



Universidad Tecnológica de La Habana

“José Antonio Echeverría”

Facultad de Ingeniería Informática

Simulación basada en agentes en el problema de máxima cobertura

Informe de las prácticas profesionales de 4to año

Autor:

Jesús Hernández Barrios (jhernandezb96@gmail.com)

Tutor:

Dr. C. Mailyñ Moreno Espino (my@ceis.cujae.edu.cu)

Profesora Titular de la Facultad de Ingeniería Informática, Cujae, La Habana, Cuba.

La Habana, Cuba

Enero, 2021

Resumen

Las patrullas policiales juegan un papel importante en el servicio público para responder a incidentes, disuadir y prevenir delitos. El mal diseño y planificación del patrullaje policial puede traer consigo innumerables efectos negativos como los accidentes, y puede afectar varios factores importantes en la actuación del patrullaje como el tiempo de respuesta promedio. En nuestro enfoque, los planes de patrullaje generados se evalúan utilizando un modelo de simulación basado en agentes. En la actualidad, la simulación de procesos es una solución para obtener dicha información. Según el tipo de simulación empleado, los resultados serán más o menos cercanos a la realidad. La simulación basada en agentes, por sus características, aporta ventajas en cuanto a simulaciones de redes viales.

El presente trabajo describe el proceso de desarrollo y mejora de una herramienta de simulación basada en agentes que contribuye al proceso de configuración del patrullaje policial. El desarrollo de esta solución está fundamentado en agregar mejoras en el diseño e implementación de dicha herramienta logrando ver, lo más real posible, su efectividad en una situación específica, teniendo en cuenta experiencias previas con el fin de lograr un resultado ajustado a las necesidades y capacidades reales. El resultado fue la creación de una herramienta GIS para la simulación basada en agentes del patrullaje policial del municipio de Centro Habana utilizando el framework MASON.

Palabras clave: GIS, Patrullaje policial, Simulación basada en agentes.

Abstract

Police patrols play an important role in public service to respond to incidents, deter and prevent crime. Poor design and planning of police patrols can have innumerable negative effects such as accidents and can affect several important factors in patrol performance such as average response time. In our approach, the generated patrol plans are evaluated using an agent-based simulation model. Currently, process simulation is a solution to obtain such information. Depending on the type of simulation used, the results will be more or less close to reality. Agent-based simulation, due to its characteristics, provides advantages in terms of road network simulations.

This work describes the development and improvement process of an agent-based simulation tool that contributes to the police patrol configuration process. The development of this solution is based on adding improvements in the design and implementation of said tool, making it possible to see, as real as possible, its effectiveness in a specific situation, taking into account previous experiences in order to achieve a result adjusted to the needs and actual capabilities. The result was the creation of a GIS tool for the simulation based on police patrol agents in the municipality of Centro Habana using the MASON framework.

Keywords: GIS, Police patrol, Agent-based simulation

Índice

Introducción	1
Capítulo 1: Fundamentos Teóricos	7
1.1 Introducción.....	7
1.2 Agentes Inteligentes.....	7
1.2.1 Sistemas multi-agente	8
1.3 Modelado y simulación	8
1.3.1 Simulación basada en agentes	10
1.4 Problema de Máxima Cobertura	11
1.4.1 Definición del MCLP	12
1.5 Sistema de Información Geográfica	14
1.6 Patrullaje Policial	15
1.7 Plataformas de agentes.....	17
MASON.....	17
JADE	18
JADEX.....	20
NetLogo	21
RePast	23
GAMA.....	23
1.8 Comparación entre plataformas	25
1.9 Arquitectura de MASON	25
1.9.1 Estilos y patrones arquitectónicos	26
1.9.2 Vista general de la arquitectura	27
1.9.3 Vista detallada de la arquitectura	28
1.10 Conclusiones parciales.....	31
Capítulo 2: Solución Propuesta	33

2.1 Introducción	33
2.2 Propuesta inicial del modelo matemático	33
2.2.1 Distancia mínima entre dos puntos	35
2.3 Modelo del Dominio.....	36
2.3.1 Clases del modelo del dominio.	36
2.4 Diagrama de actividades del flujo completo del negocio	37
2.5 Reglas del negocio	38
2.5.1 Identificación de los Requisitos.....	39
2.5.2 Requisitos no funcionales	40
2.6 Diagrama de casos de uso del sistema	41
2.6.1 Actores del sistema.....	42
2.7 Vista de la arquitectura: Estructuración en capas	48
2.7.1 Enfoque por reutilización.....	48
2.8 Patrones de diseño	50
2.8.1 <i>Facade</i>	50
2.8.2 Data Access Object.....	52
2.8.3 <i>Template</i>	52
2.8.4 <i>Factory</i>	52
2.8.5 <i>Singleton</i>	52
2.8.6 <i>Adapter</i>	52
2.8.7 <i>Bridge</i>	52
2.8.8 <i>Proxy</i>	53
2.8.9 Observer	53
2.9 Persistencia de datos del software.	53
2.10 Interfaces visuales del software	54
2.10.1 Interfaz visual About	55

2.10.2 Interfaz visual Console	55
2.10.3 Interfaz visual Displays.....	56
2.10.4 Interfaz visual de Inspectors.....	57
2.10.5 Interfaz visual Auxiliar.....	57
2.11 Interfaz de usuario.....	58
2.11.1 Visualización	58
2.11.2 Inspectores.....	59
2.12 Conclusiones parciales.....	59
Capítulo 3: Validación de la herramienta.....	60
3.1 Introducción	60
3.2 Pruebas funcionales.....	60
3.2.1 Guiadas por casos de uso	60
3.2.2 Clases de equivalencia.....	61
3.2.3 Análisis de los resultados de las pruebas	62
3.3 Diseño de experimentos.....	63
3.3.1 Fase de planeación.....	63
3.3.2 Fase de diseño	64
3.3.3 Fase de conducción.....	64
3.3.4 Fase de análisis.....	65
3.3.5 Conclusiones del DoE	68
3.4 Conclusiones parciales.....	68
Conclusiones Generales	70
Recomendaciones	71
Referencias bibliográficas.....	72

Índice de Tablas

Tabla 1: Comparación de plataformas de agentes	25
Tabla 2: Glosario de términos.	37
Tabla 3: Reglas del negocio	39
Tabla 4: Identificación de los requisitos.....	40
Tabla 5: Requisitos no funcionales.....	40
Tabla 6: Actores del sistema	42
Tabla 7: caso de uso:" Configurar Simulación"	42
Tabla 8: Caso de uso:" Ejecutar Simulación"	45
Tabla 9: Caso de uso:" Generar Datos"	47
Tabla 10: Descripción de paquetes de la arquitectura	49
Tabla 11: Descripción del caso de prueba: "Mala configuración de la simulación"	60
Tabla 12: Descripción del caso de prueba: "Poner en funcionamiento la simulación"	61
Tabla 13: Clases de equivalencia.....	62
Tabla 14: Pruebas de las clases de equivalencia	62
Tabla 15: Factores controlables.....	64
Tabla 16: Rendimiento de cada tratamiento	64

Índice de Figuras

Figura 1:Esquema general del proceso de modelado científico	9
Figura 2: Esquema de Paradigmas de Simulación	11
Figura 3: Representación binaria del problema MCLP.....	13
Figura 4: Representación gráfica del problema MCLP.....	14
Figura 5: Sistema de información geográfica	15
Figura 6: Arquitectura de plataforma JADE	19
Figura 7: Arquitectura MASON, patrón MVC.....	26
Figura 8: Relaciones primarias entre el modelo y la GUI visualización / control	27
Figura 9: Diagrama de la simulación de núcleo de MASON y de la programación de eventos discretos	28
Figura 10: Diagrama del paquete Network de MASON.....	29
Figura 11: Diagrama del código de control de GUI de MASON	30
Figura 12: Instalación Inspector de MASON	31
Figura 13: Modelo del dominio	36
Figura 14: Diagrama de actividades.....	38
Figura 15: Diagrama de casos de uso del sistema	41
Figura 16: Diagrama de estructuración en capas. Enfoque por reutilización	48
Figura 17: Diseño de las utilidades, iteración 1	50
Figura 18: Diseño de las utilidades, iteración 2 (patrón Facade)	51
Figura 19: Estructura del fichero XML para los datos del problema MCLP.	54
Figura 20: Estructura de fichero XML para las soluciones del problema PoliceCarsSimulation.	54
Figura 21: Interfaz visual About.....	55
Figura 22: Interfaz visual Console	56
Figura 23: Interfaz visual Display.	56
Figura 24: Interfaz visual Inspectors.....	57
Figura 25: Interfaz visual Auxiliar	58
Figura 26: Arquitectura de la GUI.....	59
Figura 27: Gráfica de efectos principales	65
Figura 28: : Grafica de Pareto	66

Figura 29: Gráfica de probabilidad normal.	67
Figura 30: Grafica de cubos.	68
Figura 31: Visualización del entorno de la simulación en la primera versión.	77
Figura 32: Visualización actual del entorno de la simulación.	78
Figura 33: Pasos a seguir en la creación de un archivo XML.	¡Error! Marcador no definido.
Figura 34: Formato del archivo .xml generado al finalizada la simulación	¡Error! Marcador no definido.
Figura 35: Interfaz Visual Inspector Model	¡Error! Marcador no definido.
Figura 36: Visual Entorno General	79

Introducción

En la actualidad, la puesta en marcha de un proceso o sistema sin antes tener cierto grado de confianza de cómo será su funcionamiento, no constituye una buena práctica. Generalmente la puesta en funcionamiento de un sistema es costosa, y de no ser efectivo su despliegue, puede ocasionar grandes pérdidas materiales y malestar a los usuarios y personal involucrado. Una manera de enfrentar este problema es utilizando técnicas de simulación, en donde se pretende abstraer la situación a un modelo matemático compatible y simularlo computacionalmente [1].

Una simulación computacional es un proceso que se aplica sobre un modelo que expresa el comportamiento de un proceso real o abstracto en el tiempo. El acto de simular requiere además del desarrollo de un modelo previo que sea fehaciente con el proceso que se quiere simular. Dicho modelo representa las características claves del comportamiento y funcionamiento del sistema o proceso que se quiere simular [2].

La necesidad de simular surge del propio factor tiempo y disponibilidad. Se utiliza cuando se desea realizar experimentos que son imposibles, imprácticos y con alto riesgo en un sistema real o que simplemente la fragilidad del sistema no soportaría las pruebas extensivas a los que sería sometido para el tiempo de duración del mismo, por lo que a menudo son consideradas como prototipos iniciales de un proyecto [3]. Además, provee al usuario final información de referencia y clave en el diseño de sistemas en la vida real, y posibilita al creador agudizar la eficacia de los diseños antes que los procesos o sistemas diseñados sean construidos físicamente [4].

Existen varios tipos de simulación computacional, entre ellas, no se puede determinar cuál es la más conveniente utilizar para todos los tipos de problemas [5, 6]. Según las características de un modelo, el tipo de simulación a aplicar, puede brindar resultados favorables o no. Los procesos de simulación computacional han dado a lugar al surgimiento de las Simulaciones Basadas en Agentes (ABS, por sus siglas en inglés), donde se utiliza las habilidades de los agentes para potenciar los propios procesos simulados en ambientes donde son necesarios la incorporación de las características que los agentes brindan. Este nuevo fenómeno representa hoy una alternativa interesante a la simulación tradicional ya que, con la propia evolución de los seres humanos, los procesos en los que se ve envuelto son cada vez más complejos,

haciendo que los métodos tradicionales para representar procesos simulados en los que intervienen seres humanos no correspondan con la realidad. Estos nuevos procesos complejos son denominados “Procesos adaptativos complejos” (CAS, por sus siglas en inglés) [7, 8] y responden a la incorporación del elemento proactivo para adecuarse a las situaciones de manera que un ser humano haría tradicionalmente sujeto a valores volátiles, tales como la situación problemática que tenga en el momento y espacio en el que debe tomar una decisión. Una temática en la que la ABS puede aportar ventajas es en el problema Máxima Cobertura [9].

El problema Máxima Cobertura (MCLP, por sus siglas en inglés) se describe como la selección de una cantidad predeterminada de instalaciones entre las posibles alternativas, de modo que se maximice la cobertura sobre nodos de demanda a satisfacer [10]. El problema MCLP consta, esencialmente, de un número conocido de instalaciones a ubicar (p), la localización de todas las instalaciones y la localización y demanda de los nodos que deben ser satisfechos. El objetivo de este problema es: del total de instalaciones, buscar la combinación de p instalaciones que maximicen la cobertura sobre los nodos de demanda, donde estas instalaciones están sujetas a restricciones de distancia [10, 11].

En el contexto actual cubano, en el Centro de Investigación de Tecnologías Integradas (CITI), para trabajar este problema se ha desarrollado la herramienta MCLP-Tools, que representa datos y soluciones sobre un mapa, aun así, este no abarca la simulación del comportamiento de las estaciones estaciones (dígase que debe moverse o cambiar del área de cobertura), que pueden ser desde estaciones fijas como antenas de telefonía móvil, hasta estaciones móviles como los vehículos de primer orden, dígase carros de policía, bomberos y ambulancias [12] en una situación requerida. Una mala distribución de estas estaciones puede provocar graves consecuencias, que van desde la baja cobertura en los teléfonos celulares hasta la llegada tarde de un carro de bomberos a un incendio y la posible pérdida de vidas humanas. Además, mejores planes de distribución pueden llevar a un mejor tiempo de respuesta, la familiarización de los oficiales con su área asignada, utilización eficiente del personal, distribución equitativa de la carga de trabajo, visibilidad uniforme de la policía, mayor seguridad de los oficiales, respuesta policial equilibrada a las llamadas y responsabilidad de la oficina.

Anteriormente se desarrolló un sistema que logra extraer de un modelo matemático la ubicación inicial de las estaciones de servicio de la solución del problema de máxima cobertura planteado, simulando su comportamiento en una situación previamente configurada, permitiendo observar dicho comportamiento.

Este sistema se encuentra en su etapa intermedia, y se quiere lograr que además de sus diferentes funcionalidades ya diseñadas e implementadas antes mencionadas se tomen en cuenta diferentes factores cercanos a la realidad a la hora de diseñar el patrullaje policial como son:

- El tamaño del área patrullada no debe exceder el límite de que un solo oficial pueda cubrir.
- Tener en cuenta que el límite de cada área puede ser naturales o artificiales (por ejemplo, ríos, vías férreas, carreteras, vales).
- Tener en cuenta indicadores de carga de trabajo (incidentes criminales, patrones de tráfico).
- Tener en cuenta las características significativas de las áreas (escuelas, tiendas, bancos, entre otros) ya que esto nos favorece a la hora de mejorar las medidas de rendimiento comunes que son el tiempo promedio de respuesta y la variación de carga de trabajo.
- Además de reconfigurar la Variante Operativa.

Luego de definir estos diferentes factores, se pretende crear una simulación que permita ver cuál es la mejor distribución dado determinado cambio en diferentes parámetros de la configuración (dígase aumento de la complejidad de la demanda) recopilando datos que permitan apoyar la toma de decisiones y así reconfigurar el posicionamiento y movimientos de las estaciones de servicio en determinados escenarios.

Según la problemática planteada anteriormente se ha identificado como **problema a resolver**: ¿Cómo tener en cuenta factores cercanos a la realidad para simular el sistema de simulación basada en agentes para las variantes operativas del municipio de Centro Habana y así hacer más efectivo su despliegue?

El **objeto de estudio** es: Simulación Computacional y Sistema de Información Geográfica (GIS); y de este amplio conjunto centra la atención en el **campo de acción** que comprenden: los agentes inteligentes, la ABS, los datos espaciales.

A partir del problema anteriormente mencionado, se define como **objetivo general**: Simular el proceso de respuesta a incidencias partiendo del problema de máxima cobertura para analizar el estudio de comportamientos de las estaciones de servicio teniendo en cuenta las características significativas de cada área.

Dicho objetivo se divide en los siguientes objetivos específicos y sus tareas:

1. Realizar un estudio del estado del arte sobre los temas principales a tener en cuenta en la simulación basada en agentes.

- Asimilar los conceptos de simulación y simulación basada en agentes.
- Realizar encuestas y entrevistas a diferentes instituciones.
- Asimilar los conceptos fundamentales a tener en cuenta en los procesos del patrullaje policial.
- Analizar los factores que influyen la distribución operativa de las patrullas y policías de a pie que se tiene en cuenta en la simulación.
- Realizar un estudio del manejo de los GIS de la plataforma MASON.

2. Rediseñar el modelo de simulación basada en agentes a partir del problema de máxima cobertura.

- Definir un modelo matemático con nuevas variables, parámetros y restricciones teniendo en cuenta las características significativas de cada área.
- Acoplar el modelo matemático teniendo en cuenta la simulación basada en agentes

3. Desarrollar el componente de simulación basada en agentes.

- Diseñar una arquitectura modular para el sistema que permita un fácil acoplamiento de módulos diferentes para el trabajo con múltiples sistemas de agentes interactuando entre sí.

- Diseñar un estándar para configuración y trabajo con ficheros externos que brinde personalización al sistema, como elegir el tipo de mapa y posiciones iniciales de los agentes.
- Definir modos de actuación a agentes inteligentes ante determinada emergencia.
- Generar automáticamente las demandas a satisfacer.
- Configurar la simulación con los parámetros de entrada.
- Crear inspectores en el entorno de simulación.
- Exportar datos de interés en la simulación.
- Mejorar la interfaz visual de la simulación.

4. Validar la solución propuesta.

- Definir y diseñar los casos de prueba.
- Desarrollar pruebas funcionales.
- Realizar un diseño de experimento.

El **valor práctico** de este trabajo es el nuevo software que permite la visualización, de forma clara y entendible a través de la simulación basada en agentes, utilizando como entrada inicial la solución de que genera el MCLP-Tool y las capacidades de GIS, donde se ubican un conjunto de unidades, aplicando simulación basada en agente y teniendo en cuenta factores como respuesta a incidencias, cambios de la variante operativa por aumento de incidencias se puede prever cómo se debe hacer la distribución de Patrullas y Policías en el Municipio Centro Habana.

En cuanto a la **estructuración**, este trabajo está estructurado en 3 capítulos. A continuación, se hará una descripción de cada uno de ellos:

Capítulo 1: Fundamentos teóricos, este capítulo contiene los fundamentos teóricos que resultan de un detallado estudio del estado del arte sobre las tecnologías de agentes, las propiedades que poseen. Se analiza, en qué consiste un proceso de simulación y los tipos que existen, prestando especial atención la Simulación Basada en Agentes. Se explica en qué consisten los Sistemas de Información Geográfica y los datos espaciales. Aborda los temas relacionados con el problema de máxima cobertura y las ventajas que posee, además de abarcar los procedimientos a tener en cuenta en

el patrullaje policial. Por último, se realiza un análisis comparativo entre diferentes plataformas de simulación basada en agentes.

Capítulo 2: Solución Propuesta, documenta el proceso de desarrollo de la herramienta, haciendo especial uso de numerosos artefactos de Ingeniería de Software. Este capítulo comienza mostrando el modelo matemático a diseñar y el diagrama de modelo del dominio y el diagrama de actividades. También se escribe el proceso de captura de requisitos que ayuda a identificar los casos de uso del sistema, así como el análisis detallado del proceso a seguir dándole solución a la situación problemática planteada.

Capítulo 3: Validación de la herramienta, se describe el proceso de validación de la herramienta de simulación basada en agentes. Para ello se realizan un conjunto de pruebas funcionales. Estas pruebas se guían por técnicas, guiada por casos de uso y clases de equivalencia. Luego, se presenta un Diseño de Experimentos donde se pone en uso la herramienta para un conjunto de experimentos a fin de sugerir una mejor configuración de la simulación.

Capítulo 1: Fundamentos Teóricos

1.1 Introducción

En este capítulo se abordan los aspectos teóricos fundamentales para desarrollar la propuesta del trabajo, entre los que se trata los Agentes y sus características. Se explica en qué consiste la simulación computacional, resaltando la simulación basada en agentes. Se realiza un análisis del proceso del modelado del problema de máxima cobertura y su importancia, se aborda como se lleva a cabo el proceso del patrullaje policial. También se explica cómo pudiese la simulación basada en agentes, favorecer en este procedimiento y se exponen las formas actuales de representar los datos espaciales. Se revisa la bibliografía de las plataformas actuales disponibles, para realizar un análisis comparativo y seleccionar una arquitectura candidata para el simulador.

1.2 Agentes Inteligentes

El paradigma de agentes ha sido concebido bajo diferentes puntos de vista y definiciones. La mayoría de estas coinciden en que son básicamente entidades de software, a los cuales se le adicionan elementos para lograr un comportamiento autónomo. Por estas características en específico, han sido objeto de ser introducidos en las simulaciones computacionales.

Según Wooldridge [13], no existe una definición aceptada de forma universal del término “agente”, y de hecho existe debate y controversia sobre este tema por la variedad de criterios. Una definición tomada de Wooldridge es: “Un agente es un sistema computacional situado en un entorno dado, que es capaz de actuar de forma autónoma para cumplir sus objetivos” [13].

Por otra parte, la FIPA (*Foundation for Intelligent Physical Agents*) [14] define al agente como una entidad de software con un grupo de propiedades, entre las que se destacan: ser capaz de actuar en un ambiente, comunicarse directamente con otros agentes, estar condicionado por un conjunto de tendencias u objetivos, manejar recursos propios, ser capaz de percibir su ambiente y tomar de él una representación parcial, ser una entidad que posee habilidades y ofrece servicios, que puede reproducirse, entre otras [14, 15]. De forma general, varios autores reconocen en los agentes diversas

propiedades, entre las que se destacan el ser autónomos, reactivos, proactivos y tener habilidad social [12]. Por estas características en específico, han sido objeto de ser introducidos en las simulaciones computacionales.

El desarrollo práctico de las simulaciones basadas en agentes se ha evidenciado gracias a la necesidad de cada vez más involucrar aspectos y características de los seres humanos dentro de los modelos de simulación tradicional. Los agentes pueden ser objetos de estudio de varias áreas, como lo son la inteligencia artificial, los sistemas distribuidos, la ingeniería de software, las redes y los sistemas autónomos, entre otras ramas de la Ciencia de la Computación. Producto de esto su definición práctica está influenciada según sea el área en la cual se tenga interés. Una definición apropiada de agente podría ser que son simplemente un sistema computacional con la capacidad de tomar acciones autónomas en un medio, para así cumplir sus objetivos [13,16].

1.2.1 Sistemas multi-agente

Cuando se habla de un sistema basado en agentes se hace referencia a un sistema multi-agente (SMA), con varios tipos de agentes (estáticos, dinámicos, reactivos, deliberativos, entre otros), con un número de entidades que puede variar. Un agente puede estar concebido como una entidad física o virtual que puede actuar, percibir en su ambiente (de forma parcial) y comunicarse con otros. Es dentro de un SMA donde se contiene el ambiente, los objetos y los agentes, las relaciones entre las entidades y las operaciones básicas que pueden realizar los agentes que la conforman específicamente dentro de los esquemas de simulación basada en agentes.

Según Wooldridge en las comunidades de agentes potencian elementos básicos de una micro sociedad en donde existen varias entidades que pueden tener o no el mismo patrón de comportamiento en dependencia de sus características previamente definidas [13].

1.3 Modelado y simulación

Varios filósofos (por ejemplo, Hesse (1963) y Hughes (1997)) que han estudiado la metodología científica tradicional han propuesto el mismo esquema general del proceso de modelado. Según estos autores, los modelos científicos se construyen para desarrollar procesos de inferencia sobre ciertos aspectos de sistemas reales previamente observados. Es mediante estos procesos de inferencia, mediante la

construcción y el uso de modelos científicos, como mejoramos nuestro entendimiento de los sistemas reales observados. Es decir, una eficaz forma de mejorar nuestro conocimiento del mundo en que vivimos es a través de la creación de modelos, no necesariamente formales [17]. El proceso de creación y uso de un modelo se muestra esquematizado en la Figura 1: Esquema general del proceso de modelado científico, la cual muestra un proceso secuencial con claridad, pero, hay que tener en cuenta que el proceso de modelado contiene en general varios bucles de retroalimentación.

