Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería
Maestría y Doctorado en Ciencias e Ingeniería

Herramientas de Software Basadas en Agentes Inteligentes para la Integración de Sistemas Multi-Agente y Simulación Basada en Agentes

TESIS

Para Obtener el Grado de

Maestro en Ciencias

Presenta:

Josué Miguel Flores Parra

Bajo la dirección de:

Dr. Manuel Castañón Puga

Co-dirigido por:

Dra. Carelia G. Gaxiola Pacheco

Tijuana, Baja California, México

Mayo del 2014

A todas esas personas importantes en mi vida, que siempre estuvieron presentes en cada paso de mi vida, a todas aquellas personas que hicieron todo en la vida para que yo pudiera lograr mis sueños, por motivarme, y darme la mano cuando sentía que el camino se terminaba, a ustedes por siempre mi corazón y mi agradecimiento. Por su bondad y sacrificio que me inspiraron a ser mejor cada día, a superar cada prueba impuesta por mi destino, por enseñarme a levantarme por mas grande que sea el problema. Y lo mas importante por confiar en mi, cuando en ocasiones ni yo mismo puedo hacerlo, con todo mi cariño está tesis se las dedico a ustedes. A mi papá, mamá, hermanos, amigos, maestros y a mi pequeño sobrino Brandon.

Universidad Autónoma de Baja California

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

COORDINACIÓN DE POSGRADO E INVESTIGACIÓN

FOLIO No. 118

Tijuana, B. C., a 8 de abril de 2014

C. Josúe Miguel Flores Parra
Pasante de: Maestro en Ciencias
Presente

El tema de trabajo y/o tesis para su examen profesional, en la Opción TESIS

Es propuesto, por los CC. Dres. Manuel Castañon Puga y Carelia Guadalupe Gaxiola Pacheco

Quienes serán los responsables de la calidad de trabajo que usted presente, referido al tema: Herramientas de Software Basadas en Agentes Inteligentes para la Integración de Sistemas Multi-Agentes y Simulación Basada en Agentes.

el cual deberá usted desarrollar, de acuerdo con el siguiente orden:

I.- INTRODUCCIÓN
II.- MARCO TEORICO
III.- JT2FIS FRAMEWORK
IV.- HERRAMIENTAS IMIX
V.- PRUEBAS Y RESULTADOS
VI.- CONCLUSIONES Y TRABAJO FUTURO

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Dr. Manuel Castañón Puga
Director de Tesis

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

Q. Noemí Hernández Hernández
Sub-Director Secretario

Dra. Carelia Guadalupe Gaxiola Pacheco
Co-Director de Tesis

Dr. Luis Enrique Palafox Maestre
Director

Agradecimientos

Deseo agradecer a CONACYT (Consejo Nacional de Ciencia y Tecnología), por su apoyo económico durante los dos años de Maestría en Ciencias e Ingeniería. A la Universidad Autónoma de Baja California, por permitirme ser parte de está gran institución. Al cuerpo docente de la Facultad de Ciencias Químicas e Ingeniería de la Universidad Autónoma de Baja California por el apoyo brindado en el transcurso del periodo académico.

Al Dr. Manuel Castañon Puga, mi director de tesis, gracias por confiar en mi, por todo su apoyo, por estar ahí en los momentos difíciles para ayudarme. Gracias por sus consejos y a su experiencia he aprendido demasiado, un ejemplo de persona capaz y dedicado.

A mi co-asesora de tesis y sinodales: Dra. Carelia G. Gaxiola Pacheco, Dra. Dora Luz Flores Gutierrez, Dr. Juan Ramon Castro Rodriguez, Dr.Guillermo Licea Sandoval, Dr. Luis Guillermo Martinez Mendez, gracias por darme la oportunidad, por el apoyo recibido durante la realización de este proyecto y por el tiempo que me han dedicado para leer este trabajo.

A mi madre, por ser mi amiga y compañera que me ha ayudado en cada paso que he dado en mi vida, gracias por estar conmigo en todo momento. Gracias por la paciencia que me has tenido para enseñarme, por el amor que me das, por los regaños que me merecía pero no entendía.

A mi padre, porque gracias a él sé que la responsabilidad se debe vivir como un compromiso de dedicación y esfuerzo, por enseñarme que no importa que tan fuerte es la caída

siempre debemos de levantarnos.

A mis hermanos, que me han enseñado a salir adelante. Gracias por su paciencia, gracias por preocuparse por su hermano menor, gracias por compartir sus vidas, pero sobre todo, gracias por estar en otro momento tan importante en mi vida.

A mi familia, ustedes queridos abuelitos, tíos y primos, porque de una u otra forma, con su apoyo moral me han incentivado a seguir adelante, a lo largo de toda mi vida.

A todos, mis amigos y amigas que me han brindado desinteresadamente su valiosa amistad; gracias por ser la sal que condimenta mi vida. Gracias por apoyarme en todos los momentos difíciles, por aconsejarme y preocuparse por mi, gracias por ser mis hermanos sin la necesidad de compartir un lazo de sangre.

Gracias a todos aquellos que no están aquí, pero que me ayudaron para que este gran esfuerzo se volviera realidad.

Flores Parra Josué Miguel

Resumen

En este trabajo se presenta un conjunto de herramientas de software que pueden ser usadas para desarrollar aplicaciones inteligentes Java, y utilizarlas en simulaciones basadas en agentes. Este trabajo está motivado por la necesidad de crear herramientas que permitan el desarrollo de aplicaciones inteligentes de forma más eficiente, para esto se propone un conjunto de bibliotecas de clases y componentes visuales desarrollados en Java.

Se presenta la biblioteca de clases `JT2FIS` desarrollada en Java para sistemas difusos que pueden ser utilizados para construir sistemas de inferencia difuso por intervalos tipo 2. Además de la biblioteca de clases `JT2FISClustering` para construir sistemas de inferencia difusos tipo 2 a partir de diferentes métodos de agrupamiento de datos.

Dentro de este conjunto de herramientas se encuentran `JT2FISPanel` y `JT2FISClusteringPanel`, componentes visuales Java para sistemas de inferencia difuso por intervalo tipo 2 que se pueden utilizar para construir aplicaciones inteligentes desarrolladas en Java. En este trabajo se describen las principales características y funcionalidades; y se muestran interfaces de usuario con el fin de comparar los componentes desarrollados con otras herramientas existentes.

También se presenta `Wíinik` una herramienta centrada en el modelado de estructuras para simulaciones con herramientas visuales de software para ayudar a los investigadores a diseñar sistemas multi-agentes en un mayor nivel de abstracción. Está herramienta propuesta permite que visualicemos las relaciones entre los agentes y añadir una base de reglas como un

atributo para cada agente. Utiliza un sistema de inferencia difusa tipo 2 con el fin de añadir incertidumbre a cada agente. Su principal objetivo es exportar este conjunto de estructuras a Netlogo que es un entorno de simulación orientado a agentes, utilizado ampliamente para las ciencias sociales, haciendo uso de la extensión `JT2FISNetLogo`.

Abstract

This paper introduces a software toolkit that can be used to develop smart Java applications, and use agent-based simulations. This work is motivated by the need to create tools that enable the development of intelligent applications more efficiently , so that a set of class libraries and visual components developed in Java is proposed.

In this work is presented `JT2FIS` developed in Java por fuzzy systems than can be used to build interval type-2 fuzzy inference systems. Besides the class library `JT2FISClustering` to build interval type-2 fuzzy inference systems from different methods of data clustering.

Within this set of tools are `JT2FISPanel` and `JT2FISClusteringPanel`, a Java visual components for interval type 2 fuzzy inference systems that can be used to build Java intelligent applications. The main features and functionalities are described in this work; and we show user interfaces in order to compare the developed components with an existing tools.

It also presents `Wíinik` a tool focused on the modeling of structures for simulations with visual software tools to help researchers to design multi-agent systems at a higher level of abstraction. It is proposed tool allows us to visualize the relationships between agents and add a rule base as an attribute for each agent. Using a type 2 fuzzy inference system in order to add uncertainty to each agent. Its main objective is to export this set of structures a Netlogo which is a simulation environment oriented agents, widely used in the social sciences, using the extension `JT2FISNetLogo`.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los sistemas multi-agente (MAS; del inglés Multi-Agent System) están siendo ampliamente utilizados tanto en el área tecnológica como en el área científica ya que permiten modelar el comportamiento de usuarios en un sistema ubicuo o el de los individuos en la sociedad, entre otras entidades. Características del comportamiento humano, como lo son la adaptabilidad, la movilidad, el aprendizaje, el razonamiento y la personalidad, pueden ser modeladas mediante sistemas multi-agente, para simular problemas de la vida real[1].

Este trabajo de investigación está enfocado principalmente en producir recursos de investigación científica basados en modelos computacionales, el cual consiste en desarrollar herramientas para programar y configurar agentes de software. Para lograr esto, es necesario experimentar con varias de las tecnologías disponibles para el modelado y la implementación. Además se pretende presentar la simulación como un medio para modelar los fenómenos reales y como un instrumento para la investigación.

1.1. Antecedentes

En los últimos años la simulación se ha convertido en una de las principales herramientas para la búsqueda de la investigación social debido a su capacidad para explorar y validar los fenómenos sociales [1].

En la actualidad el modelado de sistemas sociales realistas no se puede lograr al recurrir a un solo tipo de arquitectura o de práctica. Los sistemas sociales contienen múltiples componentes que están estrechamente relacionados entre sí, lo que presenta múltiples obstáculos en la construcción de modelos de la realidad [2].

Un sistema social se considera un sistema complejo. Estas pueden ser vistas como sistemas dentro de sistemas. Estos sistemas no se pueden entender mediante la adopción de sus partes de forma independiente [3]. Un cambio en una de las partes puede afectar a una o más partes del sistema. La interacción interna y externa es muy importante en la representación de estos sistemas complejos[4].

La tecnología de agentes se adapta bien a los problemas de la vida real y al modelado de características propias del comportamiento humano, como lo son la adaptabilidad, la movilidad, el aprendizaje, el razonamiento y la personalidad [1].

En las sociedades artificiales, un trabajo interesante y desafiante es el mostrar la interacción entre los individuos, en un proceso donde la personalidad de los actores sale a la luz [1]. Los agentes y sistemas multi-agente son cada vez más reconocidos en la literatura como un paradigma adecuado para la ingeniería de sistemas con una arquitectura orientada a servicios (SOA; del inglés Service Oriented Architecture) y servicios web (WS; del inglés Web Service), ya que proporcionan un fondo conceptual y la ingeniería que naturalmente, se ajusta a muchas complejidades relativas a SOA / WS en un alto nivel de abstracción [5]. Pero a pesar de que el paradigma MAS no se limita a ninguna disciplina en particular, todavía no existe

una tecnología totalmente desarrollada [1].

Se proponen las herramientas orientadas a agentes, en base a las múltiples ventajas mencionadas por Mauro Dragone[6], entre las cuales destacan: (i) la arquitectura resultante que se pueden extraer de la gran cantidad de trabajo llevado a cabo dentro de la comunidad MAS, tanto en términos de conocimientos teóricos, así como metodologías de ingeniería, herramientas y middleware, (ii) el diseño del sistema se puede simplificar, porque el diseñador no tiene que especificar todas las interacciones en tiempo de diseño ya que algunos pueden ser manejados de manera autónoma por los propios agentes, y (iii) la solidez del sistema aumenta potencialmente debido a que estas interacciones se manejan en tiempo de ejecución, permitiendo que el sistema en su conjunto se adapte mejor a las condiciones ambientales [6].

1.2. Protocolo

1.2.1. Planteamiento del problema

Las ciencias de la computación social en general, y las simulaciones sociales, en particular, no han desarrollado mucho interés en la metodología de los programas de investigación a partir de una filosofía de la perspectiva de la ciencia. Esto está en contraste con la ciencia matemática social de una generación anterior, que desarrolló un gran interés sobre el mismo tema. La discusión de la metodología de simulaciones sociales complejas es constructiva en el sentido de poner de relieve los desafíos importantes, pero no se centra en el agente basado en modelos, un nuevo marco de complejas simulaciones sociales, junto con las redes sociales [7].

