metodología BDI a través de su lenguaje de programación llamado GAML, un lenguaje de cuarta generación que fundamenta su estructura semántica en la lógica de predicados. Utilizando GAMA y GAML se pudieron correr simulaciones de sistemas multiagente utilizando como parámetros de entrada entre otros, un archivo shape de un mapa para simular rutas de evacuación en una ciudad ante la ocurrencia de catástrofes. Esto es representa un elemento importante para aplicar este tipo de modelos en el área de geomática al poder incorporar datos georeferenciados a una simulación prospectiva de este tipo.

El identificar las metodologías utilizadas en esta investigación permitió aprovechar las ventajas que ofrece el hecho de que sean estándares que se pueden adaptar al análisis de problemas de diversa índole, además, se identificó y probó el uso de un entorno computacional que es libre y de código abierto, lo que nos brinda la posibilidad de utilizar y/o adaptarlo conforme a las necesidades de los modelos con lo que se esté trabajando.

Por último, como trabajo futuro es importante seguir trabajando en añadir datos cartográficos que representen el escenario correspondiente a zonas de estudio de interés y que en lo que corresponde a las metodologías utilizadas en esta tesis representarían los tipos de agentes y al escenario o “mundo” en el cual se desarrollará la simulación multiagente.

BIBLIOGRAFIA

- Adam, C., Gaudou, B., Hickmott, S., Scerri, D.: Agentes BDI y simulaciones sociales. unis pour le meilleur et pour le pire. En: RIA, págs. 11–43 (2011).
- Behzadi, S., Alesheikh, AA: Presentación de un modelo novedoso de agente de creencia-deseo-intención para la planificación del uso del suelo urbano. Ing. Appl. Artif. Intell. 26 (9), 2028–2044 (2013).
- C. Cantergiani, M. Gómez Delgado y C. Vergara, «Desarrollo de un Modelo Basado en Agentes para la simulación del crecimiento urbano: Submodelo de asignación de nuevo suelo urbanizable,» Asociación de Geógrafos Españoles (AGE), 2014.
- Caillou, Philippe & Gaudou, Benoit & Grignard, Arnaud & TRUONG Chi, Quang & Taillandier, Patrick. (2017). A simple-to-use BDI architecture for agent-based modeling and simulation.
- Ferber, J. (1999). Multi-agent System. An introduction to distributed Artificial Intelligence. london: Addison Wesley
- GAMA website (2015). <http://gama-platform.org>.
- Hayes-Roth, F., Erman, L. D., Lesser, V. R., & Reddy, D. R. (1980). The Hearsay II speech understanding system: integrating Knowledge to resolve uncertainty. En ACM Computing Survey (Vol. XII, págs. 213-253).
- Hewitt, C. (1977). Viewing Control Structures as Patterns of Message Passing. En Artificial Intelligence (Vol. VIII, págs. 323-364).
- Jennings, N., Sycara, K., & Wooldridge, M. (1998). Autonomous Agents and MultiAgent Systems. A Roadmap of Agent Research and Development, I(1), 7- 38
- Kravari, Kalliopi & Bassiliades, Nick. (2015). A Survey of Agent Platforms. Journal of Artificial Societies and Social Simulation. 18. 10.18564/jasss.2661.
- Lesser, V. R., & Corkill, D. G. (1983). The distributed Vehicle Monitoring Tested: A Tool for Investigating Distributed Problem Solving Networks. AI Magazine, 4(3), 12-33.
- M. Gómez y J. Barredos, Sistemas de información geográfica y evaluación multicriterio en la ordenación del territorio, 2006.
- Muller, J. (1998). Architectures and application of intelligent agents: a survey. Knowledge Engineering Review, 13(4), 353-380.

- Muñoz, N., Cobos, C., Rivera, W., López, J., & Mendoza, M. (2010). Use of GAIA for modeling the behavior of characters in a real time strategy game. *Revista Facultad de Ingeniería Universidad de Antioquia*, (53), 214-224.
- Pavón, (2004) de Sistemas Multi-Agente, D. Agentes Inteligentes.
- R. Rodríguez, «Modelación de flujo de tránsito de autos utilizando autómatas celulares,» Centro de Investigación y de Estudios Avanzados del IPN, México, 2002.
- Rao, A., Georgeff, M.: Modelado de agentes racionales dentro de una arquitectura BDI. En: Actas de la 2da Conferencia Internacional sobre Principios de Representación del Conocimiento y Razonamiento, pp. 473–484 (1991).
- Sakellariou, I., Kefalas, P., Stamatopoulou, I.: Enhancing NetLogo to simulate BDI communicating agents. In: Artificial Intelligence: Theories, Models and Applications, pp. 263–275. Springer, Berlin (2008).
- Sanz, J. J. G. (2003). Metodologías para el desarrollo de sistemas multiagente. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 7(18), 51-63.
- shoham, Y. (1993). Agent Oriented Programming. *Artificial Intelligence*, 60(1), 52- 92.
- Smith, R. G. (1980). The contract net protocol: high level communication and control in distributed problem solver. En *IEEE Transactions on Computers* (Vol. 29, págs. 1104-1113).
- Taillandier, Patrick & Bourgais, Mathieu & Caillou, Philippe & Adam, Carole & Gaudou, Benoit. (2017). A BDI Agent Architecture for the GAMA Modeling and Simulation Platform. 3-23. 10.1007/978-3-319-67477-3_1.
- Uribe Vásquez, J. A. (2015). *Desarrollo de un modelo de simulación utilizando Sistemas Multi-Agente (SMA) aplicado al juego de la cerveza en el software Anylogic®* (Bachelor's thesis, Universidad EAFIT).
- W. Jager et al. (eds.), *Advances in Social Simulation 2015*, *Advances in Intelligent Systems and Computing* 528, DOI 10.1007/978-3-319-47253-9_2.
- Wooldridge, M., & Jennings, N. (1995). Intelligent agent: theory and practice. *Knowledge Engineering Review*, X (2), 115-142.