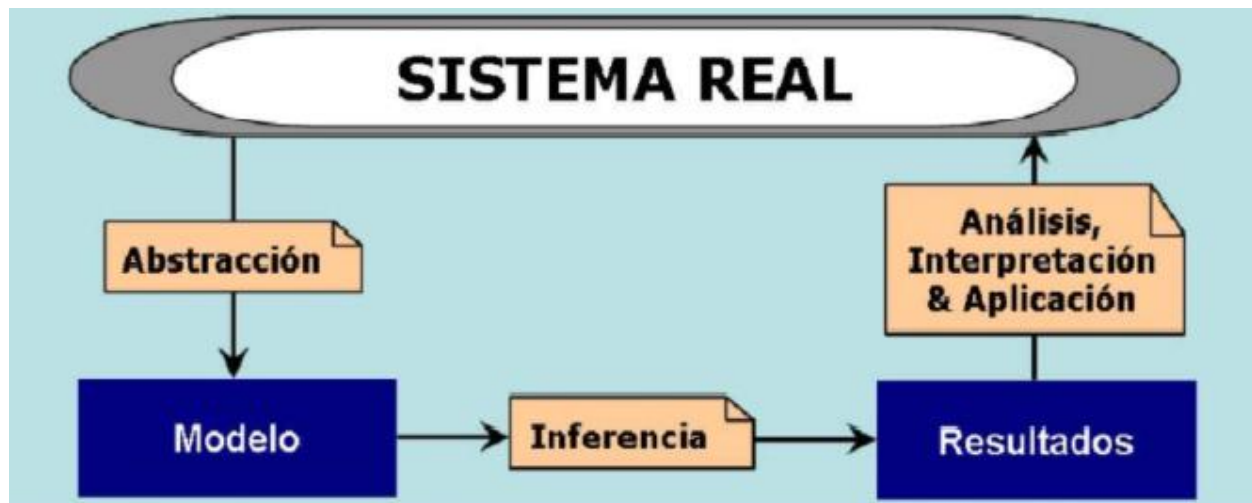


Figura 2:Esquema general del proceso de modelado científico

Varios autores coinciden que la simulación consiste esencialmente en una habilidad para representar procesos de la vida real en un software a través de un modelo que nos permite a su vez observar posibles comportamientos, para de esta forma poder tomar decisiones respecto a dicho proceso [18, 19] .

Entre sus ventajas evidentes se encuentran la capacidad de evaluar la factibilidad de un proceso antes de construirlo en la vida real por el costo en tiempo y recursos. Dentro del campo de la simulación se encuentran varios paradigmas que resultan útiles al aplicarse en dependencia del nivel de abstracción del proceso. Es esencialmente una técnica de muestreo estadístico controlado, que puede usarse para estudiar el desempeño de sistemas informáticos. En una simulación, se utiliza una computadora para evaluar un modelo numéricamente y recolectar datos para estimar las características del modelo [20]. La simulación no resuelve los problemas por sí misma, sino que ayuda a identificar los problemas relevantes y a evaluar cuantitativamente las soluciones alternativas [5].

1.3.1 Simulación basada en agentes

Como una forma de resolución y debido fundamentalmente al paradigma de agentes, emergen las simulaciones basadas en agentes (ABS), que permite la resolución de problemas en los diferentes niveles de abstracción. Con este fin se utilizan objetos activos que se representan en las entidades del proceso (agentes), que poseen reglas de comportamiento individual. Aunque la misma permite representar todo tipo de proceso, es evidente que su mayor campo de acción consista en explotar al máximo las características de agente que se asocien con procesos en donde los seres humanos sean las entidades básicas del sistema, ya que no posee sentido alguno utilizar esta forma de modelación si las entidades actúan de forma pasiva y reactiva a los eventos del modelo. Por otra parte, podría resultar aún más complicado intentar modelar procesos complejos que contengan esencialmente entidades que sean seres humanos a través del empleo de sistemas dinámicos por el alto nivel de dificultad que podría representar generar ecuaciones que respalden el comportamiento tan diverso de los seres humanos, y no teniendo en cuenta de alguna forma todos los factores que pudieran intervenir [21].

La Simulación Basados en Agentes se refiere a una categoría de modelos computacionales que invoca las acciones dinámicas, las reacciones y los protocolos de intercomunicación entre los agentes en un entorno compartido, para evaluar su diseño y rendimiento y obtener información sobre su comportamiento y propiedades emergentes [22]. Desde el punto de vista de la simulación, la función de un componente individual puede variar desde reglas reactivas muy básicas hasta modelos de comportamiento cognitivos más sofisticado [23].

El objetivo de ABS es modelar sistemas complejos que adoptan un enfoque ascendente a partir de los agentes individuales [24]. Un enfoque concreto de ABS es modelar y simular escenarios realistas con un grupo de agentes autogobernados, ya sea como entidades simples dentro de los fragmentos de códigos de cómputo o como objetos considerablemente inteligentes. Esto posiblemente se considere sinónimo de capacidades de resolución de problemas del ser humano con estados infinitos, creencias, confianzas, decisiones, acciones y respuestas. Adquirir el conocimiento

adecuado del sistema para construir un modelo conceptual y lógico apropiado es una de las tareas más desafiantes de la simulación [25].

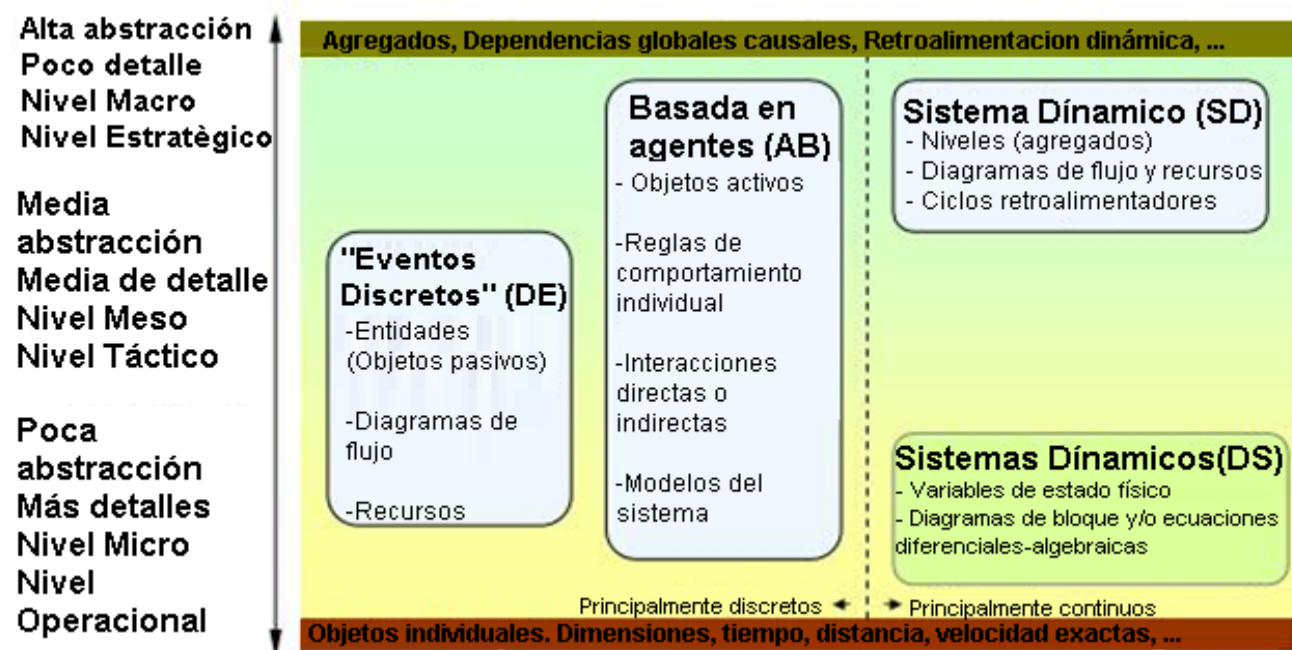


Figura 3: Esquema de Paradigmas de Simulación

1.4 Problema de Máxima Cobertura

Una temática en la que la ABS puede aportar ventajas es en el problema Máxima Cobertura. Entre los problemas de optimización más conocidos se encuentran los problemas de localización y cobertura. Estos toman en cuenta situaciones donde es muy importante la ubicación de instalaciones para suministrar servicios, ya que cubrir la demanda de los usuarios es la principal prioridad [26], por tanto, surge la siguiente problemática: ¿Cómo saber dónde ubicar las instalaciones para satisfacer la mayor demanda de servicios posibles?

En los problemas de localización se encuentran los problemas de cobertura. Estos problemas tienen varias clasificaciones, entre ellas se conocen: problema de localización de máxima cobertura (MCLP, por sus siglas en inglés) [27] y problema de localización de cobertura de conjuntos (SCLP, por sus siglas en inglés) [28]. En ambas variantes una demanda está cubierta si se puede cubrir en un tiempo/distancia predefinido, por una instalación [29].

El problema MCLP fue introducido en 1974 por Church y Reville [22]. En este problema la decisión está basada en buscar la mejor combinación de un número

conocido de instalaciones que tenga un máximo de cobertura sobre las demandas de los nodos. El espacio de soluciones contiene un gran número de combinaciones válidas y esta cantidad depende de la magnitud del problema, total de instalaciones, número de instalaciones a ubicar y total de nodos de demanda [27]. El MCLP ha sido objeto de estudio en numerosos trabajos y desde su primera publicación hasta el 2015, ha tenido más de 1550 citas en la literatura académica [30]. Varias de las aplicaciones de este problema son: la localización de las estaciones de patrullas de la policía [31,32], organización y distribución de ambulancias [33,34], distribución de redes de celulares [35,36], localización de las áreas para el transporte de recursos peligrosos [37], localización de sensores móviles [38], localización de estaciones de bomberos [39] y ubicaciones de cámaras en una red de tráfico urbano [40], entre otras.

1.4.1 Definición del MCLP

El MCLP parte del concepto de cobertura siguiente: un nodo de demanda puede ser cubierto si y solo si está dentro del radio de cobertura una instalación abierta [27]. A continuación, se muestran los parámetros, variables y restricciones del MCLP.

Parámetros:

- i : índice y cantidad de nodos de demanda.
- j : índice y cantidad de nodos de instalación.
- a_i : demanda del nodo i .
- d_{ij} : distancia de un nodo de demanda i a la instalación j .
- p : Cantidad de instalaciones a ubicar.
- S : radio de cobertura de las instalaciones.
- N_i : $\{j | d_{ij} \leq S\}$ es el conjunto de instalaciones cuya distancia al nodo i es menor que S .

Variables:

- X_j : Variable binaria donde el valor 1 significa que la instalación j está abierta. $X_j \in \{0,1\} \ j \in J$
- Y_i : Variable binaria donde el valor 1 significa que el nodo de demanda i está siendo cubierto.

$$Y_i \in \{0,1\} \ i \in I$$

La función objetivo del problema se define como:

$$\text{Maximizar } Z = \sum_{i=1}^n a_i Y_i \quad (1)$$

Maximizar la sumatoria de las demandas de los nodos a_i para aquellos $Y_i=1$.

Sujeto a:

$$\sum_{j=1}^n X_j = p \quad (2)$$

$$Y_i \leq \sum_{j \in N_i} X_j \quad i \in I \quad (3)$$

Donde la restricción (2) establece que la cantidad de instalaciones abiertas debe ser igual que el valor p y la restricción (3) que un nodo de demanda solo puede estar cubierto si se encuentra a una distancia d_{ij} de una instalación abierta menor que S [27]. En la Figura 3 se muestra la representación del problema MCLP, la cual consiste en una cadena binaria donde la cantidad de 1 coincide con el número de instalaciones a abrir. Se puede apreciar como de las instalaciones disponibles se abrirán los siguientes nodos: 2, 3, 4 y 9, donde $p = 4$ [27].

Nodos de Instalaciones								
Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5	Nodo 6	Nodo 7	Nodo 8	Nodo 9
0	1	1	1	0	0	0	0	1

Figura 4: Representación binaria del problema MCLP.

Gráficamente el problema se describe como un conjunto de localizaciones dispersas donde existen puntos que deben dar cobertura a otros puntos. En la Figura 4 se muestra una representación gráfica del MCLP correspondiente a la Figura 1, donde los puntos verdes ($X_j = 1$) dan cobertura a los puntos azules que representan las $Y_i = 1$, los puntos naranjas son aquellos $X_j = 0$ y los puntos rojos representan $Y_i = 0$.

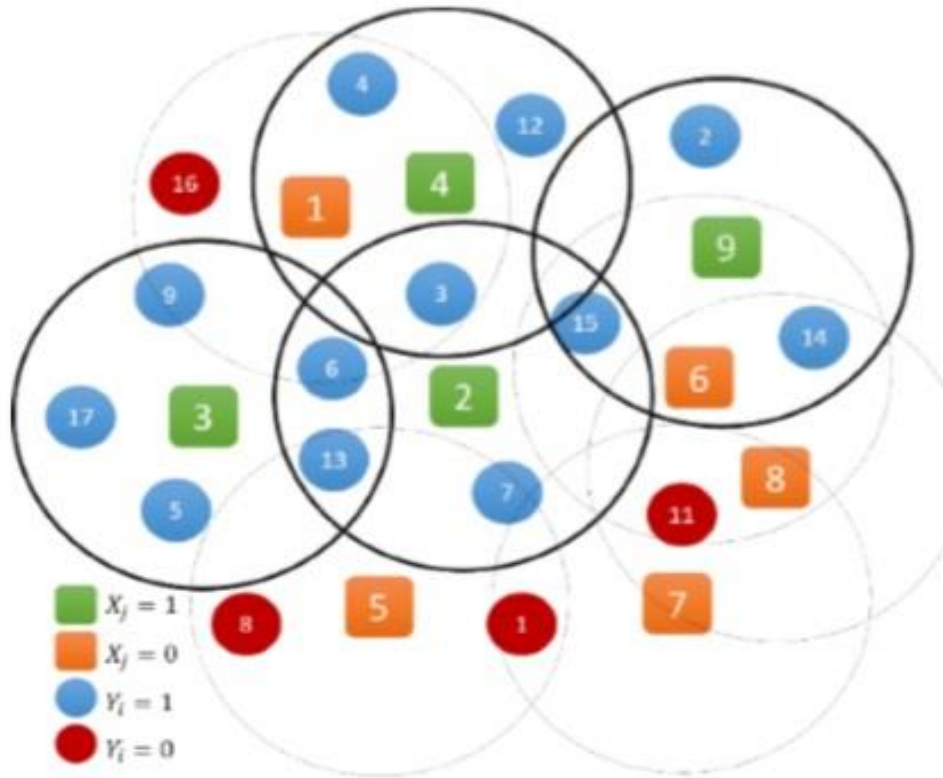


Figura 5: Representación gráfica del problema MCLP.

En este ejemplo, el valor de la función objetivo sería la suma de: $a_2 + a_3 + a_4 + a_5 + a_6 + a_9 + a_{12} + a_{13} + a_{14} + a_{15} + a_{17}$, correspondiente a la demanda de los nodos 2, 3, 4, 5, 6, 9, 12, 13, 14, 15 y 17, los cuales están cubiertos por las instalaciones abiertas 2, 3, 4 y 9.

1.5 Sistema de Información Geográfica

Están definidos por varios autores [41,42] que los sistemas de información geográfica (GIS) es la asociación de software, hardware y herramientas que permiten representar, almacenar y obtener datos geográficos con el fin de resolver problemas complejos que requieran la utilización de información geográfica para representar datos. De esta forma se plantea como requisito la incorporación de GIS a los modelos de simulación planteados para recrear con datos reales los procesos y obtener de esta forma resultados que se asocien con la realidad.

En la Figura 5 se muestra como la información de la geografía es dividida y estructurada en diferentes capas que a su vez contienen información de sus elementos, bajo un sistema de coordenadas que coincide con la geografía real. Esto permitirá

poseer dimensiones de distancia reales en donde existan proporciones de velocidad acordes entre otros parámetros.



Figura 6: Sistema de información geográfica

Existen dos modelos para representar los datos espaciales, ráster y vectorial, siendo común que ambos coexistan en un mismo sistema [43]. A grandes rasgos, el modelo ráster consiste en una matriz rectangular de celdas cuadradas o píxeles usados para representar información relativa a capas temáticas, elevaciones e imágenes satelitales, entre otras [43, 44]. A su vez, el modelo vectorial se basa en el uso de puntos independientes con atributos matemáticos asociados [43, 45], como unidad primitiva, que se agrupan para formar geometrías más complejas. Es utilizado para la representación de aspectos pocos variables y cualitativos, como la representación geométrica de territorios y la altitud [46]. En la actualidad, existen varias entidades que proveen datos espaciales. Una de estas es *Open Street Map* (OSM) [47].

1.6 Patrullaje Policial

Las patrullas policiales juegan un papel importante en el servicio público para responder a incidentes, disuadir y prevenir delitos. El patrullaje policial puede dar una sensación de seguridad a las personas que necesitan protección y desalentar a quienes puedan cometer delitos en la ausencia de una patrulla [48]. Las metas y objetivos del patrullaje policial incluyen prevención de delitos, detención penal,

cumplimiento de la ley, orden y mantenimiento de los servicios públicos y control del tráfico [49].

Normalmente, una ciudad se divide en municipios o alrededores. Cada municipio está además dividido en varias áreas y consejos populares y un oficial al mando para administrar y supervisar las operaciones de patrullaje policial. La policía puede gestionar eficazmente sus operaciones a través del diseño de las áreas y la elección de estrategias de patrullaje de las unidades policiales dentro de esas áreas. Hay tres principales tipos de estrategias de patrullaje para oficiales de patrulla: Patrullaje activo, patrullaje aleatorio, y patrullaje dirigido [54]. En patrulla activo, los oficiales de patrulla deberían aprovechar cada oportunidad para descubrir, detectar, observar e interceptar un evento inusual.

El patrullaje aleatorio significa que las rutas de patrulla deben ser aleatorias y variables [50] para que los comportamientos de patrulla no sean predichos por criminales potenciales. En patrullaje dirigido, patrulleros dan más esfuerzos para los puntos calientes del crimen para que puedan responder rápidamente y reducir los crímenes en puntos de acceso. En la práctica, los oficiales de patrulla pueden elegir una estrategia o combinarlos para acomodar las condiciones específicas en su área [49]. Otro principio de patrulla importante es la integridad del golpe, se espera que el oficial permanezca dentro del área de patrullaje asignado. La integridad del golpe puede ser absoluta o relativa, dependiendo del número de unidades de patrulla asignadas y el tamaño del área de la patrulla y la actividad dentro del área [49]. Por integridad absoluta, los oficiales de patrulla deben permanecer en el área de patrullaje asignado en todo momento. En patrullaje relativo, los oficiales de patrulla permanecen dentro del área de patrullaje y solo pueden dejarlo por una buena razón o responder a una llamada de servicio en otra área de patrullaje [49] si y solo si es autorizado por el oficial de mando a cargo. Cuando hay un incidente, se utiliza el principio de que “El coche de policía más cercano responde a la llamada”, para que llegue al incidente en un rango de 5 a 10min. La variante operativa es una de las importantes estrategias de asignación de recursos del departamento de policía. El despliegue de la fuerza de patrulla a través de los municipios es un método de gestión estándar para mejorar la capacidad de disuasión de la fuerza de patrulla uniformada. [49]. La variante operativa consiste en reubicar

semanalmente las patrullas en determinadas áreas teniendo en cuenta la situación operativa del área (cantidad de delitos, cantidad de emergencias, etc.) y la situación demográfica.

1.7 Plataformas de agentes

A continuación, se realiza un estudio de algunas de las plataformas de agentes actuales escogidas entre las más reconocidas según FIPA [55].

MASON

MASON (Stands for Multi-Agent Simulator Of Neighborhoods... or Networks... or something...) es una plataforma de simulación basada en agente, de eventos rápidos y discretos, diseñado para ser la base para grandes simulaciones en Java de propósito personalizado, y también para proporcionar funcionalidad más que suficiente para muchas necesidades de simulación de peso ligero. Contiene una biblioteca de modelos y un conjunto opcional de herramientas de visualización en 2D y 3D [56,47].

Constituye un esfuerzo conjunto entre el Laboratorio de Computación Evolutiva de la Universidad George Mason y el Centro de Complejidad Social. Sus diseñadores son: Sean Luke, Gabriel Catalin Balan, Keith Sullivan y Liviu Panait, con ayuda de Claudio Cioffi Revilla, Sean Paus, Keith Sullivan, Daniel Kuebrich, Joey Harrison y Ankur Desai [56].

Características:

- 100% Java (1.3 o superior).
- Rápido, portátil y bastante pequeño.
- Los modelos son completamente independientes de la visualización, que se pueden agregar, quitar o cambiar en cualquier momento.
- Los modelos pueden ser controlados y recuperados, y migrados dinámicamente a través de las plataformas.
- Puede producir resultados idénticos entre plataformas.
- Los modelos son independientes y pueden ejecutarse dentro de otros marcos de trabajo y aplicaciones Java.
- Visualización 2D y 3D.
- Puede generar capturas de pantalla, películas QuickTime, gráficos.

Objetivos de diseño:

- Posibilidad de ejecutar muchos tipos de simulaciones
- Sistemas sociales complejos
- Modelado físico
- Inteligencia artificial
- Eficiente con grandes números de agentes
- Alto grado de modularidad y flexibilidad
- Núcleo pequeño y fácil de entender
- Separar las herramientas de visualización
- *Check-pointing* y recuperación

Las bibliotecas se proporcionan para visualizar en 2D y en 3D (utilizando Java3D), manipular el modelo gráficamente, tomar capturas de pantalla y generar películas (utilizando *Java Media Framework*).

GeoMason:

Como parte de la integración con GIS, MASON cuenta con una extensión opcional llamada GeoMason la cual añade soporte para datos geoespaciales y se publica bajo la Licencia Libre Académica.

JADE

JADE (*Java Agent DEvelopment Framework*) es un marco de trabajo de software totalmente implementado en el lenguaje Java. Simplifica la implementación de sistemas multi-agente a través de un *middleware* que cumple con las especificaciones FIPA [58] y mediante un conjunto de herramientas gráficas que soportan las fases de depuración e implementación. Un sistema basado en JADE puede distribuirse entre máquinas (que ni siquiera necesitan compartir el mismo sistema operativo) y la configuración se puede controlar a través de una interfaz gráfica de usuario (GUI, *Graphical User Interface*) remota. La configuración puede cambiarse incluso en tiempo de ejecución moviendo agentes de una máquina a otra, según sea necesario. JADE está completamente implementado en lenguaje Java y el requisito mínimo del sistema es la versión 5 de JAVA [59] (el entorno de tiempo de ejecución o el JDK) [60].

JADE es software libre y es distribuido por Telecom Italia, titular de los derechos de autor, en código abierto bajo los términos y condiciones de la licencia LGPL [61] (Licencia Pública General Menor Versión 2) [62].

JADE está entre las plataformas más conocidas y utilizadas debido a las funcionalidades que permite desarrollar. Entre las mismas, facilita construir sistemas de agentes para el manejo de información y recursos a través de redes de computadoras, bajo la complacencia de las especificaciones FIPA, para sistemas multi-agente interoperables [62].

JADE es un *middleware* que consiste en un entorno de ejecución, donde los agentes pueden vivir, y estar activos en determinado host, antes de que otros puedan ser ejecutados en el mismo. Ofrece una biblioteca de clases, que los programadores pueden usar (directamente o modificando las mismas) para desarrollar los agentes, así como una suite de herramientas gráficas, que permite administrar y monitorear la actividad de los agentes que se estén ejecutando [63].

La plataforma puede ser distribuida a través de sistemas heterogéneos, y brinda soporte a agentes móviles. La arquitectura de la comunicación ofrece una flexible y eficiente mensajería, donde JADE crea y administra una cola de mensajes ACL privada para cada agente. Todo el estándar FIPA de comunicaciones ha sido implementado: protocolos de interacción, ACL, lenguaje contenido, esquemas, ontologías y protocolos de transportación. El mecanismo para el transporte de mensajes utiliza Java RMI [64].