El punto de partida para el diseño e implementación de cualquier modelo formal se caracteriza por un equilibrio entre la teoría y la evidencia [8]. Los investigadores tienden a trabajar

de forma aislada, el diseño de todos sus modelos desde cero y presentación de sus resultados sin que nadie más pueda reproducir lo que encontraron. Existen riesgos considerables si todo el mundo sólo funciona en su propio modelo. Parte de la razón es que los modelos tienden a ser muy atrayentes en especial a la persona que ha construido el modelo. Lo que se necesita es una tercera persona para comprobar los resultados. Sin embargo, no siempre está claro cómo las personas, que son los modeladores, puede interpretar o utilizar esos resultados, ya que es muy difícil de replicar los modelos de simulación de lo que se informa en las publicaciones [9].

Los grandes proyectos computacionales, a menudo multidisciplinarios, requieren de una metodología de investigación científica que parece ser cualitativa y organizada de manera distinta, aunque basada en simples simulaciones sociales que tienen un alcance limitado, y por lo general se desarrollan a través del trabajo de un solo investigador. Grandes proyectos que contienen complejas simulaciones sociales requieren características metodológicas más robustas para que puedan involucrar a un gran número de participantes, disciplinas, instituciones, y un largo período de tiempo. Una diferencia importante radica en la interdisciplinariedad de la investigación, las preguntas no pueden ser respondidas por el conocimiento de una sola disciplina, y por el cálculo (a diferencia de matemáticas), el formalismo, en especial la orientación a objetos para el modelado [7].

El programa de desarrollo debe partir de algún modelo o modelos más simples (la construcción de bloques) y luego sistemáticamente proceder a través de una secuencia de modelos hasta que el modelo final se alcanza [7].

La complejidad social es causada por el deseo humano de una calidad de vida que es más alta de lo que se puede lograr individualmente. La acción colectiva es una necesidad frecuente de vida social, no una actividad opcional, para disfrutar de los avances en la calidad de vida, desde la caza de los grandes animales a la exploración espacial. La complejidad social en la

historia, desde la antigüedad hasta la actualidad, y en el futuro, es el resultado de la gente organizada en muchos niveles y escalas, desde pequeños grupos en países, a los sistemas mundiales de uso de la información y otros recursos y tecnologías para lograr y mantener las metas que no se puede alcanzar o ser sostenida sobre una base individual [15].

La complejidad social y el nuevo paradigma de la ciencia social computacional cubre tres áreas: (1) fundamentos de la identidad del campo, los pioneros, y supuestos epistemológicos, (2) conceptos básicos científicos, principios y temas de la naturaleza y la dinámica de la complejidad social, y (3) cuestiones prácticas relación con los recursos necesarios, los programas de enseñanza, centros de investigación, infraestructura e instituciones que van desde lo local a lo global [10].

El mundo social consiste en primer lugar de personas, ideas, humanos de artefactos, y sus relaciones, no en las variables y ecuaciones. Las entidades sociales pueden ser modeladas como objetos de cálculo que encapsulan los atributos y operaciones. Variables y las ecuaciones son necesarias, pero sólo después de la ontología fundamental de los objetos y las asociaciones se identifican. El modelado orientado a objetos y la programación es la formal, "lengua franca" de ciencias de la computación social, similar a la del cálculo infinitesimal de la física [10].

Las ciencias sociales se refiere a cómo la gente piensa (psicología), a la riqueza (la economía), en como se relacionan entre sí (sociología), como gobernarse a sí mismos (ciencia política), y crean la cultura (antropología). "Agentización" es el proceso de la formalización de una teoría social como un modelo basado en agentes [11].

Con base en estas ideas, las orientaciones futuras de la complejidad basada en la ciencia social debe incluir la maduración de los conocimientos existentes, pero recientes, así como la creación de nuevos conocimientos a través de métodos computacionales y afines (análisis de la visualización), incluyendo modelos híbridos. Las principales teorías sociales deben ser "agentizadas", analizadas y probadas con datos empíricos de la cognición humana de las

relaciones interpersonales [10].

1.2.2. Justificación

En el mundo se enfrentan diferentes procesos de crisis, como la crisis ecológica, la crisis energética, la crisis alimentaria, la crisis económica, etc. [12]. En las últimas tres décadas, México ha transitado por un proceso importante de cambios estructurales: económicos, sociales y políticos[13].

El crecimiento urbano, la escasez de recursos naturales, la diversidad cultural y étnica producida por la migración y el crecimiento poblacional, la violencia, el narcotráfico, la colindancia con un país desarrollado, como lo es los Estados Unidos, entre otras características [12] [14] [15], plantean la necesidad de lidiar con estos problemas tan complejos a través de diferentes estrategias de investigación científica para proponer soluciones que ofrezcan un desarrollo sustentable en el futuro.

Entre muchas de las estrategias que pudieran tomarse para comprender estos fenómenos está la simulación a través de computadora. La simulación por computadora es una herramienta que algunos científicos en la actualidad han estado introduciendo de manera paulatina en sus actividades de investigación y con diferentes grados de aplicación. Las computadoras tienen la capacidad de facilitar las tareas de explorar, analizar, almacenar y reproducir fenómenos de manera artificial, cuya utilidad los científicos han ido adoptando y aceptando hasta cierto punto [16].

Las aplicaciones que pueden tener estas herramientas son muy amplias ya que puede producir una gran variedad de resultados educativos: conceptos y habilidades, cooperación y competición, pensamiento crítico y decisión; empatía, conocimiento de sistemas sociales, políticos y económicos; sentido de la eficacia; conciencia del factor azar, y hacer frente a las consecuencias de los actos.

Una de sus aplicaciones seria en el área de la sociología ya que se podría lograr un acercamiento mayor a los problemas complejos, facilitar el modelado computacional a través de esta herramienta implementando agentes inteligentes capaces de reproducir mejor las características humanas y sus organizaciones.

La investigación local y regional puede verse beneficiada si se incorpora en su metodología de investigación el modelado computacional al contar con herramientas de simulación basadas en sistemas multi-agente y capacidad de cómputo.

1.2.3. Hipótesis

- La existencia de un marco de trabajo de desarrollo de software que de soporte a un sistema basado en agentes inteligentes permite el uso de sistemas multi-agente como base para una metodología de simulación de sistemas complejos.

1.2.4. Objetivos

Objetivo general

Desarrollar un conjunto de herramientas y componentes que ayuden en la simulación basada en agentes de sistemas complejos utilizando agentes inteligentes y sistemas multi-agente.

Objetivos específicos

Para la consecución de este objetivo principal se deben de considerar algunos objetivos particulares, como los que a continuación se presentan:

1. Investigar las diferentes herramientas existentes para la creación de modelos utilizados

en simulación basada en agentes.

2. Investigar las diferentes bibliotecas de clases y componentes para la construcción de sistemas inteligentes.

3. Simplificar la interpretación de los niveles o dimensiones a través del modelado basado en agentes utilizando agentes cognitivos.

4. Evaluar los resultados de este conjunto de herramientas, para concluir si los modelos pueden generar simulaciones con un grado alto de experimentación significativa y comprensible para los usuarios.

1.2.5. Metas

Metas de desarrollo tecnológico

1. Una biblioteca de clases para construir sistemas de inferencia difusos.

2. Un componente visual para la interacción con sistemas de inferencia difusos.

3. Un algoritmo para implementar un método de minado capaz de generar un sistema de inferencia difuso.

4. Un componente visual, que ayude en el proceso de minería de datos.

5. Una extensión para implementar sistemas de inferencia difusos dentro de una herramienta de simulación.

6. Una herramienta de modelado que ayude a diseñar estructuras complejas de agentes cognitivos.

1.2.6. Metodología

El proyecto sigue un modelo de desarrollo en espiral, por medio del cual se lleva a cabo refinaciones progresivas que retroalimentan las nuevas fases del ciclo de diseño, implementación y pruebas. Como se muestra en la figura 1.1.

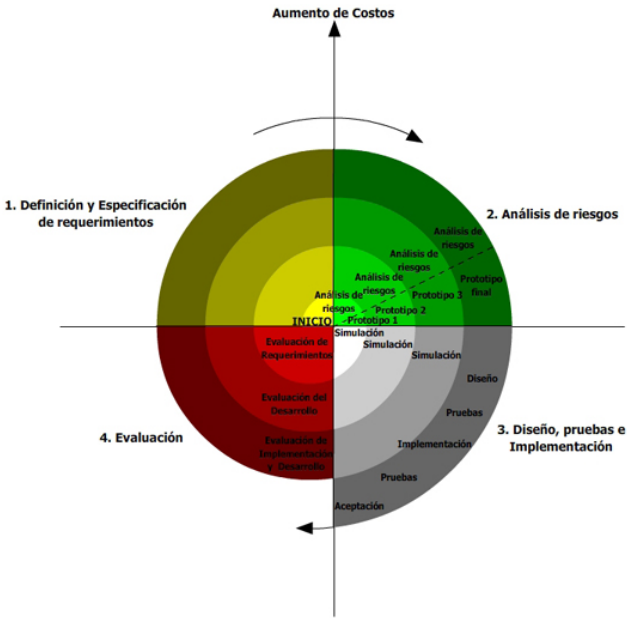

Figura 1.1: Metodología de Desarrollo en Espiral.

Con esta metodolgía se construyeron herramientas de software, partiendo de sistemas simples que ciclos posteriores se fueron refinando para crear herramientas mas complejas. A contiuación, se describen brevemente cada una de las fases y sus ciclos.

Fase 1: Planeación.

En la fase de planeación se lleva a cabo principalmente una revisión del estado del arte y la formulación de los antecedentes y requerimientos de sofware necesarios para comenzar el proceso de desarrollo.

Ciclo 1. Investigación del estado del arte. En este ciclo se reúne la bibliográfia disponible sobre avances relacionados a las diferentes herramientas existentes para la creación de modelos utilizados en simulación social, componentes y bibliotecas de clases para la creación de sistemas de inferencia difusos y los diferentes algoritmos de agrupamientos de datos existentes, con el fin de entender el trabajo previo.

Ciclo 2. Formulación de los antecedentes. En este ciclo de reviso y estudió la información recopilada en el ciclo anterior para establecer los antecedentes y requerimientos iniciales del trabajo de investigación.

Fase 2: Desarrollo.

En la fase de desarrollo se llevó a cabo primeramente la construcción de bibliotecas de clases base. Posteriormente se construyeron componentes visuales útiles para construir sistemas mas complejos con interfaces de usuario gráficas. Finalmente se construyeron prototipos de herramientas de modelado con interfaces de usuario gráficos que utilizaron en parte los componentes desarrollados.

Ciclo 1: Desarrollo de la biblioteca de clases de sistemas de inferencia difusa. En este ciclo de desarrolló una biblioteca de clases que implemente el paradigma de programación orientada a objetos y diferentes de patrones de diseño de software, que permita a programadores crear sistemas de inferencia difusos de una forma rápida y clara.

Ciclo 2: Desarrollo de extensión NetLogo implementando la biblioteca de clases de sistemas de inferencia difusa. En este ciclo se desarrolló una extensión NetLogo para agregar sistemas de inferencia difusos en el modelado de simulaciones sociales, utilizando NetLogo e implementado la biblioteca de inferencia difusa desarrollada en la Fase 3.

Ciclo 3: Implementación de un algoritmo de minería de datos en Java. En este ciclo se implementó un algoritmo de minería de datos lo suficientemente eficiente para el trato de grandes cantidades de información, que permita crear sistemas de inferencia difusa gastando un mínimo de requerimientos de hardware.

Ciclo 4: Desarrollo de los componentes visuales. En este ciclo se desarrolló diferentes componentes visuales, utilizando patrones de diseño y refactorización de código, que fueran fáciles de utilizar o agregar en diferentes proyectos de programación, para ayudar a los programadores en el uso de sistemas de inferencia difusos.

Ciclo 5: Desarrollo de herramientas de modelado. En este ciclo se desarrolló diferentes prototipos de herramientas visuales, utilizando patrones de diseño y refactorización de código, que sean fáciles de utilizar en diferentes proyectos de modelado, para ayudar a los investigadores en el uso de agentes cognitivos.

Fase 3: Pruebas.

En la fase de pruebas se llevó a cabo primeramente una planificación de casos de estudio que ayudaron a establecer escenarios de prueba para los diferentes desarrollos. Después se diseñaron casos de prueba que fueron aplicados a las bibliotecas y componentes propuestos. Los resultados fueron analizados y descritos de manera comparativa considerando otras bibliotecas y/o herramientas similares.

Ciclo 1: Planificación de casos de estudio. En este ciclo se propusieron diferentes casos de estudio, que sirvieron para establecer escenarios de prueba para cada uno de los componentes y bibliotecas desarrolladas.