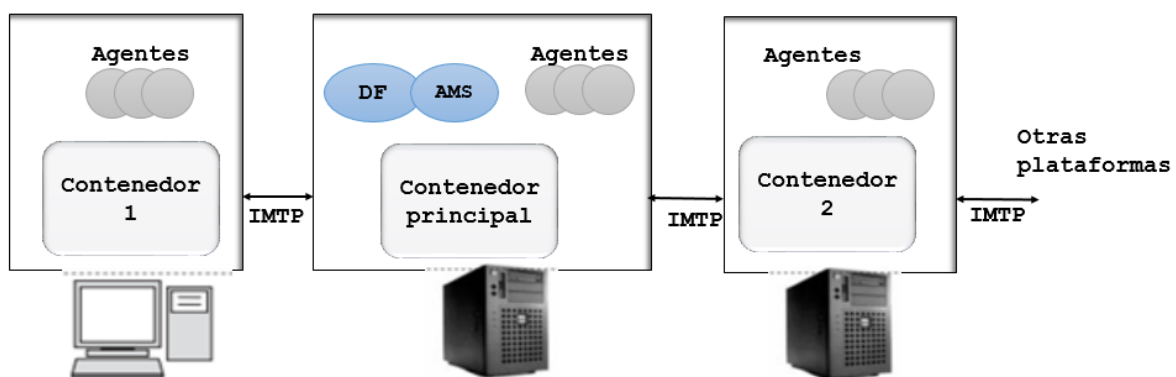


Figura 7: Arquitectura de plataforma JADE

La Figura 6 muestra los elementos principales que integran la arquitectura de JADE. Dicha plataforma está compuesta por contenedores de agentes, que pueden estar

distribuidos en una red de computadoras. Los agentes viven en estos contenedores, que son procesos Java que provee el entorno de ejecución JADE, y todos los servicios necesarios para el *hosting* y ejecución de agentes. Hay un contenedor especial (Principal), que es el primero en ser ejecutado, y entre sus funcionalidades está registrar las referencias de otros contenedores que componen la plataforma, así como todos los agentes, incluyendo el estado actual de los mismos y localización, además del control de dos agentes especiales, uno denominado AMS, encargado de supervisar la plataforma y el DF, quien posee una lista de servicios registrados por defectos que pueden ser utilizados por los agentes [65].

JADEX

Distintas plataformas dan soporte al desarrollo de aplicaciones basadas en sistemas multi-agente, identificándose dos grupos: por un lado, aquellas que obedecen las especificaciones FIPA, destacándose JADE, mientras que, en otro, se encuentran las plataformas centradas en el razonamiento, teniendo como base la arquitectura BDI [66]. Estas diferencias, constituyen la principal motivación de la plataforma Jadex BDI, la cual cubre las particularidades de ambos grupos, en un único entorno, cumpliendo con los estándares FIPA, así como dando soporte al razonamiento BDI [67].

Este proyecto nace en el 2002 como parte de investigaciones entre el Departamento de ciencias de la computación de la Universidad de Hamburgo (Alemania) y el MedPage [66]. JADEX es un marco de trabajo basado en JADE, para ingeniería de razonamiento, bajo especificaciones FIPA, que facilita el desarrollo de agentes a través de los lenguajes de programación Java y de marcado XML. Es distribuido bajo licencia GNU-LGPL [68].

En esta plataforma, no se introduce un nuevo lenguaje de programación, lo que deviene en una gran ventaja, dado que los agentes pueden ser implementados bajo los conocimientos del estado del arte, así como los principales entornos de desarrollo integrado como Eclipse e IDEA. Otra ventaja, radica en que los agentes representan componentes activos con capacidades de razonamiento individual, lo que se traduce en el hecho de que pueden exhibir comportamientos reactivos y proactivos [68].

JADEX incorpora el modelo BDI en un agente JADE mediante creencias, metas y planes. Las creencias pueden ser cualquier tipo de objeto Java y ser almacenadas en

una base de creencias. Las metas representan motivaciones concretas (por ejemplo, estados a lograr) que tienen influencia en el comportamiento del agente y, para lograrlas, se ejecutan planes implementados como clases Java. Todo este estado de los agentes, es capturado en un Archivo de Definición de Agente (ADF). El ADF es un fichero XML donde los desarrolladores definen creencias iniciales y metas mediante expresiones establecidas para tales propósitos [69]. El lenguaje para estas expresiones es Java, con una extensión de constructores OQL [70].

NetLogo

NetLogo es un entorno de modelado programable multi-agente para simular fenómenos naturales y sociales. Es utilizado por decenas de miles de estudiantes, profesores e investigadores de todo el mundo. Fue escrito por Uri Wilensky en 1999 y ha estado en continuo desarrollo desde entonces [71].

Es particularmente adecuado para modelar sistemas complejos que se desarrollan con el tiempo. Los modeladores pueden dar instrucciones a cientos o miles de "agentes" que funcionan de forma independiente. Esto hace posible explorar la conexión entre el comportamiento micro-nivel de los individuos y los patrones macro-nivel que emergen de su interacción [71].

NetLogo permite abrir simulaciones explorando su comportamiento bajo diversas condiciones. También es un entorno de creación que permite a los estudiantes, profesores y desarrolladores de planes de estudio crear sus propios modelos. Es bastante simple para estudiantes y profesores, pero lo suficientemente avanzado como para ser una herramienta poderosa para los investigadores en muchos campos [71].

Tiene una amplia documentación y tutoriales. También viene con la Biblioteca de Modelos, una gran colección de simulaciones pre-escritas que se pueden utilizar y modificar. Estas simulaciones abordan áreas de contenido en las ciencias naturales y sociales, incluyendo biología y medicina, física y química, matemáticas e informática, y economía y psicología social. Existen varios planes de estudio basados en modelos que utilizan NetLogo y otros están en desarrollo [71].

NetLogo es la próxima generación de la serie de lenguajes de modelado multi-agente, incluyendo StarLogo y StarLogoT. NetLogo se ejecuta en la máquina virtual Java, por lo que funciona en todas las plataformas principales (Mac, Windows, Linux, et al). Se

ejecuta como una aplicación de escritorio. También se admite la operación en línea de comandos [71].

Características:

- Sistema:
 - Libre, código abierto
 - Portabilidad: se ejecuta en Mac, Windows, Linux y otros
 - Soporte de conjunto de caracteres internacional
- Programación:
 - Totalmente programable
 - Sintaxis accesible
 - Los agentes móviles (tortugas) se mueven sobre una cuadrícula de agentes estacionarios (parches)
 - Los agentes de enlace conectan a las tortugas para hacer redes, gráficos y agregados
 - Amplio vocabulario de primitivas de lenguaje incorporadas
 - Valores de funciones de primera clase (también conocidos como procedimientos anónimos o cierres)
 - Las ejecuciones son reproducibles en multiplataforma
- Ambiente:
 - Centro de mando para la interacción directa
 - Interfaz constructor, botones, controles deslizantes, interruptores, selectores, monitores, cuadros de texto, notas, área de salida
 - Pestaña de información para anotar su modelo con texto e imágenes con formato
 - HubNet: simulaciones participativas utilizando dispositivos en red
 - Monitores de agente para inspeccionar y controlar agentes
 - Funciones de exportación e importación (datos de exportación, guardar y restaurar el estado del modelo, hacer una película)
 - BehaviorSpace, una herramienta de código abierto utilizada para recopilar datos de múltiples ejecuciones paralelas de un modelo
 - System Dynamics Modeler

- NetLogo 3D para modelar mundos 3D
- El modo Headless permite realizar lotes desde la línea de comandos
- Visualización:
 - Parcelas de líneas, barras y dispersiones
 - El control deslizante de velocidad le permite avanzar rápidamente su modelo o verlo en cámara lenta
 - Ver su modelo en 2D o 3D
 - Formas vectoriales escalables y giratorias

RePast

Repast es un conjunto de herramientas de simulación y modelado basado en agentes, libre y de código abierto. Actualmente están disponibles tres plataformas Repast, cada una de las cuales tiene las mismas características principales pero un entorno diferente. Para estas características. *Repast Symphony (Repast S)* extiende la cartera de Repast ofreciendo un nuevo enfoque para el desarrollo y la ejecución de la simulación [72].

RePast es un marco de trabajo para la simulación basada en agentes creado por *Social Science Research Computing* de la Universidad de Chicago. Proporciona una biblioteca integrada de clases para crear, ejecutar, mostrar y recopilar datos de una simulación basada en agentes [73].

El modelo *RePast* típico contiene un conjunto de agentes. Estos agentes pueden o no ser homogéneos, o tal vez estos agentes están dispuestos en una jerarquía (una empresa y sus empleados, por ejemplo). Independientemente de su composición, cada agente tiene cierto comportamiento, las interacciones de las cuales un modelador está interesado en explorar [74].

El tiempo de ejecución de *Repast S* está diseñado para incluir funciones avanzadas de almacenamiento, visualización y activación de comportamiento del agente, así como nuevas instalaciones para el análisis y la presentación de datos.

GAMA

GAMA es un entorno de desarrollo de simulación y modelado para construir simulaciones basadas en agentes.

Entre sus características principales se encuentran:

- Múltiples dominios de aplicación: Garantiza la posibilidad de utilizar GAMA para cualquier dominio de aplicación que desee.
- Lenguaje de alto nivel e intuitivo basado en agentes: Permite escribir sus modelos fácilmente usando GAML como lenguaje.
- GIS y modelos dirigidos por datos: Ofrece instanciar agentes de cualquier conjunto de datos, incluyendo datos GIS, y ejecutar simulaciones a gran escala (hasta millones de agentes).
- Interfaz de usuario declarativa: Se pueden declarar interfaces que soporten inspecciones profundas en agentes, paneles de acción controlados por el usuario, pantallas multidimensionales 2D / 3D y aspectos de agente.

Consiste en una sola aplicación que se basa en la arquitectura RCP proporcionada por Eclipse [76]. Dentro de este único software de aplicación, a menudo denominado plataforma, los usuarios pueden realizar, sin necesidad de software adicional de terceros, la mayoría de las actividades relacionadas con el modelado y la simulación, es decir, edición de modelos y simulación, visualización y exploración utilizando herramientas dedicadas [77].

Se logra, en primer lugar, proporcionar algunos antecedentes sobre las nociones importantes que se encuentran en toda la plataforma, especialmente las del espacio de trabajo y los proyectos y explicar cómo organizar y navegar a través de los modelos. Luego una vista a la edición de modelos y sus diversas herramientas y componentes (editores dedicados y herramientas relacionadas, por supuesto, pero también validadores). Finalmente, se muestra cómo ejecutar experimentos en estos modelos y el soporte que la interfaz de usuario puede proporcionar a los usuarios en esta tarea [78].

En Resumen, GAMA es una plataforma de desarrollo novedosa que provee de recursos amplios para llevar a cabo simulaciones sociales basada en agentes, incluye integración nativa con archivos de datos espaciales para su posterior interacción con agentes sociales. No presenta mecanismos de importar archivos tropos que utiliza el lenguaje de modelado i*. Es una tecnología que cuenta con un respaldo amplio de documentación en cuanto a guías, tutoriales, fórum, etc. Su implementación se rige por un lenguaje de programación propio, relativamente similar a Java. Cuenta con el IDE

eclipse *gama-plataform*, el cual brinda métodos de detección de errores de código, ejemplos, etc. Sin duda, esta constituye una propuesta conveniente como base de la arquitectura para el desarrollo de la herramienta en cuestión.

1.8 Comparación entre plataformas

Luego de hacer una síntesis de los detalles más importantes de algunas plataformas de agentes disponibles, la Tabla 1: Comparación de plataformas de agentes, muestra una comparación resumen teniendo en cuenta las características necesarias que más se adecuan a la problemática.

Tabla 1: Comparación de plataformas de agentes

Plataformas	Lenguaje	GIS	Licencia	Soportes para el usuario
MASON	Java	Sí	Académica libre	API, tutoriales, documentación, listas de correo y ejemplos
JADE	Java	No	LGPL	Documentación, tutoriales y ejemplos
JADEX	Java	No	GNU- LGPL	Documentación, tutoriales y ejemplos
NetLogo	NetLogo	Sí	GPL	Documentación, FAQ, tutoriales y listas de correo
RePast	Java, C++, C#	Sí	BSD	Lista de correo, tutoriales, FAQ y ejemplos
GAMA	GAML	Sí	BSD	Videos tutoriales y ejemplos

Según la problemática planteada con anterioridad, es posible afirmar que las plataformas recomendables a seleccionar para implementar la propuesta de solución son MASON, NetLogo, RePast y GAMA. Dado que la curva de aprendizaje de estas plataformas por lo general suele ser elevada, y se cuenta con recursos limitados durante la fase de desarrollo, se seleccionó MASON debido a que se contaba con experiencias previas con este marco de trabajo.

1.9 Arquitectura de MASON

Tras la elección de MASON como plataforma de agentes y marco de trabajo, se realizará un estudio más detallado. Se expondrán los elementos de la arquitectura de MASON que se consideran más importante para el desarrollo de la herramienta destacado patrones que usa.

1.9.1 Estilos y patrones arquitectónicos

Estilo llamada y retorno: Patrón Modelo Vista Controlador (MVC, *Model-View-Controller*).

En la Figura 7, que provee el sitio web oficial de MASON se ve una evidente separación de los modelos y los controladores.

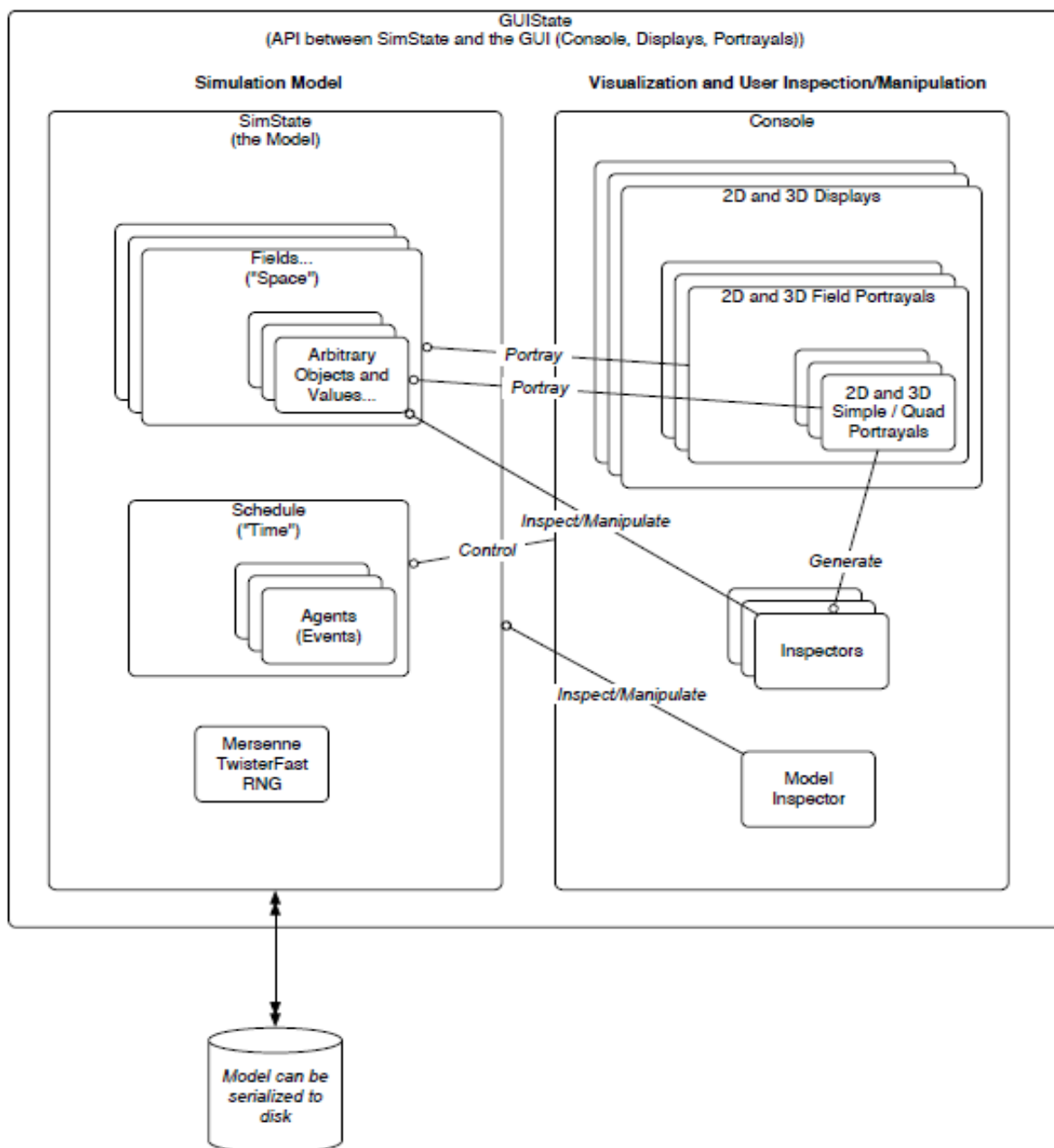


Figura 8: Arquitectura MASON, patrón MVC

1.9.2 Vista general de la arquitectura

MASON se divide en dos partes. La primera parte es el modelo, y la segunda parte es la visualización, 2D o en 3D. Excepto cuando se elige tener objetos de modelo que se muestren, el modelo y la visualización están totalmente separados. Esto significa que el modelo se puede ejecutar sin visualización; ejecutar con visualización de diferentes tipos; y tener su visualización cambiada, añadida o eliminada en cualquier momento. El modelo también puede ser *checkpointed*, lo que significa que puede ser congelado y escrito en el disco, para descongelarse y continuar incluso en otro tipo de equipo. En la Figura 8 se muestra una visión general de la arquitectura MASON.

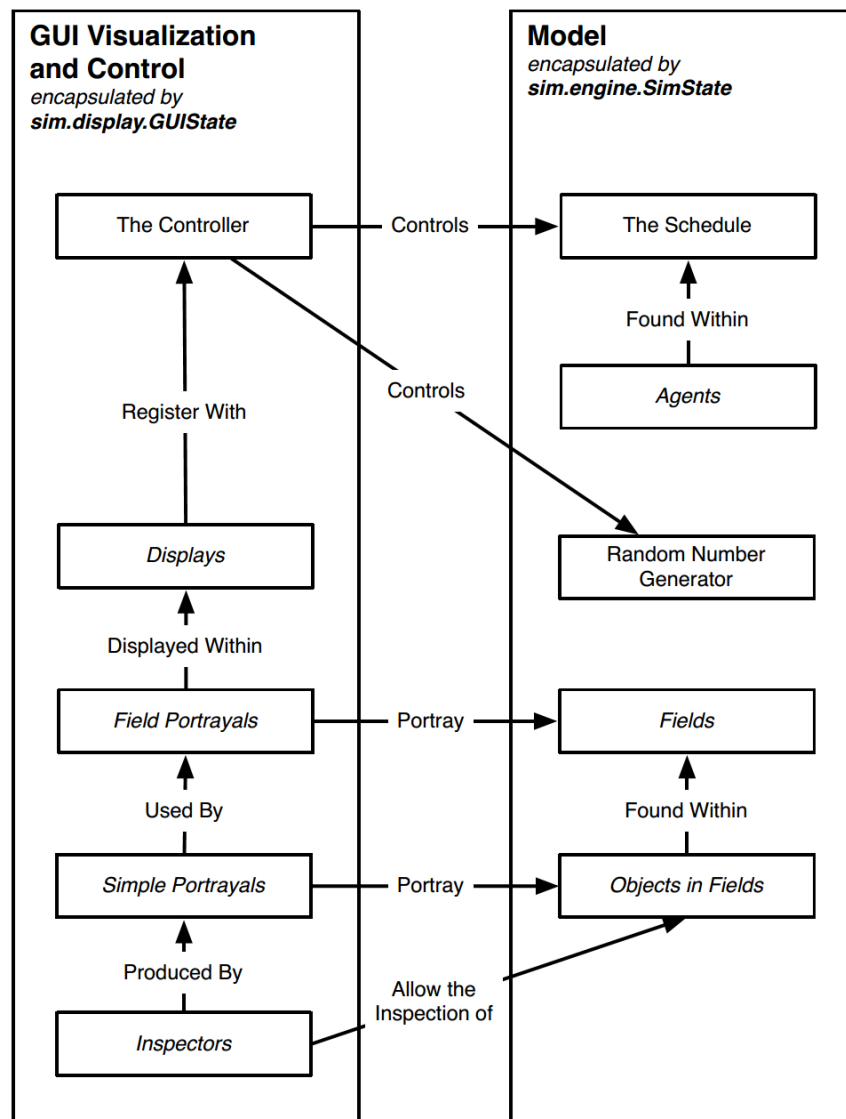


Figura 9: Relaciones primarias entre el modelo y la GUI visualización / control

1.9.3 Vista detallada de la arquitectura

El modelo de simulación está totalmente encapsulado dentro de una sola clase: una subclase de *sim.engine.SimState*. MASON creará una única instancia de su subclase para mantener el modelo. El propósito de esta clase es darle un lugar para almacenar todos y cada uno de los elementos que considere necesarios para su simulación.

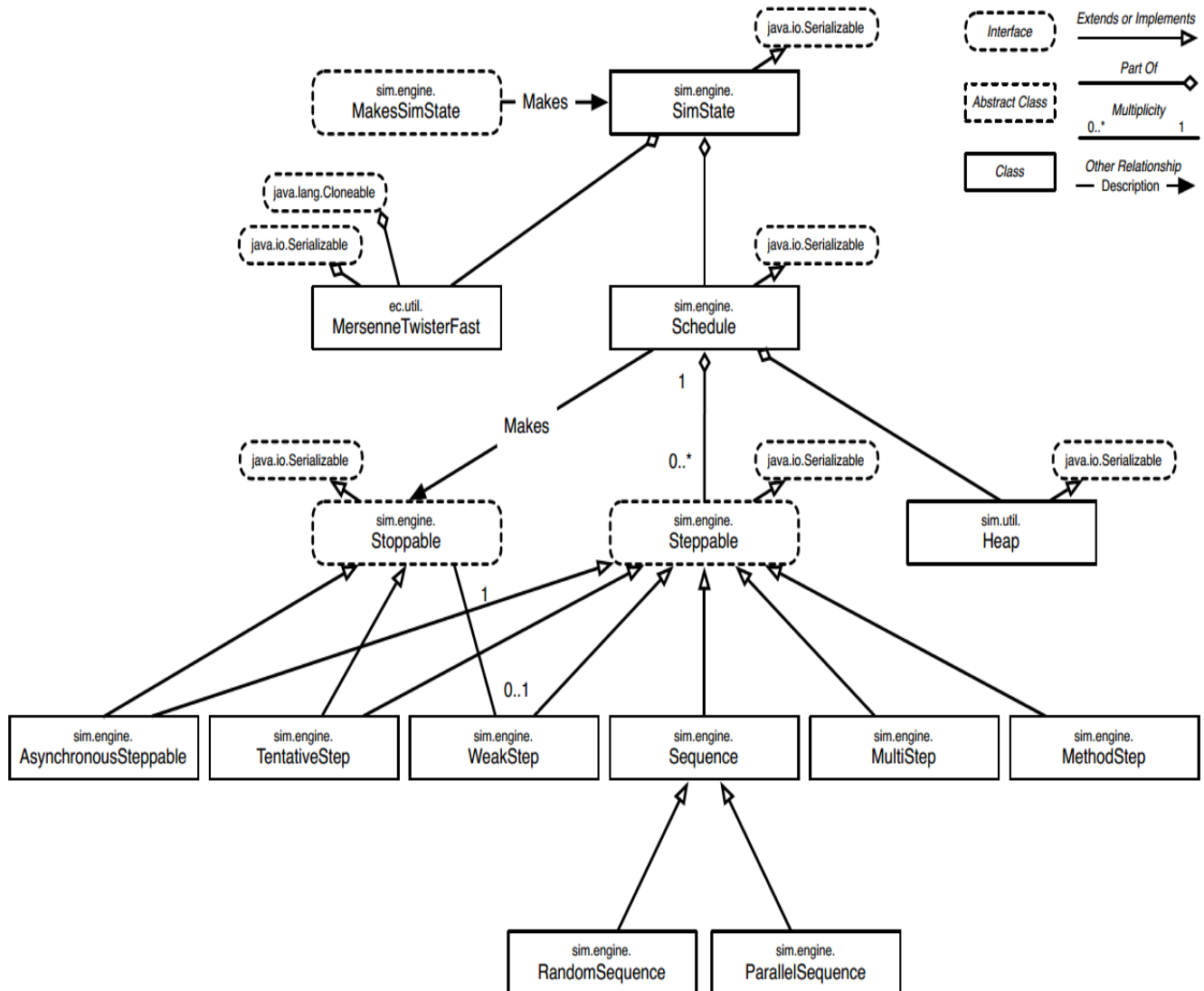


Figura 10: Diagrama de la simulación de núcleo de MASON y de la programación de eventos discretos. MASON puede serializar todo el modelo a un archivo de punto de control, lo que significa que puede congelar la simulación en el tiempo y guardarlo en su totalidad en el archivo. Puede reiniciar el modelo desde este archivo, incluso en un equipo diferente o en un sistema operativo, o bajo una facilidad de visualización, y la simulación

continuará donde lo dejó sin variación alguna. Para que esto sea posible, todos los objetos del modelo MASON son `java.io.Serializable`, lo que significa que pueden ser (en gran parte) automáticamente escritos o leídos de un flujo. Esto está descrito en la Figura 9.

Network: La Figura 10 muestra el paquete `sim.field.network`, el cual consiste en la clase `sim.field.network.Network`, que contiene el gráfico (o red) y la clase `sim.field.network.Edge`.

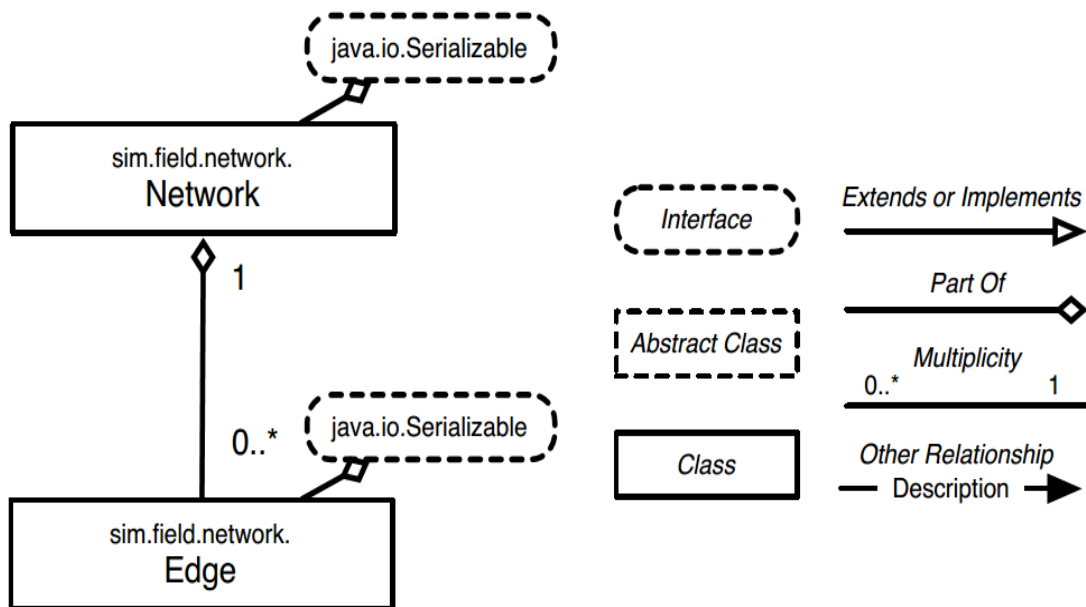


Figura 11: Diagrama del paquete Network de MASON

Modelo y control: En la Figura 11 se muestra la separación clara del modelo y control.

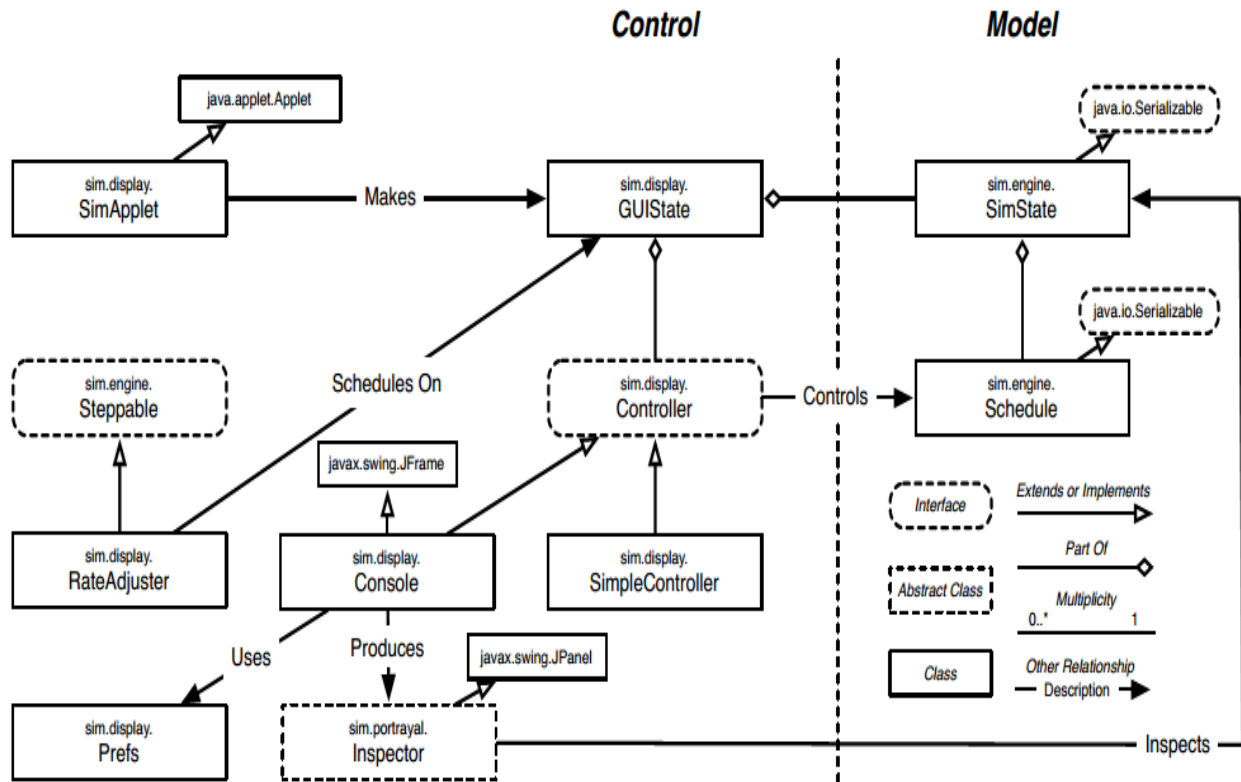


Figura 12: Diagrama del código de control de GUI de MASON

Los inspectores son widget GUI que permiten al usuario inspeccionar, rastrear, trazar un gráfico y modificar un objeto de modelo, un valor específico o la propiedad de un objeto. Los inspectores son subclases de la clase `sim.portrayal.Inspector`, y con la excepción de `sim.portrayal.SimpleInspector`, todos ellos se encuentran en el paquete `sim.portrayal.inspector`. La Figura 12 muestra un diagrama de las clases de Inspector principales. Los inspectores son generalmente producidos por los aparatos o por otros Inspectores, y su trabajo es inspeccionar los objetos en el modelo según sea necesario.

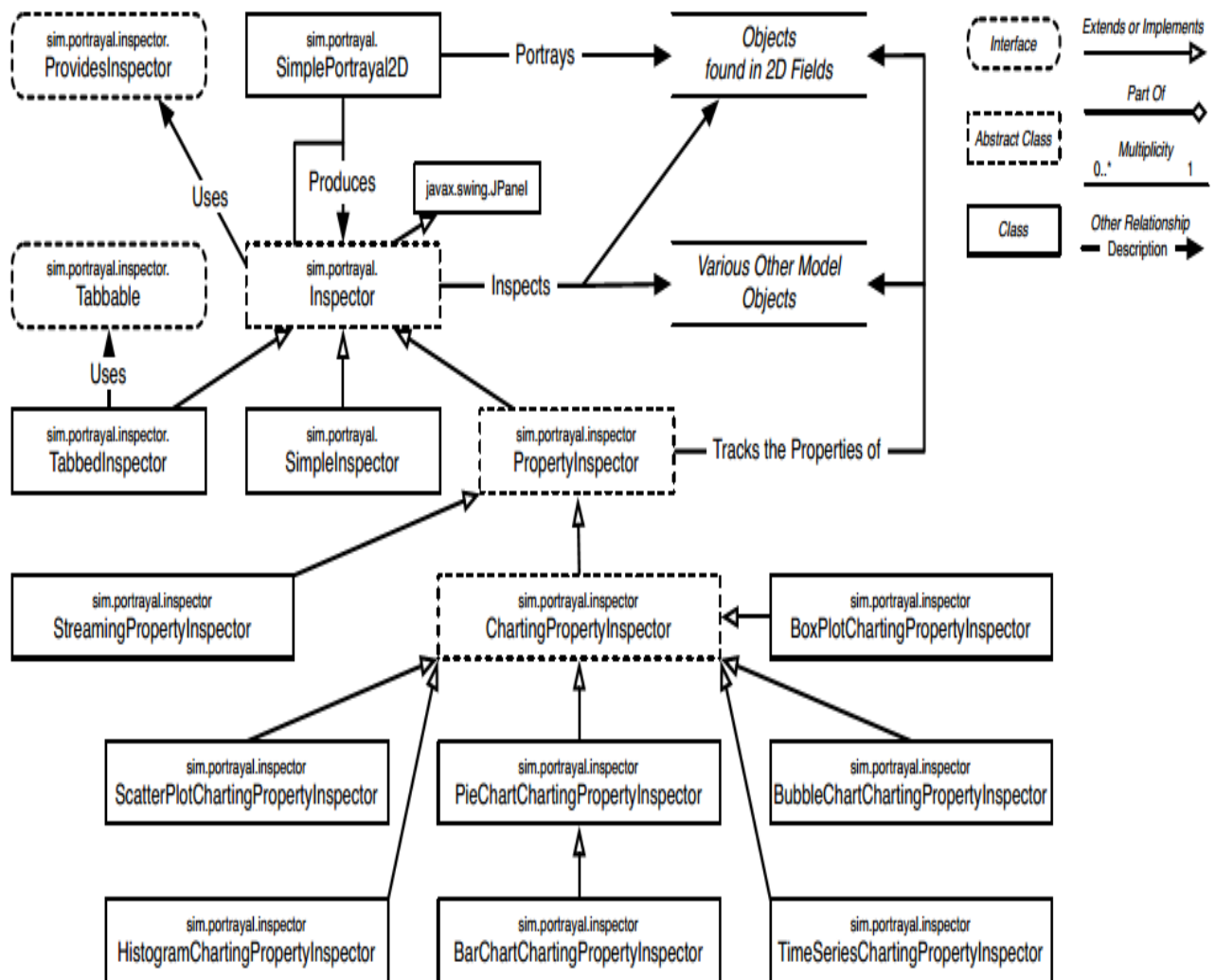


Figura 13: Instalación Inspector de MASON

1.10 Conclusiones parciales

- Se logró identificar y documentar lo referente a los procesos de simulación basada en agentes.
- Se evidenció que el problema de máxima cobertura, toma en cuenta situaciones donde es esencial la ubicación de las instalaciones para brindar un mejor servicio
- Para la captura, almacenamiento, manipulación y análisis de la información geográfica se pueden utilizar los GIS.
- La distribución espacial y la asignación de vehículos de policía a los distritos afectan el tiempo de respuesta promedio y la carga de trabajo de las patrullas.

- La variante operativa es una estrategia eficaz en donde se reubican las patrullas en determinadas áreas según la cantidad de delitos.
- Se realizó una caracterización de las herramientas de simulación basada en agentes.
- Se mantiene Mason como plataforma para la continuidad de la simulación.
- Se analizó la arquitectura de MASON para un análisis más profundo.

Capítulo 2: Solución Propuesta

2.1 Introducción

En el presente capítulo se documenta toda la propuesta de solución. Se exponen los elementos que forman parte del dominio, la relación entre las entidades y las reglas del dominio. Se presenta los diagramas en Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) necesarios para explicar el flujo de las actividades que forman parte de la arquitectura de la herramienta de simulación, como los casos de uso, diagramas de clases. Se muestra además la vista de arquitectura y los patrones utilizados durante el desarrollo de la herramienta.

2.2 Propuesta inicial del modelo matemático

A continuación, se realizará una descripción del modelo matemático de la variante que se propone en el trabajo: el problema de máxima cobertura dinámico con tipos de instalaciones (VMT-DMCLP). Dicho modelo es una generalización del DMCLP, donde existen diferentes tipos de instalaciones que se definen a partir de su radio de cobertura y los nodos de demandas pueden ser atendidos por cualquier instalación, por lo que se asume que se está brindando un solo servicio. Se toma en cuenta que la disponibilidad de los niveles de cobertura de las instalaciones puede variar con respecto al tiempo, de la misma manera que la demanda de los nodos. Esto permite modelar situaciones donde, por razones estratégicas, hay niveles de cobertura de las instalaciones que no van a operar en un periodo de tiempo específico, o dejan de existir a lo largo del tiempo.

Esta propuesta puede ser ampliamente aplicada en casos como: la localización de antenas de wi-fi con diferentes niveles de alcance los cuales varían en el tiempo. También, la localización de distintos tipos de fuerzas policiales para dar vigilancia o ambulancias para dar atención médica, los cuales varían la disponibilidad en distintos horarios. A continuación, se muestra la formulación matemática de la propuesta:

Parámetros:

- i ,: índice y conjunto de nodos de demanda.
- j ,: índice y conjunto de localizaciones(en 1 loc. Pueden estar varias instalaciones).
- t , T : índice y conjunto de periodos de tiempo.

- k , K : índice y conjunto de tipo de instalación (en este caso =1).
- a_{it} : demanda del nodo i en el periodo t .
- S_{tk} : radio de cobertura en el periodo t del tipo k .
- C_{tk} : capacidad del tipo k en el periodo t .
- d_{ij} : distancia de la demanda a la instalación.
- L_{jt} : mínimo de instalaciones a abrir en la ubicación j en el periodo t .
- H_{jt} : máximo de instalaciones a abrir en la ubicación j en el periodo t .
- F_{jtk} : 1 si la localización j en el periodo t es fija para el tipo k {0,1}
- W_{jtk} : 1 si la localización j en el periodo t del tipo k está disponible {0,1}
- p_k : cantidad de instalaciones de tipo k a ubicar.
- N_{it} : $\{j | d_{ij} \leq S_{tk} \text{ y } W_{jtk} = 1\}$ es el conjunto de instalaciones de tipo k en el periodo t cuya distancia al nodo i es menor que S .

Variables:

- X_{jtk} : Variable binaria que para el valor 1 significa que la instalación j en el periodo t está abierta de tipo k y 0 lo contrario.

$$X_{jtk} \in \{0; 1\} \quad j \in J, \quad t \in T$$

- Y_{it} : Variable binaria que para el valor 1 significa que el nodo de demanda i en el periodo t está cubierto y 0 lo contrario.

$$Y_{it} \in \{0; 1\} \quad i \in I, \quad t \in T$$

- D_{jtk} : Demanda total asignada a la instalación j de tipo k en el periodo t .

$$D_{jtk} \in \{0; 1\} \quad i \in I, \quad t \in T, \quad k \in K$$

La función objetivo del problema es:

$$\text{Maximizar } Z = \sum_{i=1}^I \sum_{t=1}^T a_{it} Y_{it} \quad (1)$$

Sujeto a:

$$Y_{it} \leq \sum_{j \in N_{it}} \sum_{k \in K} X_{jtk} \quad \forall i \in I, \quad \forall t \in T \quad (2)$$

$$\sum_{t \in T} \sum_{j \in J} X_{jtk} = p_k, \quad \forall k \in K \quad (3)$$

$$\sum_{k \in K} p_k \leq J \quad (4)$$

$$\sum_{j \in J} \sum_{t \in T} W_{jtk} > p_k, \quad \forall k \in K \quad (5)$$

$$\sum_{t \in T} \sum_{j \in J} W_{jtk} \geq 1 \quad k \in K, \quad \forall j \in J \quad (6)$$

$$\sum_{k \in K} X_{jtk} \leq 1 \quad j \in J, \quad \forall t \in T \quad (7)$$

La restricción (2) muestra que un nodo de demanda puede ser cubierto por una instalación abierta si esta pertenece al conjunto $Nitk$.

La restricción (3) establece que la cantidad de instalaciones abiertas de tipo k debe ser igual al valor pk .

La restricción (4) muestra que la suma de las instalaciones de tipo k que se deben abrir debe ser menor o igual que el total de instalaciones disponibles.

La restricción (5) establece que el total de instalaciones de tipo k que existen debe ser mayor que la cantidad de instalaciones de tipo k que se van a abrir para todos los periodos de tiempo.

La restricción (6) establece que todas las instalaciones deben ser de al menos un tipo en algún periodo de tiempo.

La restricción (7) establece que una instalación j solo puede ser abierta para un tipo k . Se debe notar que el modelo propuesto es una generalización del MCLP, ya que para $T = 1$ y $K = 1$, es equivalente al modelo MCLP.

2.2.1 Distancia mínima entre dos puntos

Considerando que se quieren optimizar las soluciones de MCLP sobre un GIS, surge otro problema clásico como el de menor distancia entre dos puntos en el espacio, ya que se espera que la distancia a recorrer por las patrullas hacia las emergencias sea la menor posible. Por lo tanto, no podemos dejar de lado la utilización de algoritmos como Dijkstra y A*, por solo mencionar algunos, pero la gran pregunta reside ahora en qué algoritmo utilizar en esta situación. Tengamos en cuenta que se está extrayendo un grafo bidireccional (en su mayor parte) de las calles de Centro Habana por las que los vehículos van a moverse. Para elegir que algoritmo usar, nos auxiliamos en el trabajo: Una Mirada al Análisis de Redes de Transporte en Cuba, desde el Punto de Vista de los Datos [78], en el que se llega a la conclusión (a grandes rasgos) que dadas las características de las redes viales de Cuba (en este caso de Centro de Habana), las complejidades de los algoritmos de distancia mínima, y las capacidades de cómputo de las computadoras modernas, la eficiencia de los algoritmos arroja resultados muy similares y es recomendado cualquiera teniendo en cuenta las especificidades de la

situación. Considerando esto, y que MASON trae por defecto una implementación de A*, se decidió utilizar este para calcular la distancia mínima entre dos puntos del mapa.

2.3 Modelo del Dominio

En la Figura 13 se muestra el modelo del dominio donde se representan los principales conceptos a tratar para llevar a cabo el modelado del negocio, se encuentran varias relaciones, así como la multiplicidad de las mismas. Para el entendimiento de estas entidades en la Tabla 2 se argumentarán cada uno de sus conceptos.

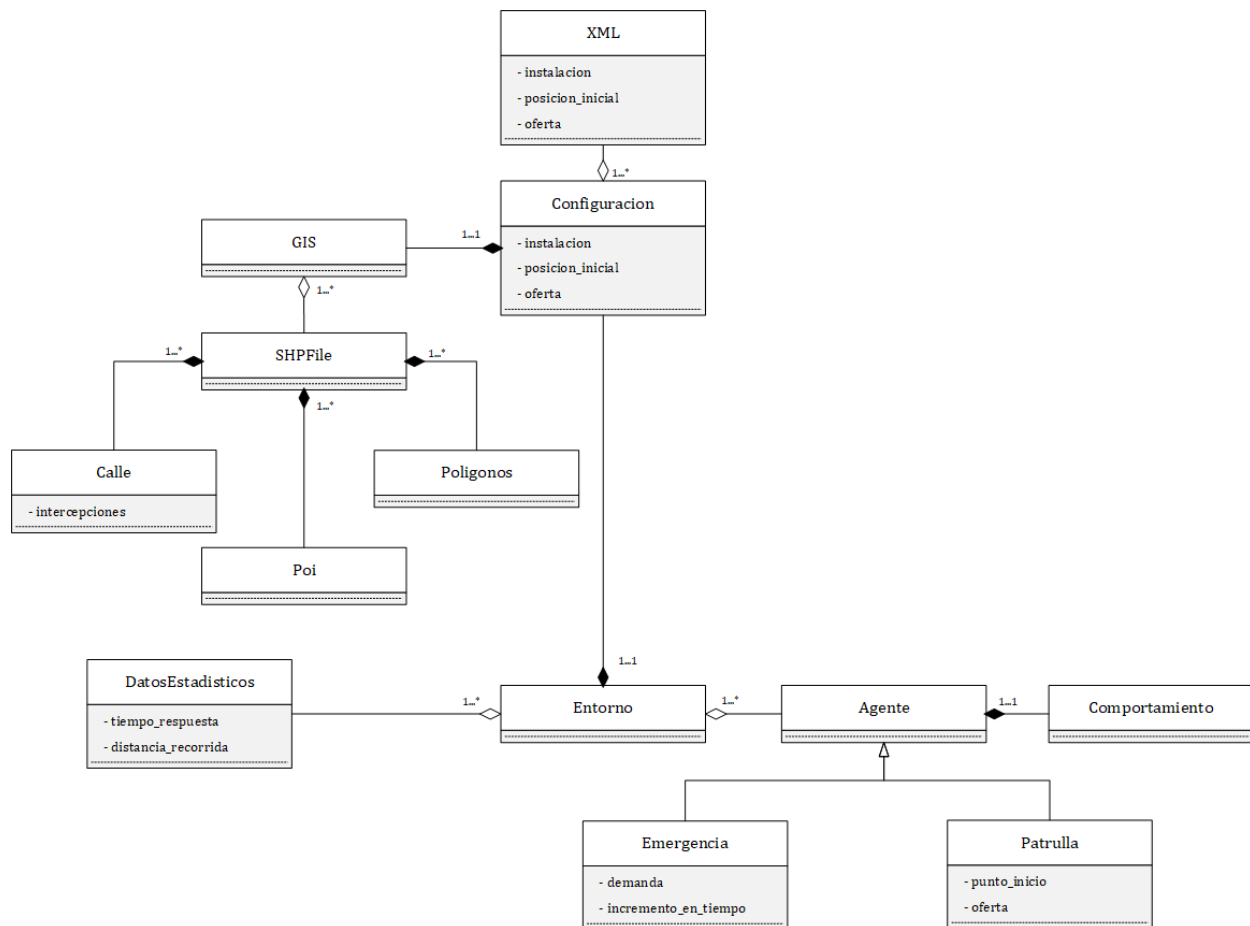


Figura 14: Modelo del dominio

2.3.1 Clases del modelo del dominio.

A continuación, en la tabla 2 se detallan las principales entidades que son utilizadas en la solución propuesta, representadas en el modelo del dominio de la Figura 13.

Tabla 2: Glosario de términos.

Clase	Descripción
GIS	Capa con el conjunto de ficheros que contienen la información geográfica
SHPFiles	Ficheros .shp con la información espacial
Calle	StringLines por los que se van a desplazar los agentes
Poligono	Representación de áreas, edificios y todo aquel polígono que se quiera agregar a la simulación para ayudarla visualmente.
POI	Point of interest. Puntos de interés como escuelas, semáforos y todo aquel punto que se quiera agregar a la simulación para ayudarla visualmente.
Configuración	Contiene los parámetros de la simulación: cantidad de estaciones, velocidad de la simulación, probabilidades de ocurrencia de eventos y demás parámetros.
Comportamiento	Comportamiento que va a tener cada agente en cada paso.
XML	Fichero .xml que contiene datos de interés del problema de máxima cobertura
Datos Estadísticos	Contiene los estadísticos que pueden aportar información valiosa para el análisis y la toma de decisiones.
Entorno	Este es el escenario donde se lleva a cabo la simulación. Contiene la configuración a aplicar y todos los elementos de la simulación.
Agente	Entidad de software con un grupo de propiedades que le permiten su autonomía
Emergencia	Agente emergencia generada, con su valor de demanda y el incremento de la demanda en el tiempo
Patruya	Agente patrulla, con su punto inicial según MLCP y el valor de la oferta que ofrece a la simulación

2.4 Diagrama de actividades del flujo completo del negocio

Una vista de alto nivel del flujo principal del proceso está contenida en el artefacto de la Figura 14. El proceso comienza al importar los datos necesarios al ejecutar la plataforma MASON en este caso son los archivos .shp (mapa) y la solución obtenida de la herramienta MCLPtools [9] que contiene los puntos de partida de cada agente. Luego se muestra el entorno y se ejecuta el inicio de la simulación y de esta forma

cada agente comenzará a patrullar por su área correspondiente. Una demanda entrante generará el envío del coche de policía más cercano y ese coche seguirá el recorrido más corto hacia la ubicación de dicha demanda. Después de que el coche de policía llega a la ubicación de la demanda, la policía permanecerá en ese lugar durante el tiempo de servicio. Cuando termina el tiempo de servicio, la patrulla vuelve a patrullar y de nuevo queda disponible para su envío a la siguiente demanda. Luego, se realiza un reporte y se analizan los datos verificando que los análisis sean factibles o no para mejorar la toma de decisiones.