Ciclo 2: Análisis de resultados. En este ciclo se propusieron casos de prueba que fueron aplicados a cada uno de los componentes y bibliotecas propuestas. Los resultados obtenidos fueron analizados comparativamente con otras bibliotecas y herramientas similares, como el caso de Matlab y Juzzy, ambas herramientas disponibles para la construcción de sistemas de inferencia difusos.

Fase 4: Cierre.

En la fase de cierre se llevará a cabo la publicación de resultados.

Ciclo 1: Publicación de resultados. En este ciclo se publicarán resultados en conferencias y revistas científicas conforme se tengan avances.

1.3. Como está organizado este documento.

Este documento de tesis está organizado de la siguiente manera: En el primer capítulo se hace una introducción al documento y se expone un resumen de los temas principales tratados en esta tesis. Se hace un resumen del protocolo de investigación motivo de esta tesis. En el capítulo dos se hace una revisión exhaustiva del estado del arte de los temas principales de esta investigación. Un marco teórico es descrito con detalle. En el capítulo tres se describe en detalle el marco de trabajo para el desarrollo de software denominado JT2FIS Framework. En el capítulo cuatro se describe en detalle la herramientas de modelado denominadas IMIX Tools, en particular, la herramienta Wíinik. En el capítulo cinco se describe en detalle los casos de estudio considerados en este proyecto. También se describen los resultados obtenidos en los casos de prueba y el resultado del análisis comparativo con otras herramientas similares. En el capítulo seis se exponen las conclusiones y el trabajo futuro de esta investigación. Finalmente,

en una sección de bibliografía, se lista una relación de todas las referencias citadas en este documento.

Capítulo 2

Marco teórico

Esta sección repasa los temas base de complejidad como el enfoque principal desde el punto de vista científico, modelado computacional como la estrategia principal desde el punto de vista de ciencias computacionales, simulación social como el campo principal de aplicación y herramientas de modelado y simulación como los medios actuales para desarrollar modelos y simulación.

2.1. Complejidad

La complejidad es un término que no tiene una definición clara [17, 18], el autor Warren Weaver en su artículo la ciencia y la complejidad [19] trata de abarcar esta definición desde los problemas de la simplicidad y menciona que en épocas pasadas, la ciencia física aprendió a simplificar las variables, y con esto se crearon nuevas tecnologías: como el teléfono, la radio, el automóvil, el avión, la turbina, la planta moderna de energía hidroeléctrica, etc, y por parte de la biología y medicina los problemas se analizarían diferentes ya que difícilmente pueden mantener una constante. Los seres vivos son más propensos a presentar situaciones

de mucha interconexión y en que la cantidades importantes no son cuantitativas, Por ello los problemas biológicos y médicos a menudo implican una consideración de un conjunto más complejamente organizado [19].

Otros autores como Christoph Adami [17] se dice que existe definiciones de la complejidad en los sistemas dinámicos y para los organismos biológicos, pero tienen desventajas de carácter conceptual o práctico, donde desarrollan modelos matemáticos para demostrar su teoría.

La definición que da Heylighen [18] de lo que es la complejidad lo extrae de la misma raíz latina "complexo", que significa enredados, entrelazados, que abarca. Él interpreta este significado, como que a fin de tener un complejo, es necesario dos o más componentes distintos que están conectados de manera que sean difíciles de separar, produciendo propiedades emergentes. Un sistema se vuelve más complejo cuando el número de distinciones (componentes distintos, estados, o aspectos) y el número de relaciones o conexiones aumenta. El problema con esta definición es que no da lugar a un número único o grado que permitan medir objetivamente la complejidad de un fenómeno que es. La razones que las distinciones y las conexiones no son objetivamente dadas, fácilmente entidades contables: ya que existen en diferentes niveles, en diferentes dimensiones y en diferentes tipos. A pesar de esta limitación, esta definición tiene algunas características favorables: es sencilla e intuitiva [18].

Morin en su artículo trata de dar una explicación del por qué la problemática de la complejidad apareció tan tarde [20]. La ciencia clásica rechazó complejidad en virtud de tres motivos de principios fundamentales:

1. El principio del determinismo universal, que establece que no sólo se deben conocer todos los acontecimientos pasados, sino también prever todos los eventos en el futuro.

2. El principio de reducción, que consiste en conocer cualquier señal de compuesto a partir del conocimiento sólo de sus elementos básicos que constituyen.

3. El principio de la separación, que consiste en el aislamiento y las dificultades cognitivas que separa unos de otros, que conducen a la separación entre disciplinas.

Con estos principios, se llegó a extremadamente brillante, importante, y la evolución positiva de los conocimientos científicos hasta el punto donde los límites de la inteligibilidad que se constituyó fue más importante que sus aclaraciones.

Sin embargo autores como Klein maneja el término de complejidad en el cual debe verse en una relación interdisciplinaria [21].Ya que el tratar un tema complejo es demasiado amplio para ser tratado adecuadamente por una sola disciplina o profesión. La colaboración interdisciplinaria es necesaria porque los problemas complejos no pueden ser resueltos simplemente con el conocimiento de una sola disciplina. Por lo que este autor afirma es que la complejidad no es menos plural que la interdisciplinariedad. La complejidad y la interdisciplinariedad están unidas en una amplia gama de prácticas, a partir de los estudios literarios, física, biología, políticas públicas y estudios ambientales [21]. Una vez que se describe como un fundamento o estructura lineal, el conocimiento hoy en día se representa como una red o una red con varios nodos de conexión, y un sistema dinámico.

Heylighen menciona que la ciencia de la complejidad sólo se convertiría en muchas técnicas de modelado avanzado que capturan con mayor o menor éxito solo diferentes aspectos de los sistemas complejos, pero sin abarcar una teoría general para todo. Y aunque los físicos han comenzado a centrarse en la complejidad, aplicando una gama de herramientas matemáticas para el análisis de los sistemas sociales, económicos o biológicos. Esto puede producir suficiente de ideas interesantes a corto plazo, pero a largo plazo necesitan para tomar conciencia que nunca se les proporcionará una ley absoluta de “leyes de la complejidad” que muchos aún están buscando [18]. Otro ejemplo de manejar la ciencia de la complejidad en estudios prácticos es el proporcionar herramientas para medir la integración, llamada agrupación funcional y diferenciación llamado complejidad neural que son aplicables a procesos reales de

los nervios. Esto lleva a formular criterios operativos para determinar si la actividad de un grupo de neuronas contribuye a la experiencia consciente. Estos criterios se incorporan a la hipótesis de núcleo dinámico, una propuesta relativa al comprobable sustrato neural de la experiencia consciente [22]. Otras investigaciones que han surgido de estudios multidisciplinarios y que hoy en día se encuentra en un estudio de los sistemas complejos es la cibernética [23] y modelos basados en agentes donde pueden realizar análisis de problemas de política con enfoques clásicos de modelos de predicción y optimización que se han utilizado en el software de apoyo a la toma de decisiones[24].

2.1.1. Sistemas complejos

Mitchelle y Newman mencionan que un "sistema complejo" es un grupo u organización que se compone de la interacción de muchas partes. Los sistemas complejos incluyen como por ejemplo el clima mundial, las economías y el sistema inmunológico. En esos sistemas las partes individuales llamadas "componentes" o "agentes" y las interacciones entre ellos conducen a menudo a gran escala de los comportamientos que no son fácilmente predecibles a partir de un conocimiento sólo de la conducta de los agentes individuales [25].

La estructura de un sistema complejo considera dos tipos básicos: sistemas simples y complejos. Los sistemas simples tienen dos jerarquías: las partículas y el sistema. En cambio los sistemas complejos deben tener una estructura que exige un mínimo de tres niveles jerárquicos: las partículas, los agentes y el sistema [26]

Existen varias maneras de considerar la complejidad y hacer frente a los sistemas complejos, el autor muestra un resumen de ocho teorías o formas de pensar acerca de la complejidad, lo que pone de manifiesto la diversidad de esos enfoques [27].

Matemática. La teoría moderna de la complejidad de Turing y Von Neumann, del teorema "lemma-theorem-proof structure". La llamada teoría de la complejidad de las ciencias de la computación, incluye la contribución de las clases de complejidad.

Teoría de la información. Medidas de complejidad e información en el espacio de Hamming como bits en términos de orden y aleatoriedad. Llama una teoría de límites, por ejemplo, no pueden ocupar más bits que el número de sinapsis en nuestra red natural.

Teoría ergódica. El estudio de mapas dinámicos disipativos incluidas las órbitas y determinístico sistema dinámico. Incluye la teoría del caos y la teoría de bifurcación.

Entidades artificiales. El estudio de entidades artificiales en los ordenadores, como autómatas celulares. Esta labor se traduce en simulaciones, como el juego de la vida, y la teoría del borde del caos en que la complejidad puede ser en su más alto entre la aleatoriedad y la regularidad.

Sistemas físicos aleatorios. Los sistemas que tienen la mecánica estadística de complejidad con complejo alto dimensional de atractores. Estos sistemas son típicamente no ergódicos, como al azar, colectores de percolación, la localización de vidrio hilado y redes neuronales.

Auto-organizaciones criticas (SOC). Sistemas en la que están impulsadas por algunos conservadores cantidad de manera uniforme a gran escala, son capaces de disipar sólo a fluctuaciones microscópicas, las fluctuaciones en todas las escalas (a diferencia de ciclos de estabilidad o de fallo del sistema, aunque los sistemas no se consideran de adaptación).

Inteligencia artificial. Para investigar sistemas complejos adaptativos de la construcción; como los algoritmos genéticos y sistemas de clasificador.

Wetware El intento de investigar los sistemas complejos adaptativos mediante el estudio de ellos. El intento de comprender sistemas complejos como el cerebro, sin tratar de especificar un conjunto de principios subyacentes. El enfoque naturista de estudiar los sistemas.

2.1.2. Sistemas complejos adaptativos

El campo de sistemas complejos adaptativos (CAS; del inglés Complex Adaptive Systems) es de aproximadamente veinte años, después de haber sido establecido por físicos, economistas, y científicos que estudian la complejidad. Este campo ha generado contribuciones por ejemplo para los algoritmos genéticos, los sistemas de clasificación, los ecosistemas y simulaciones que ayudan en los campos de la computación evolutiva y la vida artificial.

El marco se deriva de muchos sistemas complejos naturales y artificiales que han colaborado en la investigación en tan diversos campos de estudio como psicología, antropología, evolución genética, ecología y la teoría de gestión empresarial [27]. Aunque el campo de los sistemas complejos es relativamente joven, el sentido de la aparición de este término se asocia habitualmente con los fenómenos micro que a menudo dan lugar a fenómenos macro [28].

Otra definición consiste en la no homogénealidad, que interactúan los agentes adaptativos. Adaptativo se refiere capaces de aprender[29]. Para Brownlee un CAS es una red dinámica de muchos agentes los cuales pueden representar células, especies, individuos, empresas, naciones actuando en paralelo, constantemente y reaccionando a lo que otros agentes están haciendo. El control de un CAS tiende a ser altamente disperso y descentralizado. Si hay un comportamiento coherente en el sistema, este tiene un crecimiento de competición y cooperación entre los agentes mismos. El resultado total del sistema proviene de un enorme número de decisiones hechas en algún momento por muchos agentes individuales[27]. El estudio de los sistemas adaptativos complejos, es un subconjunto de sistemas dinámicos no lineales, que se ha convertido recientemente en un foco importante de la investigación interdisciplinaria

como en las ciencias sociales y naturales [30].

En general, los ejemplos de CAS se han extraído la mayoría de los sistemas estudiados en biología, la sociología y la economía. Algunos ejemplos citados a menudo incluyen: el desarrollo de embriones, la función del sistema inmune adaptativo, ecología, evolución genética, el pensamiento y el aprendizaje en el cerebro, los sistemas meteorológicos, las economías de mercado, los sistemas de comercio, los sistemas sociales, culturas, la política, los sistemas de tráfico, los insectos enjambres, los que llegan de las aves, la aplicación de nuevas ideas, las pruebas de las teorías científicas, y convertirse en bacterias resistentes a un antibiótico. Los modelos de simulación de ordenador juegan un papel muy importante en la investigación de CAS, donde el sistema se reduce a su más simple aspecto esencial. Estos mismos modelos de simulación demuestran los rasgos de los sistemas complejos adaptativos y, por tanto, proporcionar un terreno fértil para la experimentación controlada. Algunos métodos de modelización utilizado y desarrollado para este propósito incluyen autómatas celulares (AC; del inglés Cellular Automata), el agente basado en modelos (MBA; del inglés Model Based on Agents), redes neuronales artificiales (ANN; del inglés Artificial Neural Networks), algoritmos genéticos (GA; del inglés Genetic Algorithms), y los sistemas clasificadores de aprendizaje (LCS; del inglés Learning Classifier Systems)[27].

Una entidad que se adapte a su entorno; pero a la vez este entorno es cambiante constantemente este fenómeno se conoce comúnmente como paisaje adaptativo. Lo que se ha observado en muchos casos menos en la literatura de la complejidad es que cuando algo se agrega a un entorno que puede permitir algo más que añadir este último, algo que no podría haber existido en ese ambiente antes de la adición anterior. Esta es una extensión de los conceptos de la ecología, la biología y las ciencias sociales. Un término para este fenómeno de la literatura ecología, es la sucesión. Históricamente la sucesión se ha tomado para referirse a una secuencia bastante rígida de las comunidades de especies, por lo general conduce a lo que se llama un clímax o (menos espectacular) un estado de equilibrio[28].

Propiedades y mecanismo de los sistemas complejos adaptativos

Para Holland y Brownlee [31, 27] las propiedades de los sistemas complejos son:

Agregación. (propiedad) La complejidad surge de la interacción de los componentes más pequeños, que a su vez pueden ser los productos de los sistemas.

Etiquetado. (mecanismo) Se diferencian los agentes y posee una forma en que discriminan a los agentes con propiedades particulares.

No linealidad. (propiedad) en los agentes interactúan y no dinámicos.

Flujos. (propiedad) Agentes organizados en redes de interacción que puede desencadenar (flujo).

Diversidad. (propiedad) Agentes evolutivos para cubrir diversos nichos, que se definen por las características de las interacciones agente. El concepto de un nicho sobrevive el que habitan los agentes, y la evolución de los nichos tiene un mayor impacto sobre el sistema evolutivo de los agentes (los niveles de abstracción y control).

Modelos internos. (mecanismo) de agentes se haya cambiado a través de interacciones, y los cambios sesgo acciones futuras (agentes de adaptación). Las representaciones internas poseen información como la forma de explotar la regularidad de sus interacciones, sin necesariamente definir explícitamente la regularidad.

Bloques de construcción. (mecanismo) Los componentes son reutilizados para fines múltiples.

Gell-Mann especifica un ciclo de CAS que se compone de seis elementos básicos y cuatro en relación con las cuestiones. Él define en detalle una serie de preguntas relacionadas con la investigación de cada uno de los puntos[27]:

Granularidad gruesa (Coarse graining). (trade-off) entre la tosquedad de la capacidad de administración de la información y la finura de detalle adecuado a la información.

Identificación. Clasificación de las regularidades de la aleatoriedad de la información en el medio ambiente.

Comprensión. Percibe regularidades que se comprimen en su esquema.

Variación. Variación y mejora del esquema (la adaptación o evolución).

Aplicación. Uso de los sistemas de medio ambiente.

Selección. Selecciona las consecuencias de presiones selectivas en el mundo real, en el cual proporciona información de las partes afectadas.

Cuestiones Cuestiones relacionadas con los seis aspectos de un ciclo de vida del CAS incluyen [27]:

1. escalas de tiempo,
2. el sistema incluido como un componente de otros sistemas,
3. sistemas con los seres humanos en el bucle, y
4. sistema integrado de varios sistemas de co-adaptación.

Brownlee[27] da una definición de sistemas complejos como se estudió en la economía con seis propiedades, como redes no lineales de adaptación. Estas propiedades, al igual que Holland [31], son también citados como principios de normas CAS.

Interacción dispersa. Efectos emergente son el resultado de las acciones dispersas, posiblemente heterogéneas, los agentes que actúan en paralelo. La acción de un agente depende de las acciones previstas de un número limitado de otros agentes, y el estado agregado el conjunto limitado de agentes creados.

Ausencia de un controlador global. No se proporcionan los controles de la competencia y la coordinación entre los agentes. Acciones económicas están mediadas por las normas del medio ambiente.

Interacción transversal jerárquica. Un sistema tiene muchos niveles de organización y de interacción. Unidades de cualquier nivel de abstracción en el sistema de servir como "bloques de construcción" para la construcción de unidades en los altos niveles de abstracción. Las interacciones son más que jerárquica con enmarañada y transversales.

Continua. Adaptación comportamientos, acciones, estrategias y productos son revisados continuamente como los agentes individuales se acumulan experiencia, el sistema se adapta constantemente.

Novedad perpetua. Los cambios introducidos por la adaptación crear continuamente nuevas oportunidades para la exploración, el resultado está en curso, la novedad perpetua fuera de la dinámica de equilibrio. Debido a la continua adaptación y cambio, el sistema opera lejos del global óptimo y equilibrado. Las mejoras siempre son posibles y ocurren con

regularidad.

Dooley proporciona una definición condensada de CAS. En su definición, los agentes son los elementos base del sistema que se adaptan en respuesta a las interacciones. Las adaptaciones que se producen a operar el esquema de agente, una representación que define el agente de normas e interacciones. Agente de fitness está optimizado a nivel local en relación con el micro ambiente local. El flujo de información es no lineal. Los agentes están marcados (funciones heterogéneas) y los agregados de agentes heterogéneos pueden formar meta-agentes. Levin se diferencia de tres aspectos esenciales[27]:

Diversidad. Sosteniendo la diversidad y la individualidad de los componentes.

Las interacciones locales. Localizar las interacciones entre los componentes.

Selección. Un proceso autónomo que selecciona entre los componentes de un subconjunto de repetición o mejora, con base en los resultados de la interacción local entre los componentes.

Por otra parte autores realizan otra clasificación muy distinta para Jost [32] divide los sistemas complejos adaptativos en internos y externos lo cual los define:

Complejidad externa. La cantidad de entradas, información, energía obtenida del entorno que el sistema es capaz de manipular y transformar. Es importante que esto se puede medir como una entropía - y por lo tanto, en términos como de "energía". En este sentido, la complejidad externa es la complejidad de los datos.

Complejidad interna. Mide la complejidad de la representación de esta entrada por el sistema. En este sentido, su complejidad interna es la complejidad del modelo. El objetivo del sistema es manejar tantos datos como sea posible con un modelo tan simple como sea

posible.

2.1.3. Sistemas multi-agente

Existe diferentes definiciones de un sistema multi-agente, varios autores lo definen como un sistema en el que múltiples agentes autónomos, heterogéneos, interaccionan entre sí y con el entorno; cada uno buscando sus propias metas[33]. Sin embargo no tiene que ser heterogéneo para ser un sistema multi-agente; autores como Gómez [34] lo definen a partir de la construcción de programas que componen los sistemas distribuidos aplicando una tecnología íntimamente relacionada con la inteligencia artificial. Esta técnica es por medio de agentes siendo autónomos e inteligentes. Es cuando los sistemas distribuidos pasan a ser un sistema multi-agente.

Las propiedades de un sistema multi-agente nacen cuando Wooldridge desarrolla en su tesis doctoral una familia de lógicas para la representación de estas propiedades, en este documento construyó formalismos que pueden ser utilizados en la especificación y verificación para la acción cooperativa [35].

Los sistemas multi-agente constituyen un área de creciente interés dentro del área de la Inteligencia Artificial, por ser aplicable a la resolución de problemas complejos no resueltos de manera satisfactoria mediante técnicas clásicas. Existen aplicaciones basadas en este nuevo paradigma empleadas en distintas áreas [36], tales como control de procesos, procesos de producción, control de tráfico aéreo, aplicaciones comerciales, gestión de información, comercio electrónico, aplicaciones médicas, juegos, etc. [37]. Los sistemas multi-agente han sido cada vez más utilizados para modelar una gran variedad de fenómenos sociales. Desde hormigas hasta modelar países en la preparación para una guerra fría, la evolución en el sistema multi-agente permiten cada vez más la comprensión de los sistemas con la interacción de múltiples agentes[38].

2.1.4. Agencia distribuida

Para modelar correctamente el comportamiento humano complejo se debe definir un lienzo en el que múltiples dimensiones de la existencia pueden ser definidas. Los fenómenos emergentes son muy abundantes en los sistemas que están separados en niveles ontológicos distintos, como lo es en las sociedades humanas llenas de consumidores, coaliciones, las familias, las lealtades, las empresas, las leyes, los carteles, las instituciones gubernamentales [4].

Agencia distribuida(DA; del inglés Distributed Agency) es el nombre de un marco conceptual para la descripción de los sistemas adaptativos complejos. DA representa una teoría general del comportamiento y la estructura de la formación colectiva, que tiene la intención de redefinir la agencia y reflejarla en múltiples capas de información y la interacción, esto se puede observar en la Figura 2.1; a diferencia del enfoque tradicional en el que la agencia sólo se refleja en los agentes individuales, atomizadas y aisladas [39].

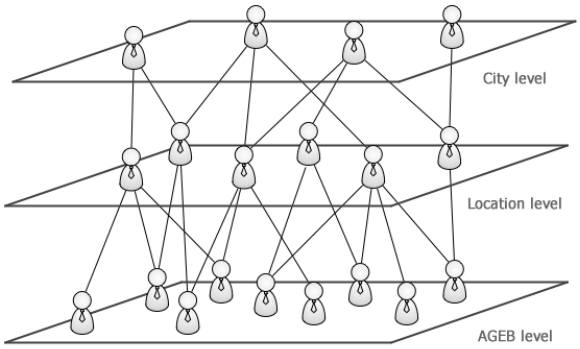

Figura 2.1: Ejemplo de agencia distribuida (Fuente: Suarez y Castañón-Puga, 2013).

El comportamiento en el marco de una agencia distribuida se expresa como el resultado de agentes contextualizadas que maximizan sus funciones de utilidad. ¿ Qué se entiende por una contextualización de los agentes? es que los agentes están formados por agentes con sus propias funciones de utilidad, y que los agentes a su vez forman parte de los agentes más grandes que ellos atraen a actuar de una manera u otra. En todos los niveles, el agente de

nivel superior o el medio ambiente tiene que tener la suficiente influencia al agente de menor nivel, que a su vez tiene que tener suficiente energía para completar la acción[39] esto se puede observar en la Figura 2.2.

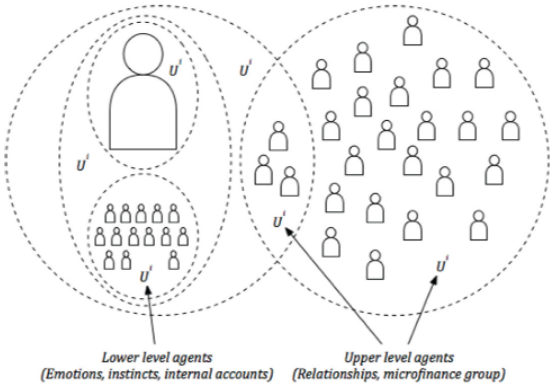

Figura 2.2: Contextualización de los agentes (Fuente: Suarez y Castañón-Puga, 2013).

El marco teórico puede servir como un punto de referencia para los investigadores sociales y computacionales comunicando propiedades estructurales y emergentes que son importantes para la comprensión de los fenómenos sociales y evolutivos, tales como empresas, las economías, los gobiernos y los ecosistemas.

DA implica que el investigador tenga que observar el comportamiento y luego usar un razonamiento de inducción hacia atrás para retratar las fuerzas que podrían haber dado lugar a las decisiones tomadas; y describe los procesos de selección y evolución utilizando una terminología común. DA plantea que para inducir el comportamiento de una situación debe tener en cuenta 2 cosas[39]:

1. Un suministro de agentes capaces.

2. Una demanda de este tipo de comportamiento.

2.2. Modelado computacional

Un modelo es una descripción simplificada de un sistema que es útil para la simulación o análisis. Los modelos computacionales han sido diseñados para aprovechar la eficiencia de la computación [40].

Los modelos basados en agentes son útiles para modelar la dinámica de sistemas que no están en equilibrio (aunque también se utilizan para estudiar el equilibrio). Y son especialmente útiles para entender las relaciones entre las decisiones individuales y el comportamiento del sistema[40].

Las organizaciones de MAS, se puede entender como entidades complejas donde una multitud de agentes interactúan, dentro de un ambiente estructurado destinado a un propósito global. Las organizaciones de agentes se asocian a menudo con la idea de la apertura y la heterogeneidad en el MAS. De entornos heterogéneos que plantean nuevas exigencias en el diseño e implementación MAS incluyendo la integración de las perspectivas globales e individuales y la adaptación dinámica de los sistemas a los cambios ambientales. Como los sistemas de crecimiento que incluyen cientos o miles de agentes [41].