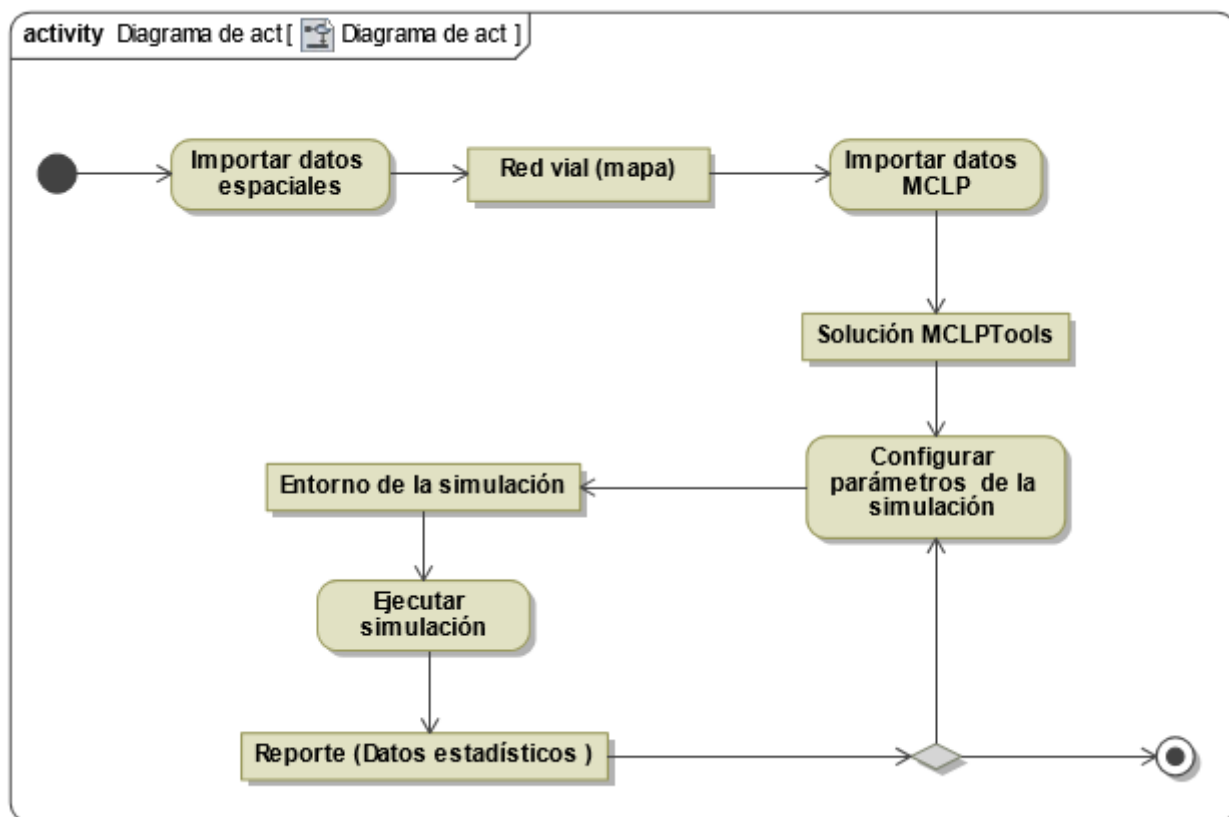


Figura 15: Diagrama de actividades

2.5 Reglas del negocio

Existen un conjunto de restricciones para obtener un correcto funcionamiento de los negocios y sistemas ante la ocurrencia de eventos inesperados y no calculados. Estas restricciones se conocen como reglas del negocio [68]. A continuación, en la Tabla 3: Reglas del negocio se identifican las principales reglas que el negocio debe cumplir para funcionamiento exitoso.

Tabla 3: Reglas del negocio

Patrón	Regla
Flujo de control	<ul style="list-style-type: none"> • Se necesita conocer la variante operativa para saber cuántas estaciones ubicar en cada área. • Se necesita que la patrulla utilice la menor distancia hacia la emergencia para poder llegar lo más rápido posible • Se necesita conocer el mapa sobre el que se ejecutará la simulación para poder ubicar las estaciones en dependencia de su radio de cobertura.
Estructura	<ul style="list-style-type: none"> • El mapa tiene calles. • El mapa tiene áreas. • Las áreas tienen límites. • Las estaciones tienen radio de cobertura. • Las estaciones tienen itinerarios. • Las estaciones tienen facultades. • Las calles tienen semáforos. • Las calles tienen intercepciones. • Las variantes operativas tienen operaciones. • Los jefes patrullas tienen tipo de autorización. • Los XML tienen coordenadas.
Precondición	<ul style="list-style-type: none"> • Solo cuando se haya seleccionado un mapa y las estaciones, estas se visualizarán.
Postcondición	<ul style="list-style-type: none"> • Solo después de finalizada la simulación se pueden analizar los datos generados.
Responsabilidad	<ul style="list-style-type: none"> • El analista es el responsable de configurar correctamente los parámetros de la simulación.
Repetición	<ul style="list-style-type: none"> • Mientras que el analista quiera configurar más estaciones puede agregar más estaciones a la simulación. • Mientras que no se esté satisfecho con los resultados de la simulación se puede reconfigurar y volver a simular el escenario.

2.5.1 Identificación de los Requisitos

En la Tabla 4 se identifican los requisitos que se proponen como requisitos candidatos para el análisis de los requisitos funcionales. Se tienen en cuenta la prioridad, con un rango de 1-5 (donde 5 es la máxima) y también el tipo de requisitos (normal, esperado, innovador).

Tabla 4: Identificación de los requisitos

Prioridad	Nombre del requisito	Tipo de requisito
5	Configurar la simulación	Normal
4	Pausar/Continuar simulación	Esperado
3	Exportar datos estadísticos	Innovador
2	Permitir despliegue multiplataforma	Innovador
5	Permitir analizar datos	Innovador
3	Reportar datos	Esperado
3	Permitir recuperación tras fallos	Esperado
4	Generar demandas automáticamente	Esperado
3	Implementación en el lenguaje java	Normal

2.5.2 Requisitos no funcionales

A continuación, se exponen los requisitos no funcionales a partir de los problemas frecuentes encontrados a partir de la descripción del negocio antes expuesta para darle solución a cada uno de ellos.

Tabla 5: Requisitos no funcionales

Problemas frecuentes	Requisitos no funcionales
Rendimiento	El sistema debe funcionar de manera eficiente, que el sistema se ralentice por cuestiones de rendimiento podría significar una variación en los datos, por lo que para asegurarnos de generar datos correctos la aplicación debe tener un buen rendimiento.
Portabilidad	La herramienta debe admitir multiplataforma para que de esta forma pueda ser ejecutada desde diferentes Sistemas Operativo.
Apariencia y usabilidad	Tener una buena apariencia es clave en este tipo de sistemas, ya que se comprende mejor el entorno que queremos modelar. En cada momento se debe brindar retroalimentación del estado de los procesos. El sistema debe permitir que el usuario pueda entender, recordar y manipular con facilidad todos sus

		elementos.
Tratamiento de excepciones	de	Cualquier excepción que se lance en la ejecución de la herramienta debe ser tratada acorde a una política que impida pérdida de datos y notificarla al usuario para que pueda buscar en ayuda y soporte técnico.
Confiabilidad		Debe poder responder ante los fallos e informar al sistema o aplicación en el cual va a ser utilizado si podrá mostrar la información más tarde, además debe estar activo 24 horas.
Interface Interna		Se estructurará la aplicación de manera que se hagan módulos que permitan un fácil manejo para el desarrollador, también se utilizara información que proviene de servidores externos.
Software		Como requisito de software para el mayor funcionamiento del servicio, se debe instalar previamente, la máquina virtual de Java, en su versión 8.0 o superior, en el entorno de trabajo.

2.6 Diagrama de casos de uso del sistema

A continuación, se muestra la Figura 15, donde quedan representado los casos de uso del sistema definidos y el actor que los inicia.

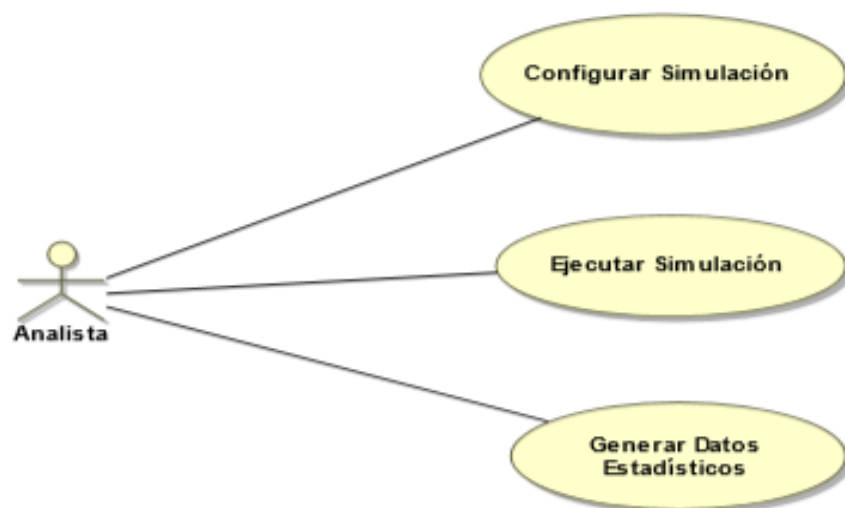


Figura 16: Diagrama de casos de uso del sistema

2.6.1 Actores del sistema

Un actor del sistema es una entidad externa relacionada al sistema y que demanda una funcionalidad al mismo. Esto incluye a los operadores humanos, pero también incluye a todos los sistemas externos [68]. Como bien se ha mostrado, el analista es el único actor del sistema. De este se realiza una breve descripción contenida en la Tabla 6: Descripción de actores del sistema.

Tabla 6: Actores del sistema


Actor del sistema	Descripción
Analista	Es el encargado de iniciar todos los casos de uso del sistema. Tiene la responsabilidad de configurar y ejecutar la simulación, así como obtener los datos estadísticos para posterior análisis. No requiere de estar dotado de conocimientos precisos de informática, más bien basta con un conocimiento básico de operador de aplicaciones.

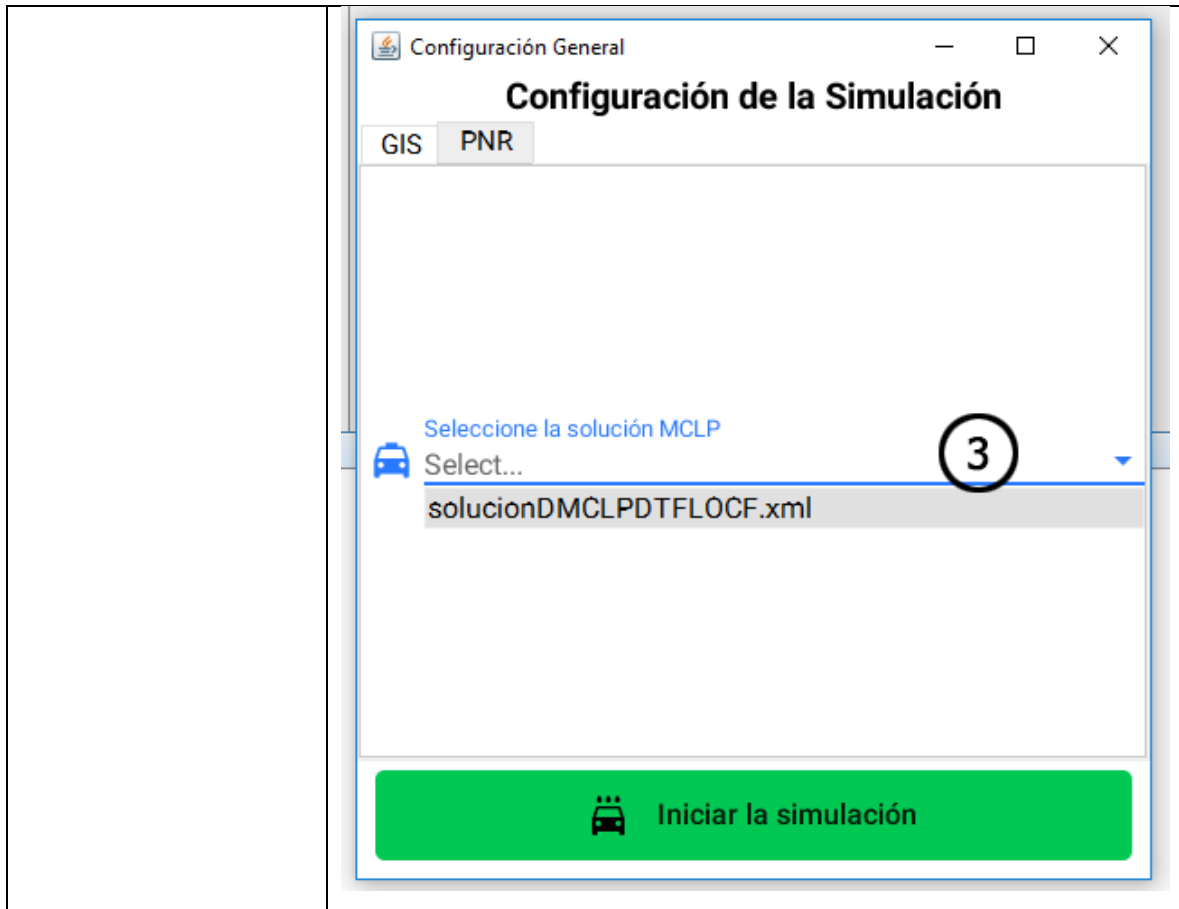
2.6.2 Descripción detallada de los casos de usos del sistema

A continuación, en las tablas: Tabla 7: Caso de uso “Configurar simulación” y Tabla 8: Caso de uso “Ejecutar simulación”, se realiza una explicación detallada de los casos de uso de sistema más importantes desde el punto de vista de la arquitectura. Asimismo, en la Tabla 9: Caso de uso “Exportar datos”, se hace una especificación de alto nivel del caso de uso del sistema: “Exportar datos”.

Tabla 7: caso de uso:” Configurar Simulación”


Caso de Uso	Configurar simulación
Actor	Analista
Descripción	El caso de uso comienza cuando se ejecuta la aplicación, donde se crea la ventana que permite configurar los detalles de la simulación. En esta ventana se puede configurar la parada o pausa automática en un paso específico, la repetición y velocidad. Es donde se seleccionan las acciones para comenzar la simulación. El caso de uso termina cuando termina la simulación, ya que durante de esta se puede reconfigurar el sistema.
Responsabilidades	Este caso de uso es el responsable de que se capture toda la configuración deseada por el usuario.

Precondiciones		Este caso de uso carece de precondiciones, basta con que el usuario tenga decidido como quiere realizar la configuración para efectuarla.
Requisitos funcionales	no	Apariencia y usabilidad
Pantallas Asociadas		



Acción del actor	Respuesta del sistema
Selección del mapa (1)	Se selecciona el mapa sobre el que se va a ejecutar la simulación, esta acción desencadena que se rellene la lista de las áreas auxiliares
Selección de áreas (2)	No responde como tal, pero posteriormente la información es usada para que el sistema dibuje de diferentes colores las áreas auxiliares que se quieren visualizar.
Selección de la solución MCLP a simular (3)	No responde como tal, pero posteriormente la información es usada para que el sistema cargue del XML correspondiente la información inicial de las patrullas.
Click en el botón Comenzar simulación (4)	Comienza el proceso de carga de los datos, localiza los ficheros GIS y XML con la información necesaria, da posteriormente pasa a cargar las pantallas de la simulación y así darle comienzo.

Tabla 8: Caso de uso: "Ejecutar Simulación"

Caso de Uso	Ejecutar simulación
Actor	Analista
Descripción	El caso de uso comienza cuando es actor asociado (Analista) comienza a ejecutar la simulación, se va visualizando por "pasos" el comportamiento de las estaciones en el entorno. Este caso de uso es el que permite ver todo el movimiento de los objetos en el mapa. Permite además acercar y/o alejar el mapa y tirarle fotos o videos a la simulación en cualquier momento. El caso de uso termina cuando termina la simulación.
Responsabilidades	Este caso de uso se responsabiliza por el objetivo más importante de esta herramienta, es decir en él se crean los datos estadísticos de la simulación. También se obtiene gráficamente todos los detalles del proceso.
Precondiciones	Este caso de uso tiene como precondición que exista una configuración que defina bajo qué parámetros se llevará a cabo la simulación.
Requisitos funcionales	no Rendimiento, portabilidad, apariencia, usabilidad, y tratamiento de excepciones.9*9
Pantallas Asociadas	 <p>The screenshot shows a software window titled "Controlador Simulación Patrullas Display". It contains a map of a city grid with numerous red and blue markers. Four numbered callouts are present: (1) at the bottom right, (2) at the top left, (3) at the top left near (2), and (4) at the top right. The map is surrounded by a yellow border. The bottom status bar indicates "Página 55 de 88", "17979 palabras", and "Español (España)".</p>

	
Acción del actor	Respuesta del sistema
Panel de simulación (1)	Panel donde se visualiza el proceso de simulación. Al dar doble-click en él se genera una nueva patrulla o demanda en dependencia de lo que se tenga seleccionado.
Click en el botón Tomar foto (2)	Saca una foto a la simulación justo en el momento en el que se encuentre.
Click en el botón	Comienza la filmación del proceso de simulación hasta

Filmar (3)	que se toque nuevamente el mismo botón y la pause.
Click en los botones de cambiar tamaño (4)	Aleja, enfoca o acerca la simulación respectivamente cada uno de los tres botones de cambio de tamaño en la visualización del mapa.
Click en el combo Box de tipo de agente (5)	Da la opción de seleccionar un tipo de agente (Patrulla o Emergencia) que será visualizado al hacer Click en un lugar específico en el Panel de simulación
Click en los botones de escala (6)	Aleja o carga la escala de los agentes dentro del Panel de simulación, a menor escala mayor tamaño.
Click en el botón Generar Aleatoriedad (7)	Se encarga de generar aleatoriamente en el Panel de simulación la opción señalada en el combo box (5) sin hacer click.
Click en el botón Reiniciar Simulación (7)	Al hacer click se reinicia la simulación, permitiendo reconfigurar los parámetros
Click en el botón play (9)	Comienza la simulación
Click en el botón pausa (10)	Pone en pausa la simulación
Click en el botón stop (11)	Pone en stop la simulación

Tabla 9: Caso de uso: "Generar Datos"

Caso de Uso	Generar Datos
Actor	Analista
Descripción	El caso de uso comienza cuando es actor asociado (Analista) decide generar datos específicos de la simulación. Al presionar el botón asociado el sistema debe generar un fichero con toda la información de la simulación en progreso, que permitirá al actor más adelante analizarlos. Esto permite que el análisis de la información generada se puede hacer independientemente del entorno de simulación. El caso de uso termina cuando todos los datos se exporten.
Responsabilidades	Este caso de uso se responsabiliza por la generación de toda la información que puede ser extraída de la simulación para un posterior análisis y la correcta toma de

		decisiones.
Precondiciones		Este caso de uso tiene como precondición que se haya ejecutado exitosamente una simulación.
Requisitos funcionales	no	Rendimiento y tratamiento de excepciones.

2.7 Vista de la arquitectura: Estructuración en capas

Los diagramas de estructuración en capas de paquetes y subsistemas muestran la arquitectura del sistema a un alto nivel de abstracción.

2.7.1 Enfoque por reutilización

Debido a que el sistema descrito puede ser extendido funcionalmente en el futuro, se modela su arquitectura cumpliendo el enfoque por reutilización. Esto se muestra en la Figura 16.

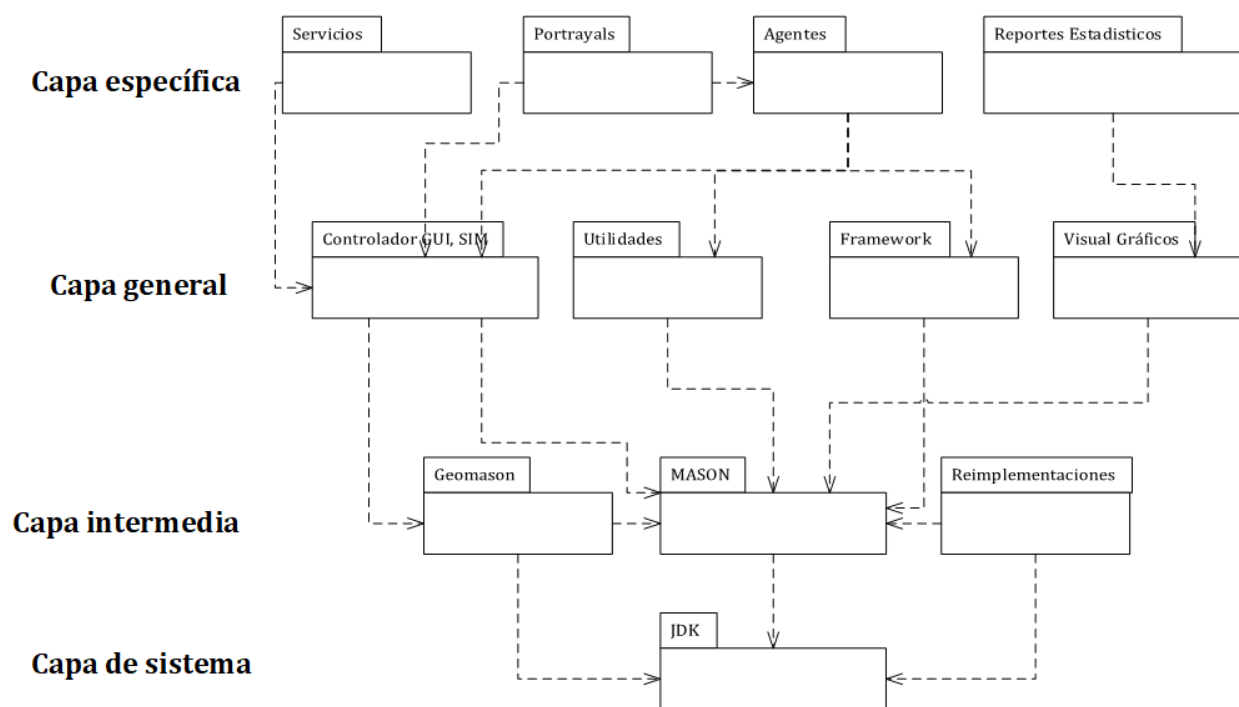


Figura 17: Diagrama de estructuración en capas. Enfoque por reutilización

La capa específica contiene los paquetes que agrupan clases propias y muy específicas del sistema, las cuales no se suelen reutilizar, como la implementación de los agentes y los reportes estadísticos específicos del entorno. En la capa general se

ubican los paquetes que tienen algún grado de reutilización o que requieren de mínimos cambios para ser reutilizados, como clases de utilidades, o el framework que es usa. En la capa intermedia se agrupan los paquetes que poseen el mayor nivel de reutilización, que suelen proveer los marcos de trabajo o bibliotecas que son desarrollados por terceros para ser reutilizados cuando sea necesario, como el mismo MASON o GeoMASON. Finalmente, la capa de sistema sobre el que corre la aplicación, en este caso, el JDK de Java.

2.7.2 Descripción de paquetes de la arquitectura.

Los nombres que se asignan a los paquetes son sugerentes de acuerdo a las clases que agrupan. En la Tabla 10: Descripción de paquetes de la arquitectura se da una breve descripción del contenido de los mismos.

Tabla 10: Descripción de paquetes de la arquitectura

Paquetes	Descripción
Servicios	Contiene integración a servicios como el de recursos e imágenes
Portrayals	Contiene la visualización de cada uno de los agentes
Agentes	Contiene la implementación de los agentes
Reportes Estadísticos	Contiene la lógica de los reportes estadísticos que se soliciten
Controlador GUI, Sim	Contiene los controladores generales de la simulación, tanto para visualización como lógica. Dentro de estos se encuentran los agentes interactuando entre sí.
Utilidades	Contiene clases de utilidad como la implementación de A* y notificaciones del sistema
Framework	Contiene el marco de trabajo sobre el que se va a trabajar sobre MASON. Las mismas clases de MASON implementando interfaces para seguir estándares.
Visuales Gráficos	Contiene la implementación gráfica de los reportes estadísticos
GeoMASON	Paquete GeoMASON para el trabajo con GIS
MASON	Framework de MASON
Reimplementaciones	Reimplementaciones de clases de MASON con estilo patrón proxy para agregar comportamiento antes que se ejecuten los métodos propiamente de MASON.

JDK	Java Development Kit
-----	----------------------

2.8 Patrones de diseño

A continuación, se resumen los principales patrones de diseño utilizados para el desarrollo del sistema

2.8.1 *Facade*

Como se ha explicado en epígrafes anteriores, el paquete “util” contiene una variedad de funcionalidades implementadas que fueron necesarias utilizar en la solución desarrollada. Pese a que no forman parte directamente de la lógica de la solución, las funcionalidades del paquete constituyen dependencias de esta lógica. Una variedad de paquetes hace uso de las funcionalidades del paquete, en principio, esto provoca un acceso desordenado, lo que hace más complicado el modelado. Esto se evidencia en la primera iteración del diseño de estas utilidades, la cual se puede observar en la Figura 17: Diseño de las utilidades, iteración 1. Con propósito de eliminar este problema frecuente se decide hacer uso del patrón *Facade*.