Las teorías formales son necesarias para describir la interacción y la estructura organizativa y entender la relación entre las funciones de organización de los agentes.

En la sociedad MAS el diseño proporciona estructuras para el desarrollo de modelos organizativos, pero también puede permitir el análisis de cómo las organizaciones naturales puede ser ampliada o mejorada. Las herramientas necesarias para la modelización, la simulación de las organizaciones agente proporcionará nuevos conocimientos sobre teoría de la organización [41].

2.2.1. Modelado de sistemas complejos

Para Ahmed [29] las prácticas habituales de modelado son:

1. Ecuaciones diferenciales ordinarias, ecuaciones en diferencias y ecuaciones diferenciales parciales.
2. Los autómatas celulares.
3. Teoría de juegos evolutiva.
4. Modelos basados en Agentes.
5. Redes.
6. Fraccional de cálculo.

Existen varios problemas al analizar un sistema complejo como la interacción de los individuos están obligados a cambiar de entornos, una nueva teoría matemática para estudiar esta evolución son los sistemas complejos adaptativos. El cual encuentra un campo emergente en la dinámica de la adaptación. Este sistema podría explicar los efectos a largo plazo de las interacciones entre los procesos evolutivos [42]

El problema de modelar sistemas complejos

El modelar sistemas complejos no produce resultados exactos. Cuando una simulación es desarrollada con un modelo del sistema, los valores de cada variable son registrados pero este solo representan una muestra o a veces es difícil interpretar u obtener los datos necesarios para la simulación. Por eso el promedio en una muestra de observación solo a veces provee un estimado de lo esperado, es decir, una simulación solo provee estimados, no resultados exactos. Meses de esfuerzo pueden ser requeridos para reunir información, construir, verificar

y validar modelos, diseñar experimentos y evaluar e interpretar los resultados. El costo de una simulación es alto, ya que depende de la obtención de todo tipo de información tanto cualitativa como cuantitativa. El establecimiento y mantenimiento de capacidad de simulación, envuelve tener mejor personal, software, hardware, entrenamiento y otro tipo de costos [43].

Hay muchas facetas para un balanceo y comprensivo estudio de la simulación. Ya que una persona debe tener conocimiento de una gran variedad de áreas antes de llegar a ser un practicante de la simulación. Este hecho es algunas veces ignorado, sin embargo como resultado, cada estudio puede incorrectamente ser desarrollado, o podría estar incompleto, o podría caer en otro tipo de caminos, quizá resultado de una falla del esfuerzo de la simulación [43].

El problema de modelar los sistemas no lineales

El desarrollo de las capacidades computacionales ha redefinido la manera en que los investigadores ven el mundo. Investigaciones científicas han pasado de los paradigmas basados en una concepción lineal del mundo a un paradigma más holístico basado en la no linealidad. La distinción entre sistemas lineales y no lineales de la ciencia no ha sido plenamente resuelta en la literatura [38]. El paradigma lineal se basa en la hipótesis de la independencia de que los agentes se utilizan para describir la realidad a través de un determinado modelo. En contraste, el crecimiento no lineal del mundo se basa en una percepción de la interdependencia.

La diferencia entre el lineal y no lineal, es que los paradigmas lineales el agregado es igual a la suma de sus partes, sin embargo en los no lineales la suma total es más que la suma de sus parte; teniendo un excedente. Según el paradigma no lineal, los sistemas no pueden reducirse a sus partes, y es por esta razón que este campo es comúnmente conocido como complejidad o como la ciencia de la sorpresa [38].

Estos sistemas están compuestos por poblaciones de agentes cuya adaptación ha sido resultado de interacciones complejas en dinámica no lineal, los cuales son fenómenos emergentes del sistema. Estos sistemas tiene que ver con comparar ejemplos naturales y artificiales de CAS con las propiedades y procesos, e investigar simulaciones de modelos simplificados de los sistemas naturales. Ofrece una conceptualización y el marco para una clase de sistemas complejos y sus consiguientes fenómenos utilizando ambas herramientas computacionales. El campo es intrínsecamente interdisciplinario, ligado a la ciencia de la complejidad, teoría de sistemas, teoría de control y débilmente relacionados con campos como la mecánica estadística, la inteligencia artificial, la teoría de juegos, y optimización [44].

2.2.2. Modelado basada en agentes

La aplicación de agentes en las ciencias naturales y ciencias sociales es un tema de investigación muy divulgado en el campo de la Inteligencia Artificial (AI; del inglés Artificial intelligence) y Sistemas Distribuidos. La AI fue una de las primeras áreas donde se empezó a trabajar en este tema [45].

Los agentes iniciaron de la investigación de la Inteligencia Artificial (AI [46] y más concretamente de la Inteligencia Artificial Distribuida (DAI; del inglés Distributed Artificial Intelligence) [33]. Al principio, la AI hablaba de los agentes como programas especiales cuya naturaleza y construcción no se llegaba a detallar.

Los sistemas multi-agente tienen un campo visual más amplio de todo el sistema, incluyendo la comunicación entre agentes, coordinación, colaboración, descomposición de trabajo, y resolución de conflictos por negociación [47]. Lo que es pieza clave para una simulación más cercana a la realidad.

Los sistemas multi-agente: consiste en agentes autónomos que trabajan juntos para resolver problemas, caracterizados porque cada agente tiene una información o capacidad in-

completa para solucionar el problema, no hay un sistema global de control, los datos están descentralizados y la computación es asíncrona. Los agentes deciden dinámicamente que tareas realizaran [45].

El agente está situado en algún entorno, dentro del cual actúa de forma autónoma y flexible para cumplir con su objetivo principal. Pero este agente recibirá entradas de su entorno y a la vez debe elegir una acción, según el autor [35] tiene ciertas propiedades como son las siguientes:

Autonomía Un agente debe tener una serie de objetivos y debe poseer conocimientos de su entorno, de tal forma que partiendo de sus conocimientos sea capaz de decidir qué es lo que más le conviene para su propio beneficio.

Sociabilidad Un agente puede colaborar con otros agentes y llevar a cabo una tarea con un objetivo en común. Sin embargo puede un agente decidir dependiendo de las características individuales y la percepción de su entorno agruparse o no hacerlo.

Reactividad Un agente debe ser capaz de percibir estímulos externos, tanto para estar de acuerdo a su entorno cambiante, como para saber que está pasando en el mundo. Estos estímulos afectarán las acciones realizadas por el agente para alcanzar sus objetivos.

Pro-actividad Debe ser capaz de elegir en cada momento las acciones que debe llevar a cabo para alcanzar sus objetivos. Es decir, una agente no sólo actúa en función de los estímulos que recibe desde el exterior, sino que puede realizar acciones como resultado de sus propias decisiones.

Inteligencia Un agente es inteligente si es racional, coherente y adaptable, en mayor o menor medida. Un agente será más inteligente cuánto más desarrolladas tenga estas características

Racionalidad Un agente se considera racional cuando tiene unos conocimientos de su entorno, unos objetivos deseables y unas reglas que determinan cómo alcanzar los objetivos a partir del conocimiento que se tiene del medio.

Coherencia El conocimiento que un agente tiene de su mundo se almacena en una base de conocimientos propia del agente. Todo este conocimiento debe guardar un alto grado de coherencia para que el comportamiento del mismo sea el esperado.

Adaptación El aprendizaje es una de las características más complejas que puede tener un agente. Un agente aprende cuando es capaz de aumentar su base de conocimientos y su base de reglas a partir de las percepciones que recibe del entorno y a partir de sus comportamientos anteriores a la hora de resolver problemas.

Movilidad Un agente móvil es aquel que puede moverse físicamente por los nodos de una red para llevar a cabo sus tareas.

2.2.3. Modelado de sistemas sociales complejos con agencias distribuidas usando sistemas de inferencia difusa.

Con el fin de construir un modelo inspirado en la agencia distribuida, se propone utilizar sistemas difusos y seguir los siguientes pasos[48]:

1. Determinar los niveles de la agencia y sus relaciones implícitas.

2. Minería de datos de un conjunto de datos y la generación de un conjunto de reglas (aplicando un método de minería de datos).

3. Modelado Multi-Agente (implementación de una herramienta de simulación basada en agentes).

4. Validar el modelo, simular y optimizar experimentos.

5. Analizar las salidas.

Aunque este procedimiento cubre todo el ciclo de vida de las agencias distribuidas y el uso de sistemas difusos en el proceso de investigación, en este trabajo estamos describiendo una herramienta para crear estructuras multi-agente. Estamos centrados en el enfoque de modelado multi-agente con el fin de diseñar una estructura de agente y añadir una base de reglas en estos agentes.

2.2.4. Inteligencia computacional

Lógica difusa

Los conjuntos difusos fueron introducidos por Lofti A. Zadeh en 1965 [49] para procesar o manipular información con incertidumbre no probabilística. Fueron diseñados para representar matemáticamente la incertidumbre lingüística; proporcionando las herramientas formalizadas para trabajar con la imprecisión intrínseca en muchos problemas; puede ser considerada como una generalización de la teoría de conjuntos clásica.

La gran diferencia de los conjuntos difusos a los conjuntos clásicos es que en un conjunto clásico, un elemento del universo pertenece o no al conjunto. Esto es, la membresía de un elemento es dura — ya sea sí está en el conjunto o no está en el conjunto. Un conjunto difuso es una generalización de un intervalo unidad [0,1]. Así, la función de membresía de un

conjunto difuso mapea cada elemento del universo de discurso a un rango del espacio el cual, en muchos casos, es el conjunto al intervalo de la unidad.

Otra diferencia es que los conjuntos duros siempre tienen funciones de membresía únicas, mientras que todo conjunto difuso tiene un número infinito de funciones de membresía que pueden representarlo. Esto permite que los sistemas difusos puedan ser ajustados a su rendimiento máximo en una situación dada. En un sentido amplio, como apuntó Lotfi Zadeh [50] cualquier campo puede ser fusificado y generalizado reemplazando el concepto de conjunto duro en un campo fuente por el concepto de un conjunto difuso. Por ejemplo, se puede fusificar[1], algunos campos básicos tales como la aritmética, la teoría de grafos y la teoría de la probabilidad para desarrollar aritmética difusa, teoría de grafos difusos y teoría de probabilidad difusa, respectivamente; también se puede fusificar algunos campos aplicados tales como redes neuronales, algoritmos genéticos, teoría de estabilidad, reconocimiento de patrones y programación matemática para obtener redes neuronales difusas, algoritmos genéticos difusos, teoría de estabilidad difusa, reconocimiento de patrones difusos y programación matemática difusa, respectivamente. Los beneficios de tal fusificación incluyen mayor generalidad, poder expresivo más alto, una habilidad elevada para modelar problemas del mundo real, y una metodología para explotar la tolerancia a la imprecisión.

La creación de un conjunto difuso depende de dos aspectos: la identificación de un universo de valores apropiados y especificar una función de membresía correctamente. La elección de la función de pertenencia es un proceso subjetivo, lo que significa que diferentes personas pueden llegar a diferentes conclusiones sobre el mismo concepto. Esta subjetividad se deriva de las diferencias individuales en la percepción y la expresión de los conceptos abstractos y tiene muy poco que ver con el azar. Por lo tanto, la subjetividad y la arbitrariedad de un conjunto difuso es la principal diferencia entre el estudio de los conjuntos difusos y la teoría

[1] fusificar,consiste en convertir una variable real en un grado de pertenencia que cuantifica el grado de posesión hacia su correspondiente variable lingüística.

probabilista [51].

En los conjuntos difusos tipo 1, una vez que la función de pertenencia se define para un concepto, este se basa en la opinión subjetiva de uno o más individuos y no muestra más de un valor para cada elemento del universo. Al hacerlo, se pierde parte de la ambigüedad que se discuten algunos conceptos, especialmente donde las personas pueden tener una opinión ligeramente diferente y todos se consideran válidos. Los conjuntos difusos de tipo 2 permiten el manejo de incertidumbres lingüísticas y numéricas. La figura 2.3 representa a dos gráficos de conjuntos difusos, a) Lógica difusa de tipo 1, y b) Lógica difusa de tipo 2.