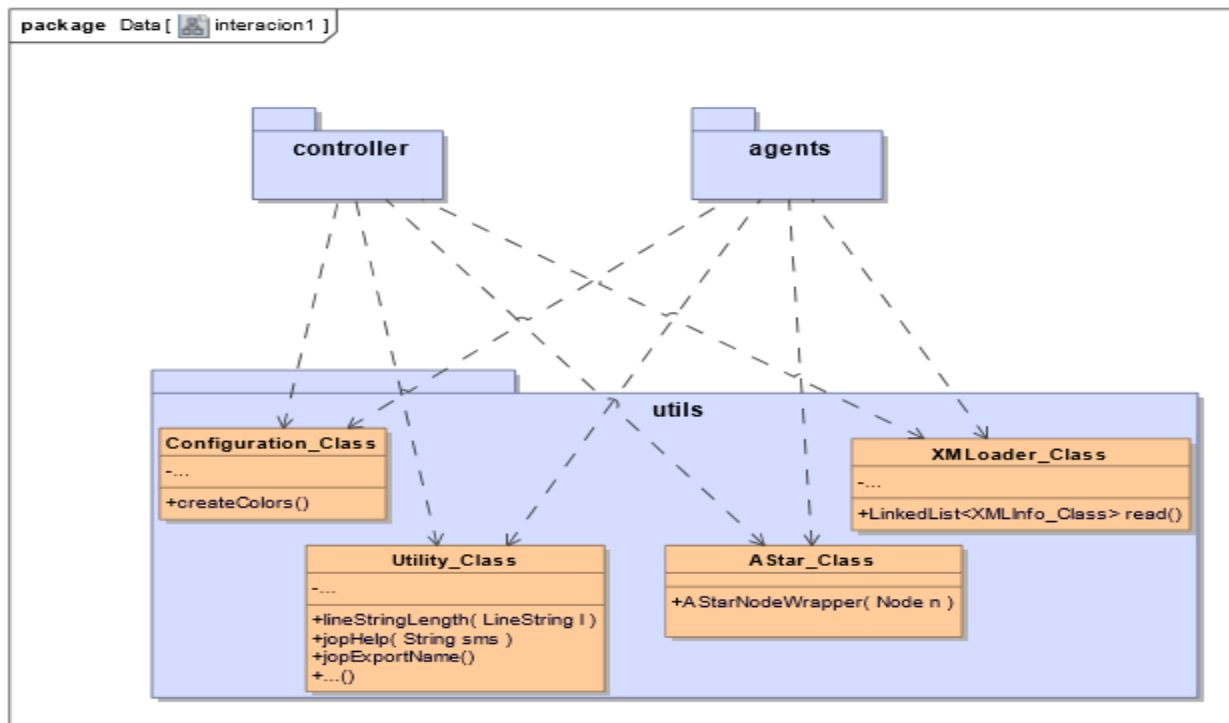


Figura 18: Diseño de las utilidades, iteración 1

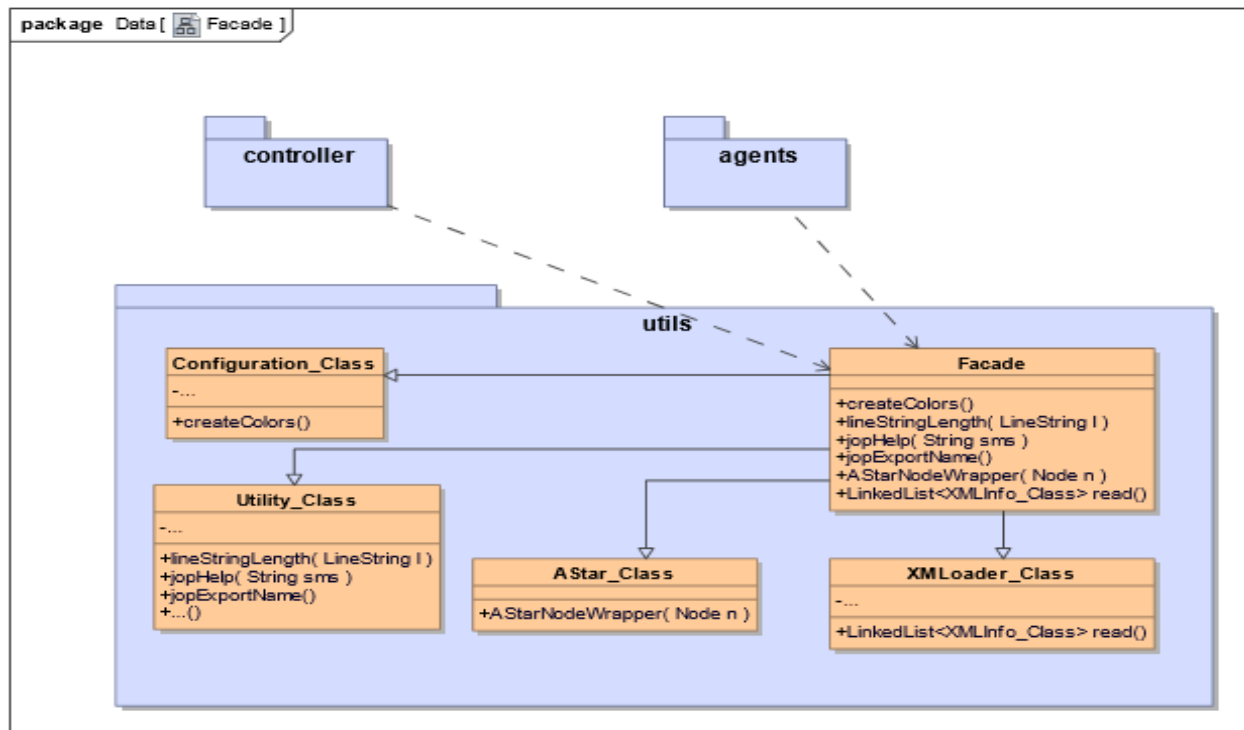


Figura 19: Diseño de las utilidades, iteración 2 (patrón *Facade*)

Al patrón *Facade* o Fachada (en español), se le da uso con el propósito de proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas [70].

Para su aplicación, se crea una nueva clase abstracta llamada “Facade” que funciona como intermediaria entre los componentes del paquete “util” y los paquetes y subsistemas externos a este. En la Figura 20: Diseño de las utilidades, iteración 2 (patrón Facade) se ilustran las modificaciones mencionadas.

Bajo acoplamiento: Solo hay una clase acoplada (*Facade*) por lo que simplifica y centraliza un conjunto de invocaciones en un único punto; proporcionando un acoplamiento débil entre el paquete “utils” y los que dependen de este.

- Ley de Demeter: Las clases cliente se relacionan únicamente con la clase “Facade” minimizando el impacto de los cambios y encapsulando en la Fachada las particularidades

2.8.2 Data Access Object

El patrón de acceso a datos (DAO, *Data Access Object*) tiene como objetivo desacoplar el acceso a datos de su subyacente almacenamiento. Este patrón propone el uso de interfaces (denominadas objetos de acceso a datos) como mecanismo para comunicar los componentes de la aplicación con los dispositivos de almacenamiento de datos (tales como bases de datos o ficheros). A través de este mecanismo se logra desacoplar la implementación de la fuente de datos utilizada.

2.8.3 Template

Template es un patrón de diseño de comportamiento que “define” una clase general abstracta con los pasos a seguir y delega el “como” hacerlos a los hijos que la implementen. Este patrón fue clave al modularizar la aplicación, ya que se definió una clase genérica para “definir” el comportamiento de los módulos, y luego cada módulo la extiende e implementa “como” se va a comportar.

2.8.4 Factory

Factory es un patrón de diseño creacional que provee una interfaz para crear objetos de una misma superclase. Utilizado para crear los agentes ya que todos extienden de la clase “Agent”

2.8.5 Singleton

Singleton es un patrón de diseño creacional que nos fuerza a tener una sola instancia de una clase. Utilizado para solo tener una simulación corriendo a la vez con los mismos parámetros

2.8.6 Adapter

Adapter es un patrón de diseño estructural que permite que objetos con interfaces diferentes colaboren entre sí. Utilizado para hacer la interacción con partes externas como la solución de MCLP-Tool y convertir la información de los ficheros .shp a clases java entendibles por el sistema.

2.8.7 Bridge

Bridge es un patrón de diseño estructural que nos permite separar en “definición” e “implementación” una clase específica. Patrón clave en la modularización del sistema,

ya que se definieron estándares, interfaces para los comportamientos generales, y luego las diferentes implementaciones que interactúan entre sí.

2.8.8 *Proxy*

Proxy es un patrón de diseño estructural que nos provee un sustituto para otro objeto, permitiendo interceptar comportamiento. Utilizado frecuentemente en el paquete de reimplementaciones para ‘inyectar’ código dentro de MASON

2.8.9 **Observer**

Observer es un patrón de diseño de comportamiento que permite crear un mecanismo de suscripción para escuchar cambios en entidades externas y actuar en consecuencia. Utilizado en la comunicación entre los módulos; el módulo de patrullas se registra al de emergencias, y cada vez que se crea una emergencia, se notifica, el módulo de patrulla obtiene la notificación y actúa en consecuencia, mandando agentes al punto.

2.9 Persistencia de datos del software.

En esta sección se hará una descripción de los mecanismos utilizados para implementar la persistencia de los datos en el software.

Según el modelo que se expone, debe existir una funcionalidad que permita guardar los datos obtenidos con el objetivo de un posterior análisis de las instancias ya creadas. Esto se debe a que muchas instancias de problemas pueden ser demasiado grandes y resultaría muy ineficiente tener que gestionar los datos cada vez que se va a trabajar con el software. Por tanto, se decidió desarrollar un mecanismo para guardar los datos y las soluciones de los problemas a través de ficheros. Se decidió utilizar ficheros de formato XML, ya que pueden ser utilizados en cualquier plataforma de desarrollo y además son fáciles de utilizar y procesar por usuarios y aplicaciones.

El equipo de desarrollo que se encarga del manejo de la herramienta MCLPTools definió una estructura para guardar los datos del MCLP. En la Figura 19 se muestra la estructura definida.

```

<?xml version="1.0" encoding="UTF-8"?>
- <DMCLP_DTF_LocF_Sol>
    <Algoritmo>Recocido Simulado Clásico</Algoritmo>
    <Evaluacion>5700.0</Evaluacion>
    <Porciento>73.78641</Porciento>
    + <Periodos cantidad="4">
    + <Instalaciones cantidad="547" cantidad_de_tipos_de_instalaciones="3">
    + <Puntos_de_Demandas cantidad="129">
</DMCLP_DTF_LocF_Sol>

```

Figura 20: Estructura del fichero XML para los datos del problema MCLP.

Para la persistencia de las soluciones de los problemas también fue necesario definir una estructura de ficheros. En la Figura 20 se muestra esta estructura. Como se puede observar ambas estructuras son fáciles de interpretar, además de tener iguales etiquetas para guardar las instalaciones y nodos de demandas.

```

<?xml version="1.0" encoding="UTF-8"?>
- <PoliceCarsSimulation>
    + <Demandas Cantidad="4">
    + <Patrullas Cantidad="36">
</PoliceCarsSimulation>

```

Figura 21: Estructura de fichero XML para las soluciones del problema PoliceCarsSimulation.

2.10 Interfaces visuales del software

Las interfaces visuales del software fueron diseñadas utilizando los componentes que contienen elementos fáciles de identificar y el manejo de estas se ha diseñado de una manera sencilla y amigable para los usuarios. Un elemento esencial en el funcionamiento de estas interfaces es que toda la interacción del usuario con la configuración del problema y la obtención de sus soluciones se realiza a través de mapas, ese es un importante aporte de este trabajo: la combinación de las capacidades de GIS con el modelo MCLP y el análisis de los beneficios que brinda dicha combinación. En el software se tiene acceso a mapas a través de archivos shape. A continuación, se describirán cada una de estas interfaces.

2.10.1 Interfaz visual About

La interfaz visual About, nos brinda la información del autor del sistema, así como un breve resumen de lo que consiste dicha aplicación.

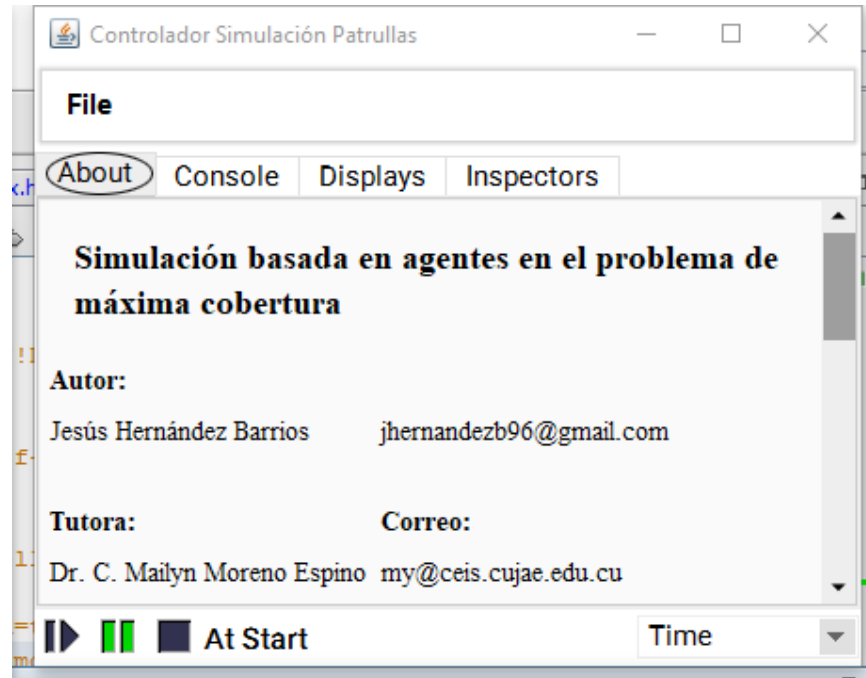


Figura 22: Interfaz visual About

Se puede observar en la Figura 21 que en dicha interfaz se encuentran los controles de play, pausa y stop de la simulación para un mejor análisis y entendimiento y manejo del entorno de simulación.

2.10.2 Interfaz visual Console

La interfaz visual Console se encarga de gestionar todo lo relacionado con la configuración y visualización del movimiento de cada agente dentro del entorno de la simulación. En la Figura 22: Interfaz visual Console se puede apreciar los diferentes parámetros a configurar.

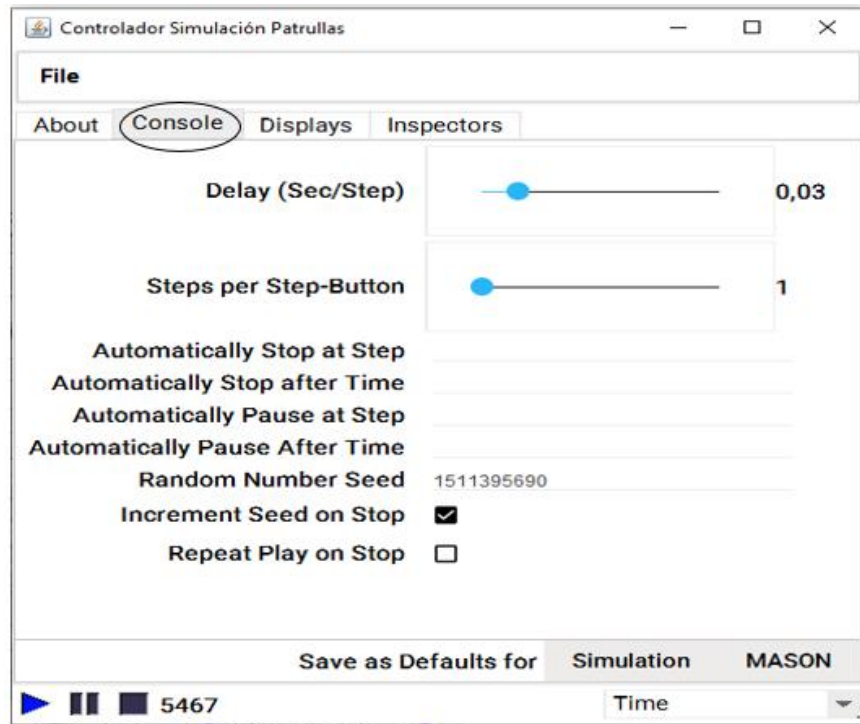


Figura 23: Interfaz visual Console

2.10.3 Interfaz visual Displays

Para la gestión de los datos se diseñó una interfaz que incluyera los campos correspondientes para medir la distancia total, la distancia promedio, el tiempo total y el tiempo promedio de cada instalación a la hora de cubrir una demanda para su posterior análisis (explicado en el epígrafe 2.11.2).

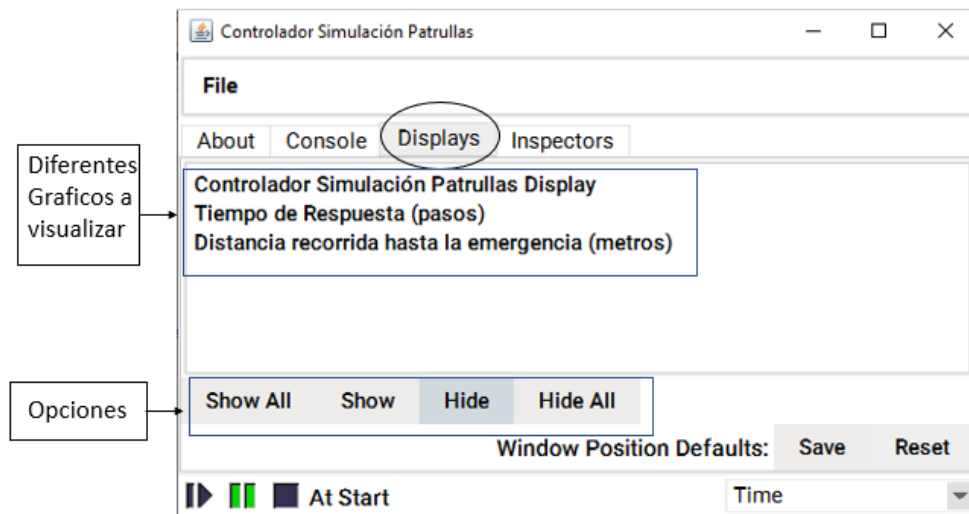


Figura 24: Interfaz visual Display.

Como se puede observar en la Figura 23 el usuario tiene la facilidad de interactuar con los resultados del sistema, a través de gráfica, tener una visualización real de los datos que está gestionando y observar cuál es su comportamiento.

2.10.4 Interfaz visual de Inspectors

En la Figura 24 se muestra un conjunto de informaciones que nos brinda la plataforma MASON al ser generados aleatoriamente diferentes agentes.

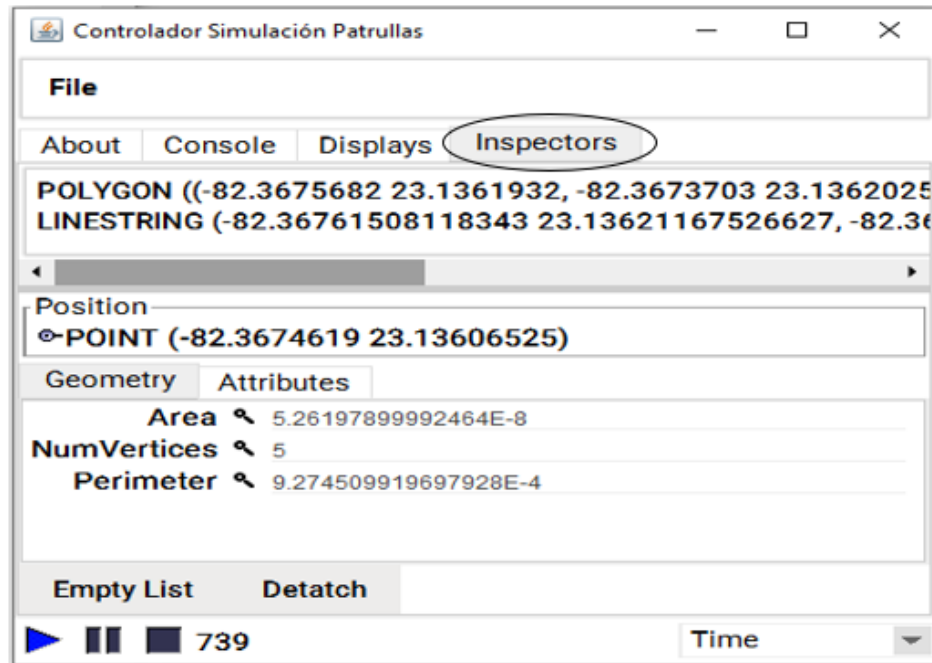


Figura 25: Interfaz visual Inspectors.

Para visualizar alguna información en la figura anterior antes se debe generar un agente aleatoriamente, en este caso la interfaz nos muestra entre otras informaciones las coordenadas, el área y perímetro donde fue creado el nuevo agente, dígame agente patrulla o agente demanda.

2.10.5 Interfaz visual Auxiliar

La interfaz visual Auxiliar (Ver figura 25) se encarga de generar en el entorno de simulación los diferentes tipos de agentes, además de poseer las opciones de aumento de escala de cada uno de ellos y de reiniciar la simulación en caso de un reajuste de los parámetros.



Figura 26: Interfaz visual Auxiliar

2.11 Interfaz de usuario

Para la interfaz de usuario (GUI, *Graphical User Interface*), MASON, en su arquitectura, contiene un grupo de clases encargadas de la visualización y un mecanismo de inspectores para que el analista interactúe con los elementos.

2.11.1 Visualización

La simulación puede tener tantos Display2D como requiera. De cerrar la ventana de Display2D, no se pierde de forma permanente, queda oculta. Esto tiene como beneficio el no requerir que se repinte, lo que permitirá que la simulación se ejecute mucho más rápido.

En la Figura 25: Arquitectura de la GUI, se observa la relación de las clases del paquete “view” con las clases de la sección de “Visualización” que provee MASON.

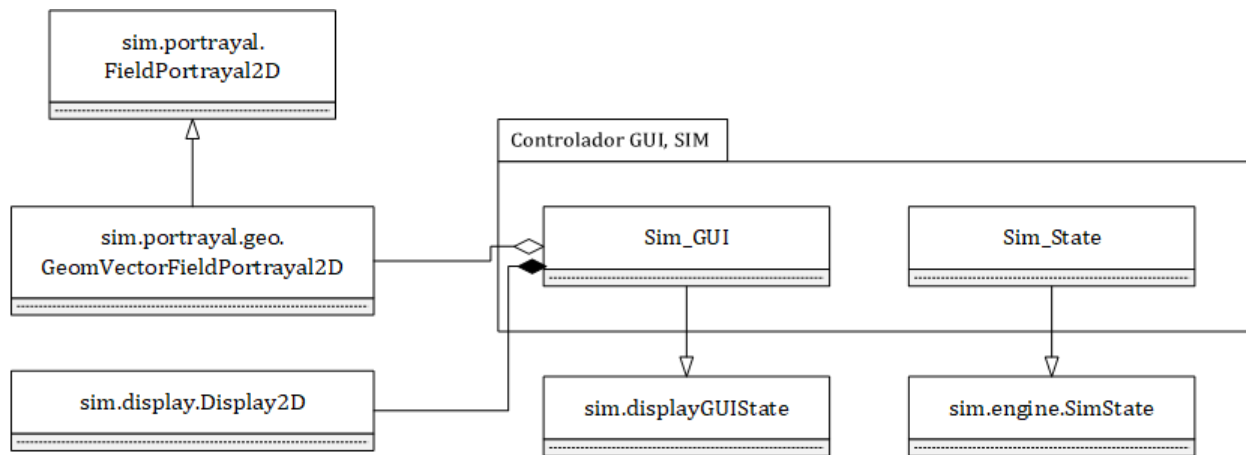


Figura 27: Arquitectura de la GUI

2.11.2 Inspectores

Para seleccionar y gestionar las propiedades de los elementos del entorno, MASON provee un mecanismo de “Inspector”. Un Inspector es un widget GUI que permite al analista del sistema inspeccionar, rastrear, trazar un gráfico y modificar un valor específico o la propiedad de un objeto del modelo.

2.12 Conclusiones parciales

- Se identificaron los principales elementos de análisis de la solución.
- Se realizó el diseño de la herramienta utilizando diagramas UML.
- Se logró visualizar los datos y las soluciones a través de un mapa haciendo uso de la biblioteca geomason.
- Se tuvieron en cuenta patrones de diseño en la implementación del software para garantizar la calidad necesaria.
- Se obtuvo una solución escalable y reutilizable que puede ser fácilmente adaptada y extendida a nuevas funcionalidades y que cumple con los requisitos definidos.
- Se desarrolló una herramienta capaz de vincular el problema de máxima cobertura con la simulación basada en agentes.

Capítulo 3: Validación de la herramienta

3.1 Introducción

En el presente capítulo se presenta la validación de la herramienta de simulación basada en agentes. Para ello se realizan un conjunto de pruebas funcionales de tipo caja negra basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software y un Diseño de Experimentos.

3.2 Pruebas funcionales

Las pruebas funcionales, también conocidas como pruebas de comportamiento, son pruebas de software que tienen como objetivo chequear que los sistemas desarrollados cumplan con las funciones para los cuales han sido desarrollados. Entre las técnicas más usadas, y las elegidas a realizar están las: guiadas por casos de uso y clases de equivalencia.

3.2.1 Guiadas por casos de uso

Se diseñaron y ejecutaron dos casos de prueba asociados a los casos de uso del sistema “Configurar simulación” y “Ejecutar simulación”.

Tabla 11: Descripción del caso de prueba: "Mala configuración de la simulación"

Caso de Prueba	Mala configuración de la simulación
Caso de Uso	Configurar simulación
Desarrollador	Jesús Hernández Barrios
Curso a Probar	Introducción correcta de los datos
Probador	Jesús Hernández Barrios
Objetivo de la Prueba	Garantizar que se seleccionen y carguen correctamente los ficheros con la información geográfica y la solución de MCLP.
Descripción de la prueba	Se selecciona las calles por las cuáles se van a mover los agentes patrullas, se seleccionan las áreas y la solución.
Condiciones:	
Condiciones de Entrada	Resultados Esperados
No existe ningún fichero con información geográfica ni solución de MCLP.	El programa debe mostrar un error diciendo que no se encuentra ninguna información a cargar.

Observaciones:
Los resultados obtenidos fueron los esperados, el software no comienza la simulación si no encuentra ningún fichero.

Por otra parte, al tener tanta importancia el comportamiento de las patrullas dentro del sistema, se le dedica un caso de prueba detallado en la Tabla 12: Descripción del caso de prueba: "Poner en funcionamiento la simulación".

Tabla 12: Descripción del caso de prueba: "Poner en funcionamiento la simulación"

Caso de Prueba	Poner en funcionamiento la simulación
Caso de Uso	Ejecutar simulación
Desarrollador	Jesús Hernández Barrios
Curso a Probar	Introducción correcta de los datos
Probador	Jesús Hernández Barrios
Objetivo de la Prueba	Correcta sincronización de las patrullas durante la simulación
Descripción de la prueba	Probar que al generarse una demanda la patrulla más cercana se dirija al lugar.
Condiciones:	
Condiciones de Entrada	Resultados Esperados
Tiene que existir al menos un agente patrulla que esté disponible para darle cumplimiento a la demanda.	Se espera que al generarse la demanda la patrulla más cercana se dirija al lugar y cambia su estado de disponible a ocupado, parpadeando su color entre rojo y azul en forma de sirena.
Observaciones:	
El resultado esperado y el obtenido resultaron iguales	

3.2.2 Clases de equivalencia

La técnica de clases de equivalencia se usa con el propósito de identificar los valores de entrada para los cuales se estima que el comportamiento sea similar. De esta forma, se crean clases agrupando a los mismos, y luego se elige un valor de cada clase y se realiza una prueba con él. Así, se estaría logrando una amplia cobertura. Los casos de uso "Configurar simulación" y "Ejecutar simulación" son los que estarán relacionados con las pruebas que se desarrollan bajo esta técnica. La Tabla 13: Clases de equivalencia, muestra, dado los rangos del dominio, los valores elegidos para analizar las clases de equivalencia. Se puede observar que se tomaron valores excesivos, de

forma tal que se realicen pruebas orientadas al rendimiento, cuando la herramienta está bajo estrés.

Tabla 13: Clases de equivalencia

Parámetro	Descripción	Valor por debajo del dominio	Normal	Excesivo
CIP	Cantidad inicial de patrullas	-1	10	200
CMP	Cantidad máxima de patrullas	-1	20	500
TPR	Tiempo promedio de respuesta	-10	15	150

Las pruebas desarrolladas siguiendo esta técnica están descritas en la Tabla 14: Pruebas de las clases de equivalencia.

Tabla 14: Pruebas de las clases de equivalencia

Casos de prueba	Datos insertados	Respuesta del sistema
Test 1: Flujo principal (datos correctos)	CIP = 10 CMP = 20 TRP= 15	La simulación carga correctamente, las configuraciones se comprueban que son las introducidas, la simulación se desarrolla de forma esperada.
Test 2: Valores fuera de rango para las cantidades de patrullas	CIP = -1 CMP = -1 TPR= 10	El formulario no impide la entrada incorrecta de estos datos, la simulación se carga, pero la cantidad de vehículos real es cero, se deduce que el software interpreta los números negativos como cero.
Test 3: Prueba de estrés Valores considerablemente grandes para las cantidades de patrullas	CIP = 200 CMP = 500 TPR = 10	Se carga la simulación sin problemas (visuales).

3.2.3 Análisis de los resultados de las pruebas

El diseño y ejecución de los casos de prueba antes descritos, constató la correcta implementación de las funcionalidades presentes en la herramienta, así como la capacidad de este de responder ante errores y brindar mensajes informativos al usuario. A pesar de que el resultado de las pruebas fue el esperado, no es posible afirmar que la aplicación esté exenta de otros errores. Esto debido a que, por cuestiones razonables de tiempo, no se diseñaron y ejecutaron casos de pruebas que cubran el 100% de la cobertura de los Casos de Uso identificados. Por lo tanto, se

recomienda que se diseñen y se ejecuten nuevos casos de pruebas en un laboratorio de calidad que abarquen las funcionalidades en su totalidad. Además, estos casos de pruebas deben ser ejecutados en diferentes sistemas operativos, para garantizar un comportamiento similar de la aplicación independiente de la plataforma donde se encuentre instalada. Respecto a las pruebas de rendimiento realizadas se puede concluir que existen fallos en cuanto a accesos concurrentes a la memoria, los tiempos de respuesta obtenidos son considerablemente pequeños y que la aplicación tiene un funcionamiento fluido.

3.3 Diseño de experimentos

Como parte de la validación de la solución propuesta se lleva a cabo un diseño de experimentos (DoE, Design of experiments). Este es definido como: “una secuencia de pasos tomados de antemano para asegurar que los datos apropiados se obtendrán de modo que permitan un análisis objetivo que conduzcan a deducciones válidas con respecto al problema planteado” [19].

A continuación, se pone de manifiesto las diferentes directrices del DoE orientado al caso “Simulación de patrullaje policial”

3.3.1 Fase de planeación

Esta primera fase comienza con el reconocimiento y formulación del problema. Se quiere realizar una simulación en la ciudad de Centro Habana, de forma tal que los vehículos circulen por esta área.

El rendimiento es tiempo promedio de respuesta, en pasos, de las patrullas luego de una simulación de 8 horas. Este tiempo se ve afectado en cada vehículo cuando la distancia entre el punto de origen y la demanda a satisfacer es muy grande. El objetivo es minimizar este tiempo.

Los factores identificados se dividen en dos grupos, controlables y no controlables. El primer grupo, acompañado de los niveles, está descrito en la Tabla 14: Factores controlables, niveles y unidad de medida. También fueron identificados otros factores que no pueden ser controlados (segundo grupo), pero no por esto menos importantes, ya que tienen una influencia considerable en el rendimiento, como la cantidad de demandas generadas y su valor de demanda

Tabla 15: Factores controlables

Id	Factor	Nivel Bajo (-1)	Nivel Alto (1)	Unidad
A	Cantidad inicial de patrulla	5	20	-
B	Velocidad. Promedio de las patrullas.	40	60	Km/h
C	Distancia del recorrido	1	10	Km

3.3.2 Fase de diseño

En esta fase se elige la resolución de diseño a aplicar. La cantidad de factores controlables son tres, esto implica que la cantidad total de tratamientos es ocho ($2 \times 2 \times 2 = 8$). Esta cantidad es aceptable (considerablemente pequeña en cuanto a ejecuciones), por lo que se decide aplicar una resolución factorial completo. El beneficio más importante que se obtiene con esta resolución de diseño es que se evita la confusión o alias, la cual estaría presente de realizar menos ejecuciones que cantidad de tratamientos total [19].

3.3.3 Fase de conducción

Para llevar a cabo la fase de conducción se realizan las 8 ejecuciones con los tratamientos correspondientes. Como herramienta de apoyo se hace uso del software estadístico Minitab [73,74].

La Tabla 15: Rendimiento de cada tratamiento, porta los resultados en cuanto a rendimiento de cada experimento realizado. El orden de tratamientos es generado por Minitab.

Tabla 16: Rendimiento de cada tratamiento

Ejecuciones	A	B	C	Tiempo
1	-1	-1	1	9
2	-1	1	1	22
3	1	-1	1	13
4	1	1	-1	10
5	1	1	1	37
6	-1	1	-1	11
7	1	-1	-1	25
8	-1	-1	-1	31

3.3.4 Fase de análisis

Minitab, luego de insertado los datos, realiza un procesamiento de estos, que tiene como resultado un conjunto de gráficas que harán posible el análisis final. En principio, se quiere conocer cuáles de los factores principales son influyentes para el rendimiento del problema.

Por su parte, la gráfica de Pareto de la Figura 27: Gráfica de Pareto de efecto y rendimiento, muestra una comparativa clara entre el efecto de cada factor o interacción. En esta se observa que el factor A (Cantidad inicial de patrulla) es influyente, y agrega que no existe ninguna interacción de factores con una influencia considerable.

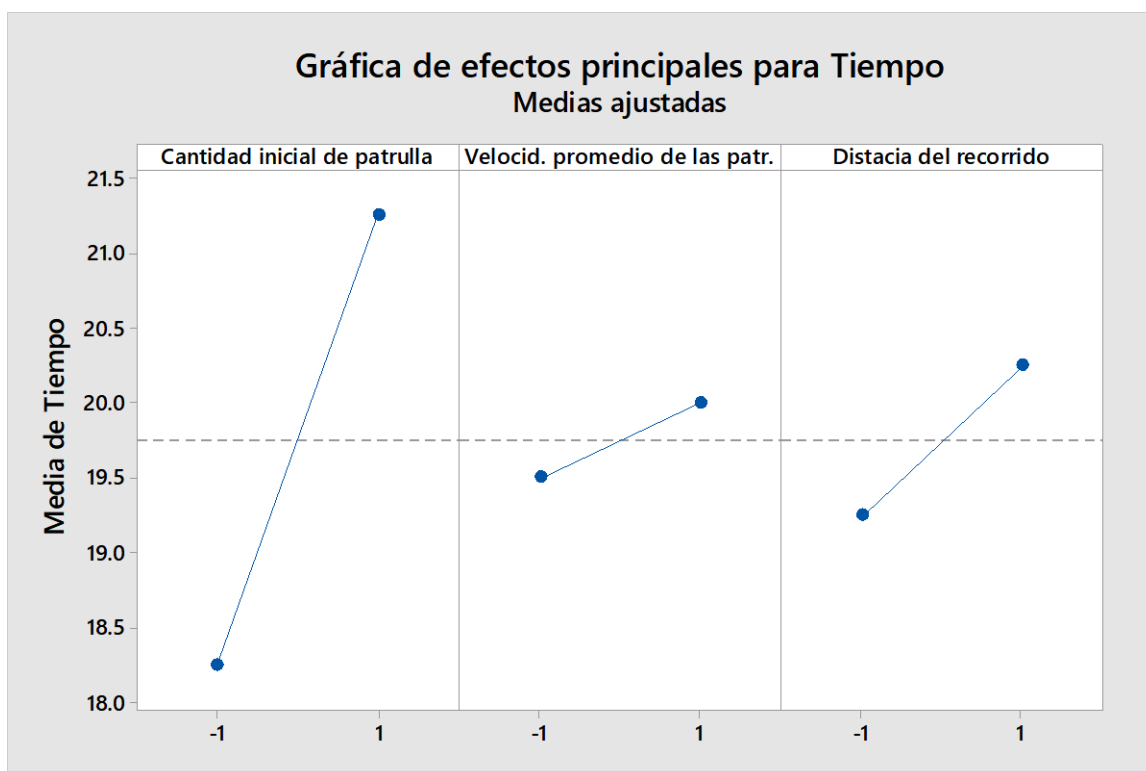


Figura 28: Gráfica de efectos principales

Por su parte, la gráfica de Pareto de la Figura 28: Gráfica de Pareto de efecto y rendimiento, muestra una comparativa clara entre el efecto de cada factor o interacción. En esta refleja que el factor BC (Velocidad promedio y distancia del recorrido) es el influyente, y agrega que no existe ninguna interacción de factores con una influencia considerable.

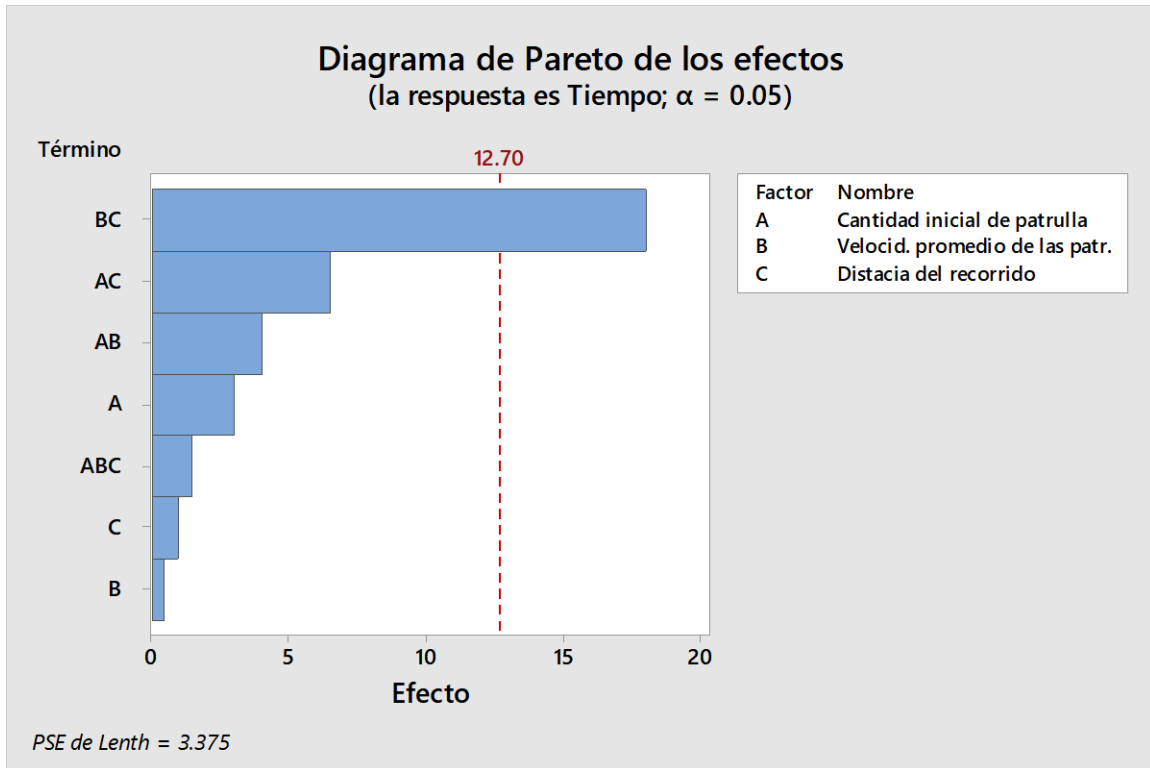


Figura 29: Grafica de Pareto

Como ambas figuras muestran valores diferentes acerca del factor influyente para tener una mayor certeza, Minitab ofrece la gráfica ilustrada en la Figura 29: Gráfica de probabilidad normal.

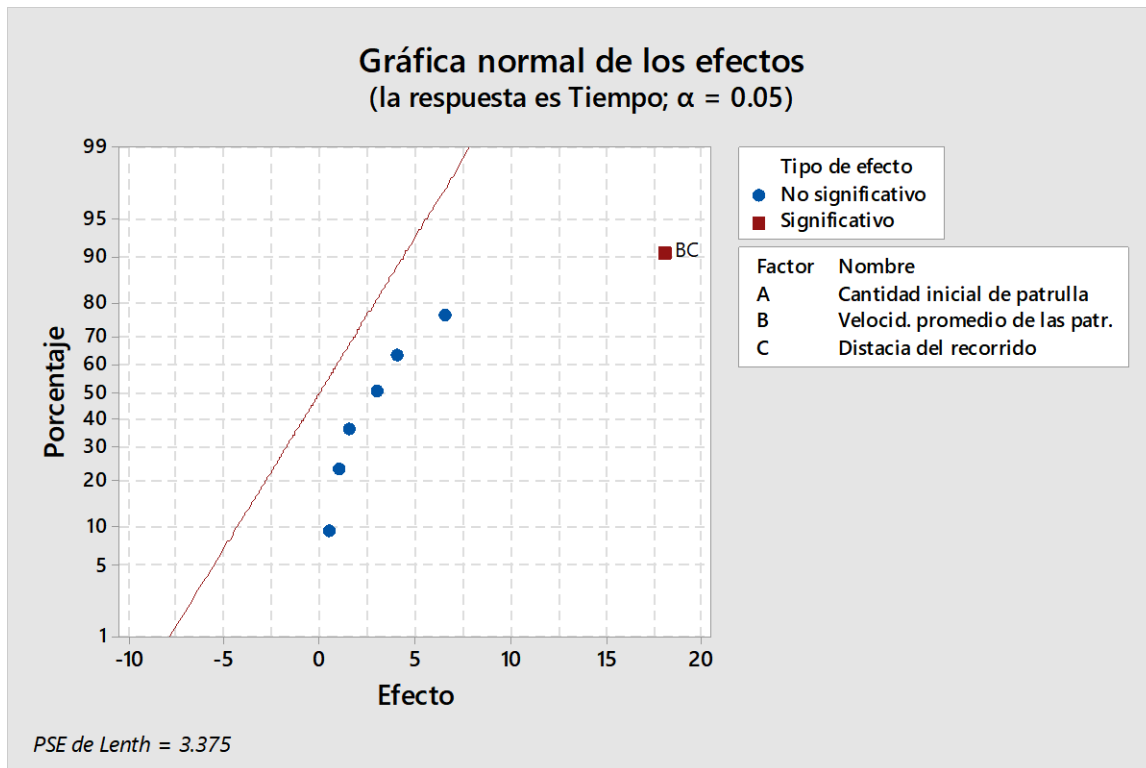


Figura 30: Gráfica de probabilidad normal.

En esta gráfica se observa que, luego de plotear los puntos de cada efecto, el punto del factor BC es el único que dista considerablemente de la recta ploteada.

A la necesidad de conocer cuál es la configuración óptima para este DoE, Minitab genera una gráfica de cubo. La Figura 30: Gráfica de cubos, denota a cada vértice del cubo con el resultado del rendimiento. El objetivo del diseño es minimizar el tiempo promedio de respuesta, por lo tanto, al buscar el mínimo, se puede apreciar que el menor tiempo promedio de las patrullas es de 10 minutos. Entonces, las coordenadas de este vértice indican el tratamiento que logró un mejor rendimiento.

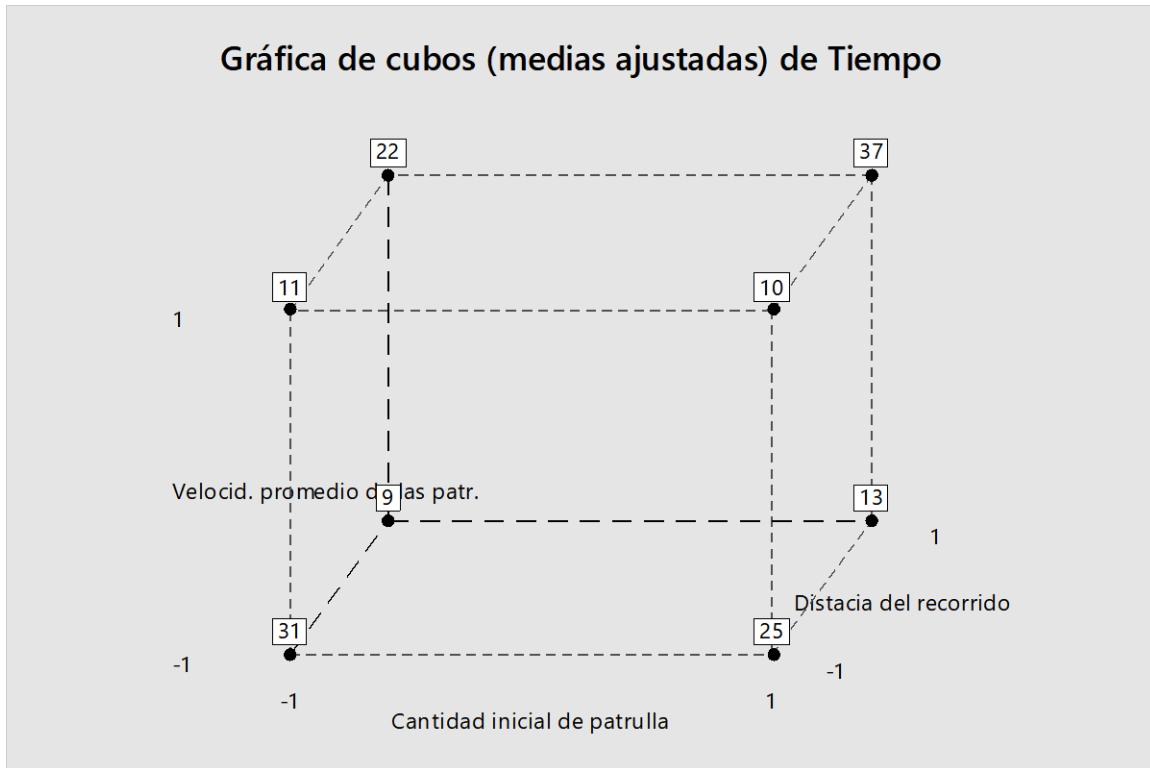


Figura 31: Grafica de cubos.

3.3.5 Conclusiones del DoE

Luego de la aplicación del DoE se puede concluir que:

- A pesar de necesitar varias graficas para determinar los factores influyentes es evidente que el factor BC (ver Tabla 15) es el más resaltado.
- Para el DoE, no existe ninguna interacción de factores con un efecto influyente.

3.4 Conclusiones parciales

En el presente capítulo se llega a la conclusión de que:

- Para la validación de la herramienta, se seleccionaron pruebas funcionales utilizando dos técnicas fundamentales: guiada por casos de uso y clases de equivalencia, con el objetivo de detectar fallas en el cumplimiento de los requisitos.
- Para la mayoría de los escenarios, los resultados obtenidos se correspondieron con los esperados o bien eran resultados aceptados.
- Se realizó un DoE donde se puso en funcionamiento la herramienta.

- Se demostró que la herramienta cumple con sus funcionalidades para la cual fue desarrollada.

Conclusiones Generales

Con la realización del presente trabajo se pudo otorgar elementos para demostrar la efectividad de la utilización de la simulación basada en agentes para la creación de modelos de simulación capaces de imitar procesos y actividades de la vida real. Tomando en cuenta principios y patrones de diseño que brindan calidad y flexibilidad, se propuso un software que permite:

- La carga y salva de datos mediante ficheros XML y JSON.
- Generar patrullas en las zonas más delictivas del municipio de Centro Habana
- La incorporación de nuevas capas en el mapa (shapeFiles) para una mejor visualización de la simulación.
- Aumentar la escala de las patrullas y las emergencias generadas para una mejor visualización.
- Generar eventos.

Luego, se arribó a la siguiente conclusión:

- La integración de los sistemas de Información Geográfica (GIS) en dichos procesos debe ser un requisito a la hora de modelar los procesos del patrullaje policial ya que aporta datos reales al ambiente de simulación.
- Se pueden optimizar considerablemente muchos procesos con un sistema de simulación bien implementado.
- Se obtuvo una solución escalable y reutilizable que puede ser fácilmente adaptada y extendida a nuevas funcionalidades gracias al uso de buenas prácticas como patrones de diseño.
- La solución fue validada mediante la ejecución de los casos de prueba y un diseño de experimento, con los que se tuvo constancia de la correcta implementación de las funcionalidades presentes en la herramienta.