En a) el conjunto de valores que se muestra es $A = \{(x, \mu_A(x))|x \in X\}$ donde $A \subseteq X$, X $A \subseteq X$, X es el universo de valores válidos y $\mu_A(x)$ es la función de pertenencia(MF), esta contiene un mapeto de cada valor de X con su valor de pertenencia correspondiente a un valor entre 0 y 1. Para b) el conjunto de valores es $\tilde{A} = \{((x, u), \mu_{\tilde{A}}(x, u))\}$, donde MF $\mu_{\tilde{A}}(x, u)$ tiene un valor de pertenencia para cada elemento del universo en función del número de miembros en el rango [0, 1], por lo que la huella se puede ver alrededor de la curva).

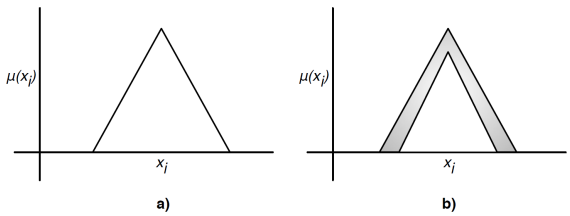

Figura 2.3: Conjunto difuso tipo 1 y conjunto difuso con incertidumbre tipo 2 (Fuente: Castañón-Puga et. al., 2013).

Sistemas de inferencia difusa Un sistema de inferencia difuso(FIS; del inglés Fuzzy Inference System) se basa en reglas lógicas que pueden trabajar con valores numéricos o entradas difusas, las reglas son evaluadas y los resultados individuales se unen (agregación) para formar lo que es la salida difusa, después un valor numérico se debe pasar por un proceso de defusificación si es necesario. La Figura 2.4 muestra un diagrama de bloques de

la estructura clásica de un FIS.

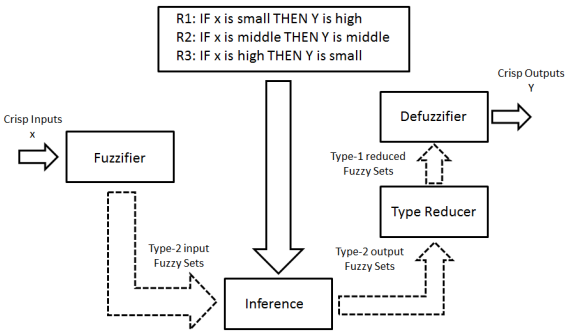

Figura 2.4: Diagrama de Bloques de un Sistema de Inferencia Difusa Tipo 2 (Fuente: Castañón-Puga et. al., 2013).

Sistemas de inferencia difusa orientado a objetos Existen bibliotecas de código y conjuntos de herramientas disponibles para construir sistemas de inferencia difusos [52, 53]. Algunas de estas bibliotecas de clases se desarrollan en un paradigma de programación orientado a objetos, principalmente en la construcción de sistemas de inferencia difusa tipo 1 [54]. JFuzzyLogic [55] y Juzzy [56] son ejemplos de bibliotecas de clases escritas en Java. La ventaja de un sistemas de inferencia difuso en Java es que podemos construir sistemas inteligentes con capacidades de lógica difusa tipo 2 utilizando un lenguaje de programación orientado a objetos. Java es un lenguaje de programación orientado a objeto robusto utilizado en una amplia gama de aplicaciones.

Minería de datos

En los últimos años, el uso de las nuevas tecnologías de la información han ayudado en la manipulación de grandes cantidades de datos, una evolución de estas tecnologías es la minería de datos que permite representar el conocimiento de almacenes de datos de forma implícita en las grandes bases de datos. La minería de datos es un campo multidisciplinario que combina técnicas de aprendizaje automático, reconocimiento de patrones, estadísticas,

bases de datos y visualización, para dirigirla a la extracción e interpretación de una enorme cantidad de datos en una base de datos. La minería de datos se centra en la necesidad de descubrir, predecir, y pronosticar las posibles acciones con algún factor de confianza para cada predicción [57].

Por otra parte, contribuye a la toma de decisiones tácticas y estratégicas proporcionados por usuarios, es capaz de medir las acciones y resultados de la mejor manera, genera modelos descriptivos para explorar y comprender los datos e identificar patrones, relaciones y dependencias que afectan a los resultados finales. Crear modelos predictivos que permitan descubrir la relación a través del proceso de minería de datos se expresan como posibles reglas de negocio[58].

Algoritmo de agrupamiento fuzzy c-means para minería de datos

Las propuestas de representaciones difusas tienen un lugar importante en la minería de datos que proporciona resultados inteligibles. Una de estas propuestas es el algoritmo de agrupamiento de las c-medias borroso (FCM; del inglés Fuzzy C-Means)[59] [60] esta hace uso de la función de pertenencia y procedimientos de cálculo centroide iterativamente para encontrar el mejor centroide(centro). FCM es uno de los algoritmos de agrupamiento más popular, la eficacia del método de agrupación se basa en la medida de la distancia entre los datos. FCM es la combinación resultante del método de minería de datos c-means con el manejo de los datos difusos. El resultado de esta combinación es adecuada, ya que considera la incertidumbre presente en los datos para evitar resultados incorrectos y crear particiones nítidas de forma correcta [59].

Adicionalmente FCM es usado para adquirir los niveles adecuados de los parámetros de los conjuntos de agrupación [61].

Sistemas evolutivos

Los sistemas complejos contribuyen a la teoría de la evolución. La primera es en la contribución de nuevos métodos de modelos matemáticos y computacionales que ayuden a representar comportamientos emergentes. La segunda es en la identificación y elaboración de ideas de otros sistemas complejos que son relevantes para la ecología y la evolución.

Mitchelle y Newman hacen mención que los agentes posiblemente puedan modelar la teoría de evolución, y mostrar un comportamiento interesante. Esto con ayuda de los algoritmos genéticos. Para realizar la auto-reproducción para la aparición de la multicelularidad a través de interacciones competitivas y cooperativas. En cada una de estas simulaciones, un comportamiento emergente del sistema se identifica y cuantifican. Los procesos de macro evolutivos es una clase adicional del agente. En el cual se dice que la macro evolución se encuentra en un nivel superior de las especies, géneros, familias y así sucesivamente. Probablemente el ejemplo más conocido de un modelo de macro evolutivo es el modelo de coevolución de Bak y Sneppen, que intenta explicar la extinción en masa como consecuencia de las interacciones de las especies [25].

Para Heylighen la evolución tiende a ser visto en términos de adaptación a un entorno externo y la auto-organización como resultado de la una dinámica interna. Pero en sistemas no hay diferencia entre interno y externo, da el ejemplo de las hormigas el cual también inspiran la auto-organización natural: cuando las hormigas encuentran comida, dejan un rastro de feromonas a lo largo de su trayectoria. Otras hormigas en busca de comida son más propensas a ir en una dirección donde hay más feromonas. Si tiene éxito, ellos también añadir las feromonas, tomando el camino más fuerte, y con más probabilidades de atraer a las demás hormigas. Si no se encuentra la alimentación, no se agregan las feromonas y el camino poco a poco se evapora. De esta manera, una colonia de hormigas, al principio, explorar su entorno al azar, pero luego poco a poco desarrolla una “ruta” que conecta el nido y las fuentes de

alimentos de forma más eficiente [18]. Otro punto importante que menciona Heylighen es que la única manera de que podamos hacer frente a problemas complejos es serle frente con modelos complejos basados en auto-organización, llegando a ser cada vez mejor adaptables a objetivos que van cambiando. Sea o no aumenta la complejidad en la evolución es una de las cuestiones centrales de la biología evolutiva. Las opiniones sobre este tema varían, pero generalmente pertenecen a uno de los tres campos[17]:

1. Sugiere que la complejidad ha aumentado.
2. Afirma que no hay pruebas suficientes para argumentar a favor o en contra de un aumento.
3. Niega que "el progreso que caracteriza la historia de la vida como un todo, ni siquiera representa una fuerza orientadora en la evolución de todos".

La mayoría están de acuerdo que nadie sabe con precisión qué se entiende por la palabra complejidad cuando se refiere a un organismo biológico. En efecto, mientras que las medidas de complejidad abundan, su relación con la biología no es siempre clara. La complejidad es tan general, un término que parece significar algo diferente a todos. Los dos principales grupos de usuarios, son físicos interesados en sistemas dinámicos, y los biólogos de reflexionar sobre la cuestión antes mencionada de una tendencia [17]. En la teoría de sistemas dinámicos, Adami se interesa en la complejidad de los procesos. Por ejemplo, los procesos de periódicos y aleatorios son percibidos como simples, con los procesos aleatorios en "el otro extremo de la escala", cualquiera que sea la escala que sea. Los procesos complejos y caóticos se considera que se encuentran en algún punto intermedio. Debido a que todos los procesos pueden considerarse como medidas de complejidad en la teoría de sistemas dinámicos suelen ser de cálculo. Estas construcciones permiten deducir la complejidad de una secuencia de símbolos de encontrar una apropiada máquina de estados finitos que produce esta secuencia

[17].

La complejidad de los organismos biológicos aún no puede ser capturada al intentar caracterizar la dinámica de todos sus procesos subyacentes. En cambio, las medidas de la complejidad biológica al referirse a la forma, función, o la secuencia que codifica. La complejidad estructural es generalmente lo que se quiere decir cuando consideramos los animales, pero esto parece ser la medida más difícil de definir. También toma en cuenta que la complejidad no se debe equiparar con éxito evolutivo una concepción errónea de que ha dado lugar a muchas controversias [17].

La complejidad de una secuencia física se refiere a la cantidad de información que se almacena en la secuencia de un entorno particular. Para un genoma, el medio ambiente es aquel en el que se replica y en el que vive su anfitrión, un concepto más o menos equivalente a lo que se llama un nicho. La definición de complejidad física debe distinguirse de la matemática (o algorítmica, o de Kolmogorov) la complejidad, que sólo se refiere a la regularidad intrínseca (o, en este caso, la irregularidad) de una secuencia. La regularidad de una secuencia es un reflejo de las leyes inmutables de las matemáticas, y no del mundo físico en el que dicha secuencia puede significar algo[17].

La aleatoriedad es en cierto modo se llama entropía en teoría de la información. La entropía es una medida del potencial de conocimiento, o si se aplica a una secuencia, una medida de la cantidad de información de una secuencia puede celebrar, y por lo tanto cuantifica la incertidumbre acerca de la identidad genética de una persona seleccionada al azar. Es útil pensar en la secuencia de la entropía como la longitud de la cinta, mientras que la información es la longitud de la cinta que contiene las grabaciones. De medición (es decir, la grabación) se convierte en la cinta de la cinta vacía llena, la entropía en la información. Como veremos, esto es lo que sucede durante la adaptación, y es la fuerza que impulsa el aumento de la complejidad. La información es una forma estadística de la correlación, y por tanto requie-

re, matemáticamente y de manera intuitiva, una referencia al sistema que la información se trata. La secuencia de su información llena de cintas le permite hacer predicciones sobre el estado del sistema que la secuencia es la información sobre. Esta capacidad de predicción implica que la secuencia y el sistema tienen algo en común, que están correlacionados [17] .

Morin menciona que al mismo tiempo que el universo altera el orden implacable de la vida, Darwin presenta la variación y la competencia como motores de la evolución. Post-darwinismo, si tiene, en algunos casos, atenuar el carácter radical del conflicto, la oportunidad, o casualidad; dentro de la concepción "creación" o "invención" las nuevas formas de vida de la organización, como las alas, los ojos de uno. Al comienzo del siglo XX, la microfísica introduce la incertidumbre fundamental en el universo de las partículas que deja de obedecer a las concepciones del espacio y el tiempo característico de nuestro universo llamado macro-física [20]. Siendo un aspecto fundamental en el proceso de la evolución.

Los trabajos de Holland [31]. en sistemas complejos han sido muy populares en el campo de la computación evolutiva, ya que ayudó a que se realizara lo que es el algoritmo genético. El aprendizaje de los sistemas de clasificación es básico en la Inteligencia Computacional (CI; del inglés Computational Intelligence) para las técnicas adecuadas de difícil dominio tales como el diseño, ingeniería y reconocimiento de patrones. Trabajo de Forrest, ayudaron al estudio del sistema inmunológico, los modelos de adaptación ayudaron a desencadenar en campos relacionados con la CI (específicamente Inspirada en la Biología de Computación (CBI; del inglés Computational Biology)). Sin duda otro ejemplo sería el campo de redes neuronales artificiales aplicadas a la clasificación y función de aproximación, teniendo también en derivados de modelos de CAS. Esta observación también fue hecha por Mitchell en 1993 como una manera viable de contribuir a la CI y CBI. Los sistemas complejos adaptativos es una interesante y una potente conceptualización, que ha contribuido modelos por agentes; en la vida artificial; biología computacional y otras variaciones de cómputo de las ciencias físicas [27].

2.2.5. Simulación social

La técnica de simulación en las ciencias sociales es apreciada como una tercera forma de hacer ciencia, en contraste con la inducción y deducción.

El manejo de la simulación como herramienta es una técnica novedosa y de rápido crecimiento en el campo de las ciencias sociales. Como la mayoría de las técnicas nuevas en los campos, la promesa es mayor que la demostración de sus logros[11].

Algunos autores como Davidsson denomina la simulación social entre las ciencias sociales y la simulación por computadora normalmente denominándola (SocSim; del inglés Social Simulation) y corresponde a la simulación de los fenómenos sociales en una computadora con cualquier técnica de simulación y suele utilizar modelos sencillos de simulación de las entidades sociales, por ejemplo: autómatas celulares, objetos, que son capaces de realizar una interacción muy elemental [45].

El resultado de simulaciones han demostrado proporcionar una poderosa alternativa para complementar, sustituir y ampliar los enfoques tradicionales de enseñanza en un ámbito como el cambio y la gestión de la innovación, ofreciendo una oportunidad de aplicar los modelos de la dinámica de organización social [45].

La simulación social puede contribuir a la comprensión de los procesos sociales; o algún tipo de teoría o modelo. En general, esas teorías se exponen en forma textual, aunque a veces la teoría se representa como una ecuación; una tercera forma es expresar las teorías como los programas computacionales. Los procesos sociales pueden ser simulados en la computadora. En algunas circunstancias, incluso es posible llevar a cabo experimentos sobre los sistemas sociales artificialmente que sería totalmente imposible o poco ético para llevarlo a cabo en las poblaciones humanas [45].

Cada relación con el modelo debe ser especificado exactamente y cada parámetro; de lo

contrario será imposible ejecutar la simulación. Esta disciplina implica también que el modelo sea potencialmente abierto a la inspección por otros investigadores, en todos sus detalles. Estos beneficios de la claridad y la precisión también tienen desventajas, sin embargo. Simulaciones de procesos sociales complejos implican la estimación de muchos parámetros y datos adecuados para hacer las estimaciones que pueden ser difíciles de encontrar. Otro beneficio de la simulación es que, en algunas circunstancias, puede dar ideas sobre la "aparición" de los fenómenos a nivel macro de las acciones a nivel micro. Por ejemplo, una simulación de la interacción de los individuos puede revelar patrones claros de influencia cuando se examina en una escala de la sociedad [45].

La simulación social muestra cómo esta nueva metodología es adecuada para el análisis de los fenómenos sociales que son inherentemente complejos. Si bien la idea de la simulación ha tenido enorme influencia en la mayoría de las áreas de la ciencia, e incluso en la programación de juegos, donde ya hay una emulación de las sociedades teniendo un impacto significativo en las ciencias sociales. El avance se produjo cuando se dieron cuenta de que los programas computacionales ofrecen la posibilidad de crear artificialmente sociedades en las que las personas y actores colectivos, pueden ser organizaciones representadas directamente y observar el efecto de sus interacciones. Esto proporcionó la posibilidad de utilizar métodos experimentales con los fenómenos sociales, y la utilización de código de computadora como una manera de formalizar las teorías dinámicas sociales [38].

Las simulaciones del mundo real incluyendo la población como un objetivo deben incluir algunos medios de validación. En econometría, en las ciencias políticas y sociología los conjuntos de datos para la verificación son abundantes. Otros ámbitos, principalmente la antropología sufren la falta de datos. El suministro de estos datos, son una preocupación secundaria. La dificultad principal que en los conjuntos de datos se adecuen a la arquitectura del agente; un ejemplo de ello son los estudios que se centraron principalmente en las raíces cognitivas de la teoría social [62].

2.3. Aplicaciones

2.3.1. Sistemas toma de decisiones

Para Robert Lempert [63], los modelos basados en agentes son una herramienta cada vez más potente para la simulación de los sistemas sociales, ya que pueden representar importante fenómeno difícil de captar en otros formalismos matemáticos. Pero, los modelos basados en agentes han proporcionado sólo un apoyo limitado para la formulación de políticas debido a que sus habilidades distintivas son a menudo más útiles en situaciones donde el futuro es impredecible. En tales situaciones, los métodos de análisis tradicionales con aplicación a modelos de simulación son menos eficaces en la toma de decisiones. Los modelos basados en agentes son herramientas de gran alcance para la simulación de los sistemas sociales, para empleo de organizaciones y Simuladores de Políticas Públicas; el uso de modelos como los simuladores de políticas proporcionan un apoyo cuantitativo a los toma de decisiones en los sectores públicos y privados. Esta cuestión tiene especial relevancia porque estos modelos tienen hasta la fecha un impacto limitado en la práctica de toma de decisiones.

¿Por qué los modelos de uso basados en agentes para apoyar las decisiones del mundo real? No hay escasez de representaciones matemáticas más tradicionales utilizadas para tales fines, incluyendo las ecuaciones diferenciales de la ingeniería y los modelos económicos, los analistas estadísticos, o modelos de dinámica de sistemas. Los modelos basados en agentes son útiles porque pueden representar información importante sobre el mundo los cuales con los modelos tradicionales no perciben fácilmente. En particular, los modelos basados en agentes tienen éxito en el que relaciona el comportamiento heterogéneo de agentes con información diferente, diferentes reglas de decisión, y situaciones diferentes para el comportamiento macro del sistema global. Se ha utilizado modelos basados en agentes para combinar la información antropológica sobre el comportamiento de los individuos y grupos en sociedades con datos

detallados sobre los cambios climatológicos del medio ambiente.

La capacidad de los modelos basados en agentes para conectar micro comportamientos heterogéneos con diferentes patrones de macro comportamiento plantea una segunda cuestión crucial. Una vez que los modelos se crean, cómo se debe ejercer para apoyar la toma de decisiones? Las fortalezas distintivas de los modelos basados en agentes, a menudo surgen en situaciones en que su aplicación constituye un desafío particular. Análisis de decisiones tradicional ha sido enormemente útil para que la aplicación sistemática de modelos de simulación apoye la toma de decisiones. Estos métodos emplean modelos de simulación para clasificar las acciones alternativas mediante la predicción de sus consecuencias futuras. Lo que significa, que los modelos de simulación son tradicionalmente útiles debido a su capacidad de predecir con certeza el futuro, ya sea en un sentido de estimación probabilística. Pero los modelos basados en agentes son a menudo más útiles en condiciones de profunda incertidumbre donde estas predicciones no son posibles.

Al abordar un problema complejo de políticas públicas, como las acciones que deben tomar a la amenaza del cambio climático global, tiene un gran número de escenarios potencialmente relevantes que deben ser considerados. Como mínimo, el análisis de políticas rigurosas requiere algún medio para definir e identificar los escenarios más importantes. Los avances que se ha hecho en simuladores de modelos basados en agentes viables también han permitido a los nuevos métodos de análisis de decisiones adecuarlas este tipo de problemas La incertidumbre surge cuando las partes de una decisión no saben o no se pueden ponerse de acuerdo sobre uno o varios componentes clave de un análisis de decisión: el modelo del sistema, las probabilidades a priori de cualquier parámetro que describe el modelo del sistema, y/o el valor de la función utilizada para clasificar los resultados del modelo; para el uso con modelos no predictivos. Estos métodos de simulación multi-escenario proporcionan una orientación sistemática y cuantitativa en cuanto a qué escenarios debe examinar la información y extraer.

Los métodos de simulación multi escenario toman una variedad de formas y están emergiendo con modelos-basado en agentes no predictivos. Todos hacen hincapié en uno o más pasos clave. En primer lugar, que utilizan conjuntos de escenarios, con cientos de millones de miembros, para capturar lo que se conoce sobre el futuro, en vez de una sola cifra o pequeño de las instalaciones. En segundo lugar, se comparan las decisiones de políticas alternativas a través de este conjunto de escenarios en función de criterios como la robustez, resistencia y estabilidad. Una estrategia sólida es aquella que funciona relativamente bien, en comparación con las alternativas, a través de una amplia gama de futuros posibles. El análisis de la política tradicional se basa en el criterio de optimización o eficiencia. Las políticas alternativas son clasificadas por su desempeño esperado dada la ponderación de probabilidades sobre los escenarios alternativos. Pero, la robustez es un criterio mejor en aquellas situaciones en las que es difícil predecir el futuro, es decir, la probabilidad de escenarios alternativos es desconocida. No es de extrañar, robustez, no la eficacia ni la optimización, es a menudo el criterio humano los tomadores de decisiones en uso en la práctica este tipo de situaciones. Finalmente, los métodos de simulación multi-escenario en cuenta explícitamente las decisiones de política como una respuesta adaptativa que evolucionará con el tiempo a la nueva información, más que un conjunto fijo de acciones. Los responsables políticos suelen lograr robustez con la planificación adaptativa tal. En conjunto, estos pasos se utiliza la computación como una herramienta interactiva para ayudar a los humanos con lo que mejor saben hacer, descubrir patrones, hacer hipótesis acerca de las mejores acciones a tomar, y poner a prueba estas hipótesis con los datos disponibles capturado en el escenario de conjuntos-en lugar de utilizar computadoras como las calculadoras para apoyar el razonamiento deductivo. Por ejemplo, la simulación multi-escenario menudo define, en el inicio del análisis, los escenarios más importantes como las de mayor utilidad en la obtención de información y apoyo por parte de las partes involucradas en la decisión. Al final del análisis, los escenarios más importantes se pueden destacar las más de las acciones políticas recomendadas[63] .

2.3.2. Sistemas sociales

Cuando se habla de sistemas sociales se debe tener ciertas características básicas en las organizaciones; una de ellas es que las consecuencias de los sistemas sociales son probabilísticas y no-determinísticos. Además el comportamiento humano nunca será totalmente previsible, ya que las personas son complejas. Por esto, la administración no puede esperar a que consumidores, proveedores, agencias reguladoras y otros, tengan un comportamiento previsible [38].

Las organizaciones son vistas como sistemas dentro de sistemas. Dichos sistemas son complejos, produciendo un todo que no puede ser comprendido tomando las partes independientemente. La organización se debe enfocar como un sistema que se caracteriza por todas las propiedades esenciales a cualquier sistema social [64].

Interdependencia de las partes Un cambio en una de las partes del sistema, afectará a las demás. Las interacciones internas y externas del sistema reflejan diferentes escalones de control y de autonomía.

Estado firme La organización puede alcanzar el estado firme, solo cuando se presenta dos requisitos, la unídireccionalidad y el progreso. La unidireccionalidad significa que a pesar de que haya cambios en la empresa, los mismos resultados o condiciones establecidos son alcanzados. El progreso referido al fin deseado, es un grado de progreso que está dentro de los límites definidos como tolerables.

Fronteras o límites Es la línea que demarca lo que está dentro y fuera del sistema. Podría no ser física. Una frontera consiste en una línea cerrada alrededor de variables seleccionadas entre aquellas que tengan mayor intercambio (de energía, información) con el sistema.

Morfogénesis El sistema organizacional, diferente de los otros sistemas mecánicos y aun de los sistemas biológicos, tiene la capacidad de modificar sus maneras estructurales básicas, es identificada por Buckley como su principal característica identificadora[65].

2.4. Herramientas

2.4.1. Herramienta MATLAB®

MATLAB® es uno de los lenguajes mas populares usado para la programación científica y numérica, con un gran cantidad de usuarios y esta va en aumento. Sin duda, es uno de principales idiomas utilizados en la investigación, educación y desarrollo con fines científicos y en aplicaciones de ingeniería [66].

MATLAB® es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico , visualización y programación. Usando MATLAB , puede analizar los datos , desarrollar algoritmos, crear modelos y aplicaciones. El lenguaje, las herramientas y funciones matemáticas integradas que permiten explorar múltiples enfoques y llegar a una solución más rápida que con hojas de cálculo o lenguajes de programación tradicionales, como C/C++ o Java®. Puede utilizar MATLAB para una gama de aplicaciones, incluyendo el procesamiento de señales y comunicaciones, procesamiento de imágenes y vídeo, sistemas de control, prueba y medida, finanzas computacionales y la biología computacional[67].

Sus características principales son[67]:

- Lenguaje de alto nivel para el cálculo numérico, visualización y desarrollo de aplicaciones.
- Entorno interactivo para la exploración iterativa, el diseño y la resolución de problemas.