Recomendaciones

Se recomienda que:

- Se tomen en cuenta otros tipos de agentes como policías en motos, bicicletas y a pie para que puedan realizar otros recorridos
- Se diseñen y se ejecuten nuevos casos de pruebas en un laboratorio de calidad que abarquen las funcionalidades en su totalidad.
- Se diseñen las áreas por consejos en el mapa de forma tal que no se salgan de su área las patrullas.
- Lograr que cada patrulla permanezca en su área a menos que esté autorizado.
- Lograr mantener la máxima cobertura de cada instalación luego de que comiencen su itinerario.
- Mejorar la representación gráfica de los vehículos y las emergencias.

Referencias bibliográficas

1. Fraga, D.B., Herramienta de Simulación Basada en Agentes para la evaluación de configuraciones semafóricas en redes viales. 2018, Cujae. p. 83.
2. M, M.M.N., "Tutorial on agent-based modelling and simulation". J. Simul, 2010.
3. M, N.M.M., "Agent Based modeling and simulation," presented at the Winter Simulation Conference. 2009.
4. R, M., Computer Simulation. A practical perspective. Academic Press, Inc, 1191. 1.
5. Law AM, K.W., Kelton WD, *Simulation modeling and analysis*. McGrawHill New York, 1991. 2.
6. M, C., Tipos de simulación. 2003.
7. C. N. North. M, "Complex adaptive systems modeling with Repast Symphony,". Springer, 2013.8.
8. Serena C, "Complex Adaptive Systems," presented at the Research Seminar in Engineering Systems. 2001.
9. Nodarse, C.P., SOFTWARE PARA MODELAR Y RESOLVER EL PROBLEMA DE MÁXIMA COBERTURA.10. 2016, ISPJAE. p. 121.
10. ReVelle, R.C.a.C., "The maximal covering location problem"Papers of the Regional Science Association. Papers of the Regional Science Association 1974. 32(1): p. 101-118.
11. Krass, O.B.a.D., "The generalized maximal covering location problem". Computers & Operations Research, 2002. vol. 29, no. 6, pp. 563–581(6).
12. Nodarse, C.P., Propuesta de solución para el problema de máxima cobertura dinámico con tipos de instalaciones. 2018, ISPJAE. p. 100.
13. Wooldridge, M., An Introduction to MultiAgent Systems, ed. n. ed. 2009: John Wiley & Sons.
14. L, T., Agent-based computational economics: modeling economies as complex adaptive systems. . Inf Sci., 2003(149(4)): p. 262–268. .
14. L, T., Agent-based computational economics: modeling economies as complex adaptive systems. . Inf Sci., 2003(149(4)): p. 262–268.
15. E, B., Agent-based modeling: Methods and techniques for simulating human systems. Proc Natl Acad Sci, 2002: p. 7280–7287.
16. Lin, P.-F.C.S.-J., "Using Swarm to Build a Multi-Agent Simulation System for Artificial Stock Market,". presented at the Proceedings of the Second Workshop on Knowledge Economy and Electronic Commerce, 2009.

17. Izquierdo LR, G.J., Santos JI, Del Olmo R Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas. EMPIRIA Rev Metodol Las Cienc Soc., 2008.
18. R, M., Computer Simulation. A practical perspective. Academic Press, Inc, 1991. 1.
19. Davidsson, "Agent Based Social Simulation: A Computer Science View,". J. Artif. Soc. Soc. Simul, 2002. 5.
20. FS, H., Investigación de operaciones. McGraw-Hill Interamericana de España S.L., 2010: p. 1008.
21. Díaz, D.I., Simulación Social Basada en Agentes: Estudio de casos 2018, ISPJAE. p. 81.
22. Zhou Z, C.W.V., Chow JH, Agent-based simulation of electricity markets: a survey of tools. Artif Intell Rev, diciembre de 2007: p. 305–342.
23. Singh D, P.L., Logan B, Integrating BDI Agents with Agent-Based Simulation Platforms. Auton Agents Multi-Agent Syst., noviembre de 2016: p. 1050–1071.
24. YB, M., *Simulation modelling for sustainability a review of the literature*. . Int J Sustain Eng, 2017: p. 2-19.
25. Abar S, T.G., Lemarinier P .Agent Based Modelling and Simulation tools: A review of the state-of-art software. . O'Hare GMPComput Sci Rev, 2017;; p. 1333.
26. Fazel Zarandi, M. H., et al.: "The large-scale dynamic maximal covering location problem", Mathematical and Computer Modelling, Vol. 57, No. 3-4, pp. 710-719, 2013.
27. Church, R. and C. ReVelle: "The maximal covering location problem", Papers of the Regional Science Association, Vol. 32, No. 1, pp. 101-118, 1974.
28. Toregas, C.: "A covering formulation for the location of public service facilities", Master's Thesis, Cornell University, 1970.
29. Fazel Zarandi, M., et al.: "The large-scale dynamic maximal covering location problem", Mathematical and Computer Modelling, Vol. 57, No. 3-4, pp. 710-719, 2013.
30. Murray, A. T.: "Maximal Coverage Location Problem: Impacts, Significance, and Evolution", Internacional Regional Science Review, Vol. 39, No. 1, pp. 527, 2016.
31. Chawathe, S. S.: "Organizing Hot-Spot Police Patrol Routes", Intelligence and Security Informatics, pp. 79 - 86, 2007.
32. Curtin, K. M., et al.: "Determining Optimal Police Patrol Areas with Maximal Covering and Backup Covering Location Models", Netw Spat Econ, Vol. 10, pp. 125-145, 2010.
33. Amponsah, S., et al.: "Location of ambulance emergency medical service in the Kumasi metropolis, Ghana", Journal of Mathematics and Computer Science Research, Vol. 4, No. 1, pp. 18-26, 2011.

34. Azizan, M. H., et al.: "Application of OpenStreetMap Data in Ambulance Location Problem", Fourth International Conference on Computational Intelligence, Communication Systems and Networks, 2012.
35. Erdemira, E. T., et al.: "Location coverage models with demand originating from nodes and paths: application to cellular network design", European Journal of Operational Research, Vol. 190, No. 3, pp. 610-632, 2008.
36. Tutschku, K.: "Demand-based Radio Network Planning of Cellular Mobile Communication System", Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings, pp. 1054-1061, 1998.
37. Caro Vela, M. D. and C. Paralera Morales: "Una propuesta para la localización de áreas de servicio y descanso adaptadas al transporte de mercancías peligrosas mediante un modelo de optimización; aplicación al territorio español", Revista de métodos cuantitativos para la economía y la empresa, Vol. 11, No. 1, pp. 17-32, 2011.
38. de Rainville, F. M., et al.: "Co-adapting Mobile Sensor Networks to Maximize Coverage in Dynamic Environments", Proceedings of the 14th annual conference companion on Genetic and evolutionary computation, pp. 1409-1410, 2012.
39. Zhou, J., et al.: "A Multi-Objective Model for Fire Station Location under Uncertainty", Advances in information Sciences and Service Sciences, Vol. 5, No. 7, pp. 1184-1191, 2013.
40. Tanaka, K.-i.: "Maximum flow-covering location and service start time problem and its application to tokyo metropolitan railway network", Journal of the Operations Research Society of Japan, Vol. 54, No. 4, pp. 237-258, 2011.
41. S. Steiniger and A. J. Hunter, "Free and open source GIS software for building a spatial data infrastructure," Geospatial Free Open Source Softw. 21st Century, pp. 247–261, 2012.
42. Karen K. Kemp, Encyclopedia of Geographic Information Science. Thousand Oaks, California 91320: SAGE Publications, Inc., 2008.
43. A. Hillier, "Working with ArcView 10," 2008.
44. T. E. Thoutireddy. P, "GeoFramework: Coupling multiple models of mantle convection within a computational framework," Electron. J. Earth Sci., vol. Volume 7, 2006.
45. M. L. Treglia, An Introduction to GIS using QGIS. 2017.
46. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology," Auton. Agents Multi-Agent Systems, vol. 8, pp. 203–236, 2004.
47. A. N. D. Bertolini, "TAOM4E: An Eclipse ready tool for Agent-Oriented Modeling. Issue on the development process," Fondazione Bruno Kessler - Irst Trento-Povo, 2006.

48. M. Rosenshine, Contributions to the theory of patrol scheduling. *Oper. Res. Q.* (1970-1977). 21, 99–106 (1970).
49. C. D. Hale, *Police Patrol, Operations and Management*. (Prentice Hall, Upper Saddle River, 1980)
50. S. J. D'Amico, S. J. Wang, R. Batta, C. M. Rump, A simulated annealing approach to police district design. *Comput. Oper. Res.* 29(6), 667–684 (2002).
51. P. E. Taylor, S. J. Huxley, A break from tradition for the San Francisco Police: patrol officer scheduling using an optimization-based decision support system. *Interfaces*. 19, 4–24 (1989).
52. K. M. Curtin, K. Hayslett-McCall, F. Qiu, Determining optimal police patrol areas with maximal covering and backup covering location models. *Netw. Spat. Econ.* 10, 125–145 (2010).
53. N. Malleson, *Agent-Based Modelling of Burglary*, PhD thesis. (School of Geography, University of Leeds, 2010).
54. Yue Zhang, Donald E Brown, *Police patrol districting method and simulation evaluation using agent-based model & GIS*. (2013).
55. FIPA. The Foundation For Intelligent Physical Agents.
56. MASON [Internet]. [citado 14 de marzo de 2017]. Disponible en: <https://cs.gmu.edu/~eclab/projects/mason/docs/> .
57. Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G. Mason: A multiagent simulation environment. *Simulation*. 2005;81(7):517–527.
58. FIPA. The Foundation For Intelligent Physical Agents.
59. Java (lenguaje de programación). En: Wikipedia, la enciclopedia libre [Internet]. 2017. Disponible en: [https://es.wikipedia.org/w/index.php?title=Java_\(lenguaje_de_programaci%C3%B3n\)&oldid=100108168](https://es.wikipedia.org/w/index.php?title=Java_(lenguaje_de_programaci%C3%B3n)&oldid=100108168)
60. Jade Site | Java Agent Development Framework [Internet]. [citado 27 de junio de 2017]. Disponible en: <http://jade.tilab.com/>
61. Qué es una Licencia LGPL | SoftDoit [Internet]. [citado 27 de junio de 2017]. Disponible en: <https://www.softwaredoit.es/definicion/definicion-licencia-lgpl.html>
62. Ramos Piloto J. Simulación basada en agentes para el chequeo de requisitos tempranos en i* [Internet] [Thesis]. Instituto Superior Politécnico: José Antonio Echeverría; 2015 [citado 12 de abril de 2017]. Disponible en: <http://tesis.cujae.edu.cu:8080/xmlui/handle/123456789/4815>
63. Caire G. Jade Tutorial. *Appl-Defin Content Lang Ontol* [Internet]. 2002; Disponible en: <http://sharon.cselt.it/projects/jade/doc/tutorials/JadeAndroid-Programming-Tutorial.pdf>

64. Laureando PG, Morandini M, Penserini L, Susi A. Knowledge Level Engineering of BDI Agents. Disponible en: <http://selab.fbk.eu/morandini/resources/ThesisMirkoMorandini.pdf>
65. Bellifemine FL, Caire G, Greenwood D. Developing multi-agent systems with JADE [Internet]. Vol. 7. John Wiley & Sons; 2007. Disponible en: <https://books.google.com/books?hl=es&lr=&id=scComY9Kq40C&oi=fnd&pg=PR5&dq=Developing+MultiAgent+Systems+with+JADE&ots=3vh9H3yUMd&sig=spwBG4KZ9v3qJeFLfolHiFNxM Qc>
66. Braubach L, Pokahr A, Lamersdorf W. Jadex: A BDI-agent system combining middleware and reasoning. *Softw Agent-Based Appl Platf Dev Kits*. 2005;143–168.
67. Pokahr A, Braubach L, Lamersdorf W. Jadex: A BDI reasoning engine. *Multi-Agent Program*. 2005;149–174.
68. Pokahr A, Braubach L, Walczak A. Jadex user guide. Hambg Ger. 2007;43.
69. SÖNMEZ D. JADE, JADEX, RETSINA, DECAF Etmen Geliştirim Platformlarının Karşılaştırılması. Disponible en: <http://ab.org.tr/ab15/bildiri/211.docx>
70. Beheshti R, Jalalpour M, Glass TA. Comparing methods of targeting obesity interventions in populations: An agent-based simulation. *SSM - Popul Health*. 2017;3:211-8.
71. NetLogo Home Page [Internet]. [citado 26 de junio de 2017]. Disponible en: <http://ccl.northwestern.edu/netlogo/>
72. North MJ, Tatara E, Collier NT, Ozik J, others. Visual agent-based model development with repast symphony [Internet]. Tech. rep., Argonne National Laboratory; 2007. Disponible en: <http://cc.ist.psu.edu/BRIMS/archives/2008/Tutorials/08-BRIMS-001.pdf>
73. Robertson A. Agent-based modeling toolkits NetLogo, RePast, and Swarm. 2005; Disponible en: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.207.1035>
74. Collier N. Repast: An extensible framework for agent simulation. *Univ Chicago's Soc Sci Res*. 2003;36:2003.
75. North MJ, Howe TR, Collier NT, Vos JR. The repast symphony runtime system. En: *Proceedings of the agent 2005 conference on generative social processes, models, and mechanisms* [Internet]. ANL/DIS-06-1, co-sponsored by Argonne National Laboratory and The University of Chicago; 2005. p. 13–15. Disponible en: <https://pdfs.semanticscholar.org/633d/c355f38738bb8968ad43c56b99f244d5c11f.pdf>
76. Inc EF. Eclipse - The Eclipse Foundation open source community website. [Internet]. [citado 14 de marzo de 2017]. Disponible en: <https://eclipse.org/>
77. freelancerviet.vn. GAMA is a modeling and simulation development environment for building spatially explicit agent-based simulations. [Internet]. freelancerviet.vn. [citado 20 de junio de 2017]. Disponible en: <http://gamaplatform.org/references#PlatformDocumentation>

78. E. Sánchez-Ansola, L. Sánchez-Jiménez, C.J. de-Armas-García, “Una Mirada al Análisis de Redes de Transporte en Cuba, desde el Punto de Vista de los Datos”, Lámpsakos, N° 10, pp. 21-33, 2013.

Anexos

Anexo 1: Entorno de la simulación.

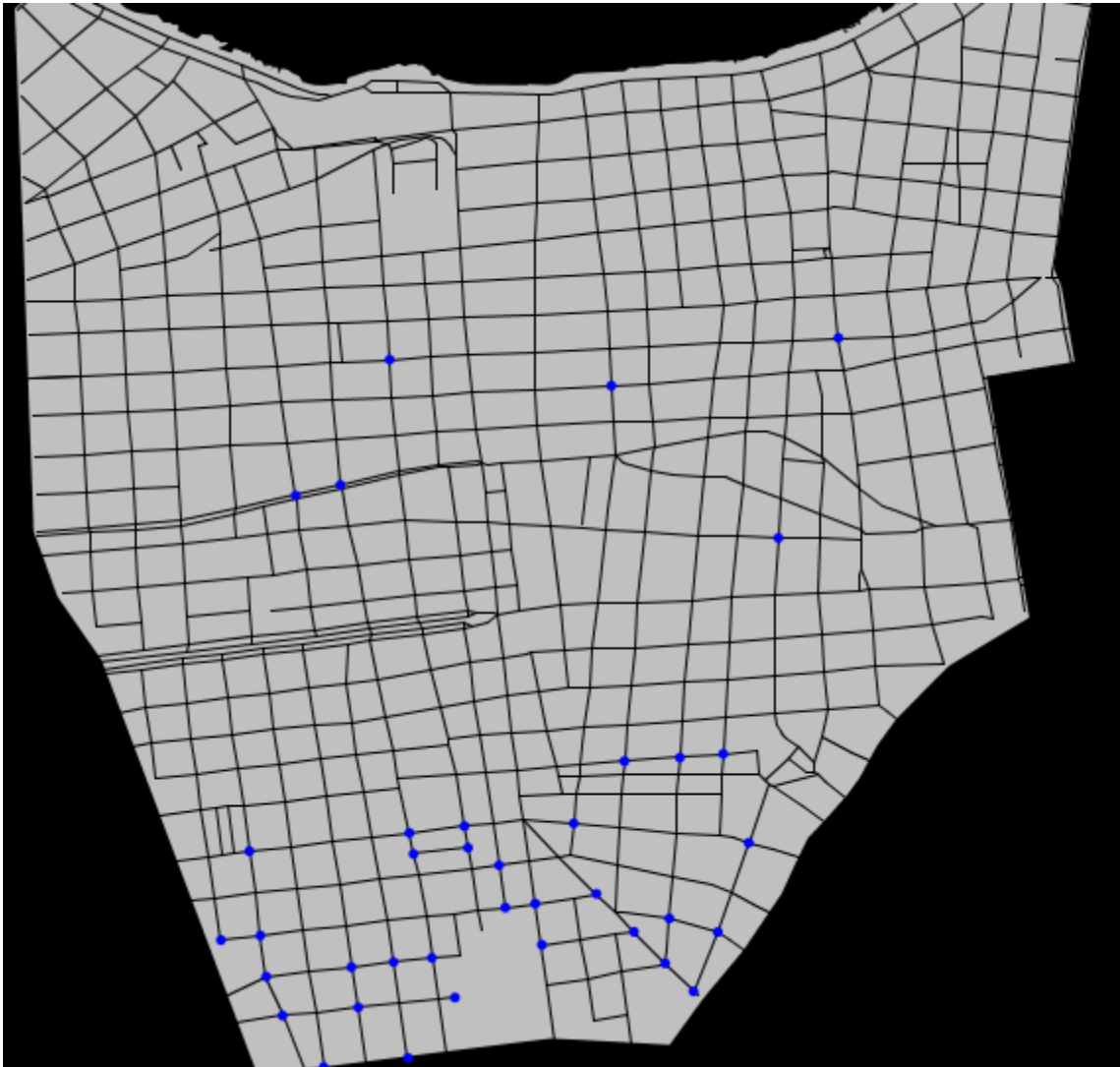


Figura 32: Visualización del entorno de la simulación en la primera versión.

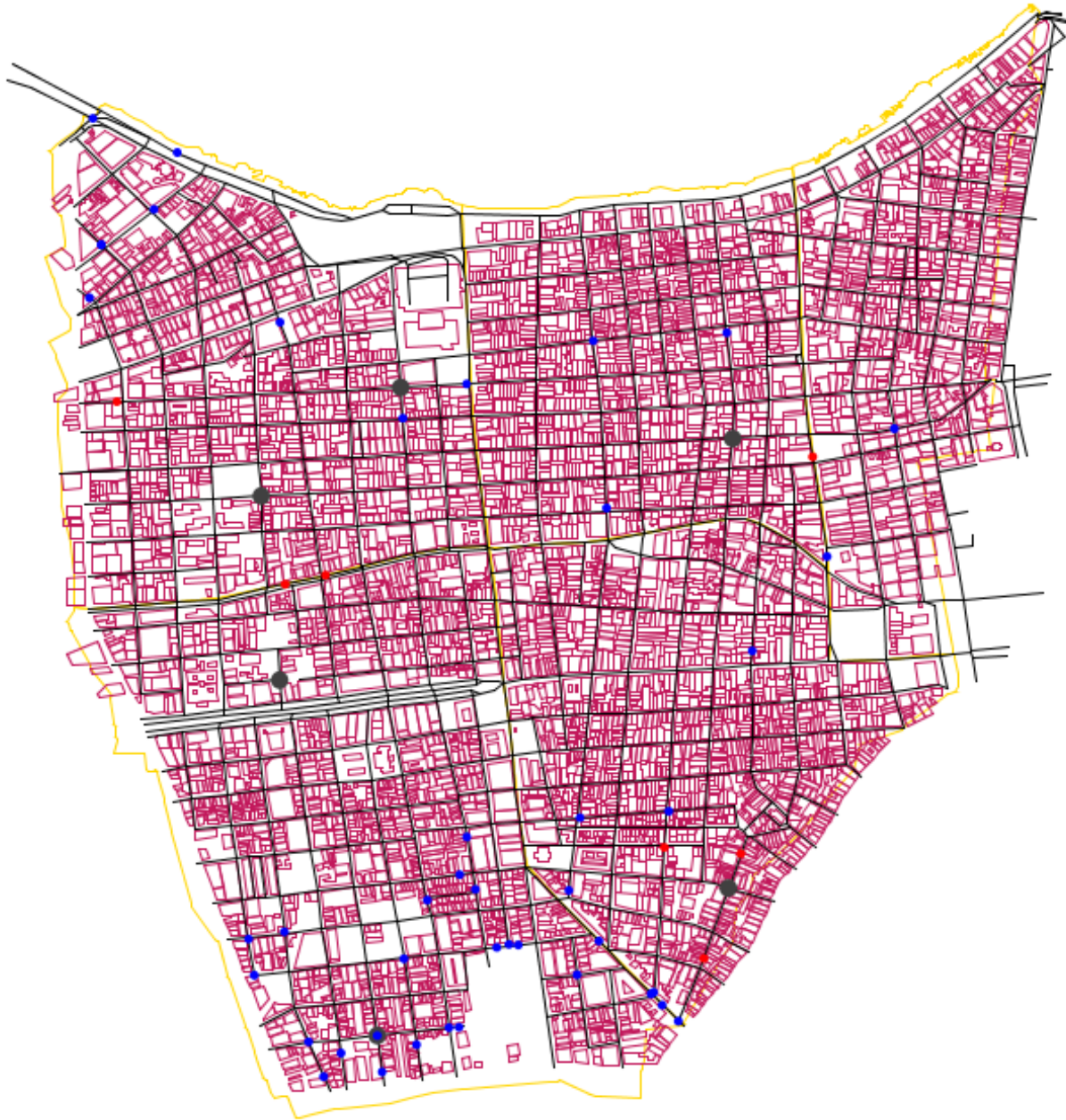


Figura 33: Visualización actual del entorno de la simulación.

Anexo 2: Visualización del entorno general

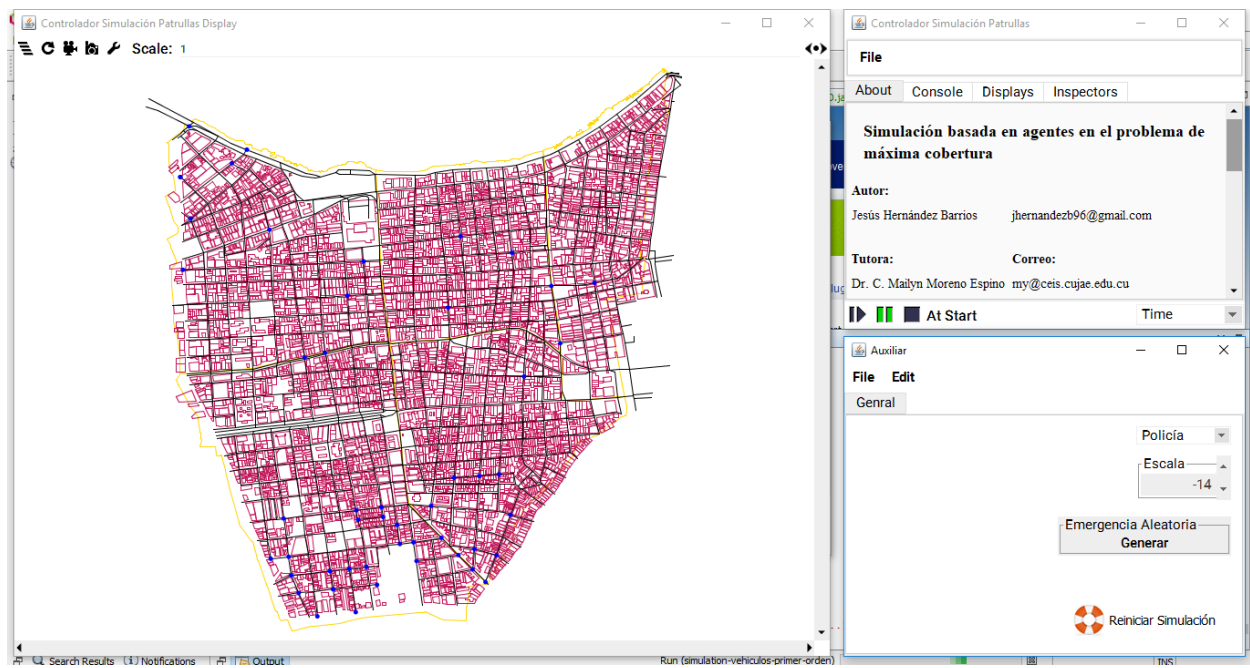


Figura 33: Visual Entorno General