- Funciones matemáticas para álgebra lineal, estadísticas, análisis de Fourier, filtrado, optimización, integración numérica, y la resolución de ecuaciones diferenciales ordinarias.
- Gráficos integradas para la visualización de datos y herramientas para la creación de diagramas personalizados.
- Herramientas de desarrollo para mejorar la calidad del código y facilidad de mantenimiento y maximizar el rendimiento.
- Herramientas para la creación de aplicaciones con interfaces gráficas personalizadas.
- Funciones para integrar los algoritmos basados en MATLAB® con aplicaciones externas y lenguajes como C, Java®, .NET y Microsoft® Excel®.

2.4.2. Herramienta NetLogo

NetLogo es un entorno de modelado programable para simular fenómenos naturales y sociales. Fue escrito por Uri Wilensky en 1999 y ha estado en constante desarrollo desde entonces [68, 69].

Fue claramente diseñado con un tipo específico de modelo en mente: los agentes móviles que actúan simultáneamente en un espacio de red con la conducta dominada por las interacciones locales durante períodos cortos de tiempo. Mientras que los modelos de este tipo son los más fáciles de implementar en NetLogo, la plataforma es de ningún modo esta limitada a ellos. NetLogo es una herramienta bien aceptada por los investigadores por su apariencia y documentación [69].

Es particularmente adecuada para el modelado de desarrollo de sistemas complejos en el tiempo. Los modeladores pueden dar instrucciones a cientos o miles de "agentes"todos los

que operan de forma independiente. Esto hace que sea posible explorar la conexión entre el comportamiento a nivel micro de los individuos y los patrones de nivel macro que surgen de su interacción[68].

Permite a los estudiantes crear simulaciones abiertas y "jugarçon ellas, la exploración de su comportamiento bajo diversas condiciones. También es un entorno de edición que permite a los estudiantes, los profesores y los diseñadores curriculares crear sus propios modelos. NetLogo es bastante simple para los estudiantes y profesores, pero lo suficientemente avanzada para servir como una poderosa herramienta para los investigadores en muchos campos[68].

NetLogo se recomienda para los modelos que son compatibles con un paradigma de corto plazo, la interacción local de agentes y un entorno de red y que no sea extremadamente compleja. También se recomienda NetLogo para los modelos de creación de prototipos que posteriormente pueden ser implementados en plataformas de bajo nivel: empezar a construir un modelo de NetLogo puede ser una forma rápida y exhaustiva para explorar las decisiones de diseño. Su velocidad de ejecución puede no ser una limitación importante para muchas aplicaciones, especialmente en comparación con el potencial de reducción en el tiempo de programación [69]. Aunque NetLogo utiliza un solo archivo contiene el código del modelo completo, las bibliotecas podrían ser creados para tener una mejor organización del modelo, sobre todo cuando se trata de modelos muy grandes.

2.4.3. Herramienta MASON

MASON se originó como una pequeña biblioteca para una gama muy amplia de simulación multi-agente que van desde la robótica a los agentes del juego a los modelos sociales y físicos. MASON surgió de la necesidad de aplicar un sistema en el que se aplique la variedad de problemas multi-agente [70].

MASON está escrito en Java con el fin de aprovechar la característica de portabilidad

ofrecida por este lenguaje, estrictas matemáticas y definiciones de tipos (garantizar la replica de resultados), y la serialización de objetos (para comprobar puntos de simulaciones) [71, 72].

Los objetivos en los que fue diseñado MASON son los siguientes[71, 72] :

- Un núcleo pequeño, rápido, fácil de entender y fácil de modificar.
- Separar la visualización, extensible en 2D y 3D.
- Producción de resultados idénticos independientes de la plataforma.
- Puntos de acceso de cualquier modelo en el disco de forma que se puede reanudar en cualquier plataforma con o sin visualización.
- Apoyo eficaz para un máximo de un millón de agentes sin visualización.
- Un apoyo eficiente a tantos agentes como sea posible en virtud de la visualización (limitado por memoria).
- Fácil incorporación en las bibliotecas existentes más grandes, incluyendo el tener múltiples instancias del sistema de co-existentes en la memoria.

MASON es altamente modular, con una explícita arquitectura en capas: las capas interiores no tienen vínculos con las capas externas de ningún tipo, y las capas externas pueden ser completamente eliminadas. MASON tiene la visión de al menos 5 capas: un conjunto de servicios básicos, la biblioteca básica del modelo, kits de herramientas de visualización, capas de simulación personalizadas, y aplicaciones de simulación que utiliza la biblioteca. Estas capas se puestran en la Figura 2.5 [70].

En cualquier momento, podremos separar el modelo a partir de la visualización, el modelo en el disco, mover lo a una plataforma diferente y dejar que continúe la ejecución, o adjuntar una colección de visualizaciones totalmente diferente. La Figura 2.6 muestra este procedimiento[71, 72].

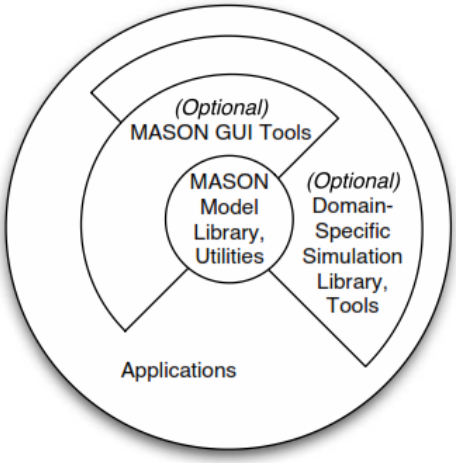

Figura 2.5: Arquitectura de MASON (Fuente: Sean Luke et. al., 2003).

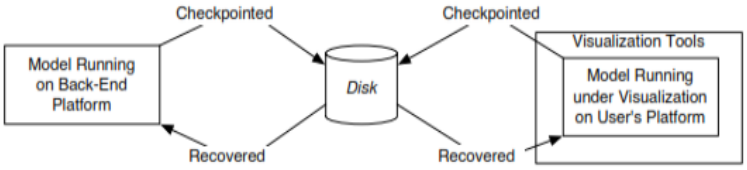

Figura 2.6: Los puntos de control y la recuperación de un modelo MASON para ejecutarse independiente o bajo diferentes tipos de visualización (Fuente: Sean Luke et. al., 2005).

2.4.4. Plataforma Jade

JADE es el middleware desarrollado por TILAB para el desarrollo de aplicaciones multi-agente distribuidas, basadas en la arquitectura de la comunicación de igual a igual. Tanto la inteligencia, la iniciativa, la información, los recursos y el control se pueden distribuir por completo en los terminales móviles, así como en los equipos de la red fija. El medio ambiente puede evolucionar de forma dinámica con sus compañeros, que en JADE son llamados agentes, que aparecen y desaparecen en el sistema de acuerdo a las necesidades y los requisitos del entorno de aplicación. La comunicación entre los compañeros, con independencia de que se están ejecutando en la red inalámbrica o alámbrica, es totalmente simétrica entre pares de poder jugar el iniciador y el papel que responde[73].

JADE está totalmente desarrollado en Java y está basado en los siguientes principios de conducción[73]:

- **Interoperabilidad.** JADE cumple con las especificaciones de FIPA. Como consecuencia de ello, los agentes JADE pueden ínter operar con otros agentes, siempre que cumplan con los mismos estándares.
- **Uniformidad y Portabilidad.** JADE proporciona un conjunto homogéneo de APIs que son independientes de la red subyacente y la versión Java. Más en detalle, JADE en tiempo de ejecución proporciona las mismas APIs tanto para el entorno J2EE, J2SE y J2ME. En teoría, los desarrolladores de aplicaciones pueden decidir el entorno de ejecución de Java en tiempo de despliegue.
- **Fácil de usar.** La complejidad del middleware se esconde detrás de un sistema sencillo e intuitivo de APIs.
- **Filosofía Pay-as-you-go.** -Los programadores no tienen que utilizar todas las características proporcionadas por el middleware. Las características que no se utilizan no requieren que los programadores sepan nada de ellas, ni ningún esfuerzo añadido computacional.

Capítulo 3

JT2FIS Framework

En este capítulo se describe en detalle el marco de trabajo para el desarrollo de software denominado `JT2FIS Framework`. El marco de trabajo fue desarrollado con el objetivo de proveer el código base para el desarrollo posterior de herramientas de simulación. Se consideró desde el inicio el enfoque de inteligencia computacional, por lo que se construyó bibliotecas y componentes que ayudan en la implemenatción de sistemas de software inteligentes.

El marco de trabajo está formado por una biblioteca de clases para construir sistemas inteligentes basados en lógica difusa tipo-2, componentes visuales que pueden ser utilizados fácilmente dentro de aplicaciones visuales, y métodos de agrupamiento enfocados en la generación automática de sistemas de inferencia difusos.

Aunque las herramientas de simulación social, sobre todo en el concepto de agentes cognitivos, no necesariamente están restringidas a incorporar sistemas de inferencia difusos, este marco de trabajo fue construido como un punto de partida para incorporar un enfoque novedoso en este campo del conocimiento.

3.1. La biblioteca de clases JT2FIS

JT2FIS[74] es una biblioteca de clases desarrollada en Java. El propósito principal de esta biblioteca es construir Sistemas de Inferencia Tipo 2 por intervalos aplicando la programación orientada a objetos.

3.1.1. Diseño JT2FIS

JT2FIS esta integrado por una estructura en paquetes que contienen una colección de clases. El contenido y la organización de estos paquetes se muestran en la Tabla 3.1.

La Figura 3.1 muestra un diagrama de paquetes UML de la biblioteca de clases JT2FIS. La colección de clases esta organizada en paquetes de código, que encapsulan su comportamiento.

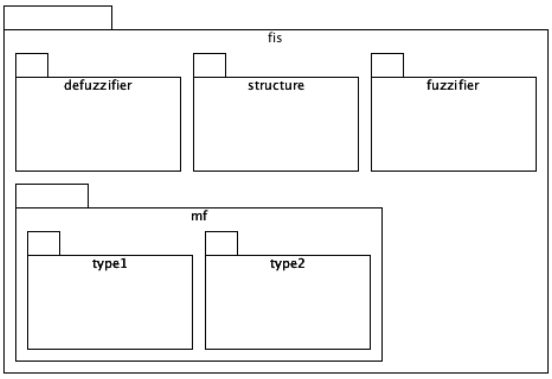

Figura 3.1: Diagrama de paquetes UML de JT2FIS.

Una de las ventajas de esta biblioteca es la herencia una de las capacidades del paradigma de programación orientado a objetos para integrar nuevas funciones de membresía. Se puede ver este enfoque en la Figura 3.13 donde MemberFunction es una clase abstracta que nos permite extender tantas funciones de membresía como se requieran.

En la versión actual de JT2FIS se cuenta con siete diferentes funciones de membresía tipo 2 y una base de funciones de membresía de tipo 1 disponibles(Gauss, Triangular, Trapezoidal).

Tabla 3.1: Contenido de la Biblioteca de Clases JT2FIS

Paquete fis	Clase FIS		
	Clase Mamdani		
	Clase Sugeno		
	Paquete defuzzifier	Clase Defuzzifier	
		Clase TypeReducer	
	Paquete fuzzifier	Clase Fussify	
		Clase PointFuzzy	
	Paquete mf	Clase MemberFunction	
		Paquete type1	Clase BellMemberFunction
			Clase GaussMemberFunction
			Clase TrapezoidalMemberFunction
			Clase TriangulerMemberFunction
		Paquete type2	Clase GaussCutMemberFunction
			Clase GaussUncertaintyMeanMemberFunction
			Clase GaussUncertaintyStandardDesviationMemberFunction
			Clase TrapaUncertaintyMemberFunction
			Clase TrapezoidalUncertaintyMemberFunction
			Clase TriangulerUncertaintyMemberFunction
			Clase TriUncertaintyMemberFunction
	Paquete structure	Clase Input	
		Clase Output	
		Clase Rule	
	Paquete util	Clase LinSpace	
		Clase Sorter	
		Clase Utilities	

En las Tablas 3.2 y 3.3 listan las funciones de membresía tipo 1 y tipo 2 con las que cuenta la biblioteca respectivamente.

La Figura 3.3 representa a JT2FIS, expresada en Lenguaje de Modelado Unificado(UML). Muestra la estructura básica de clases de JT2FIS.

La clase Fis esta compuesta por un conjunto de entradas (clase Input), salidas (clase Output) y reglas (clase Rule). Cada entrada y salida contienen un conjunto de funciones de membresía (clase MemberFunction) y cada uno de ellas podría ser un tipo específico de función de membresía.

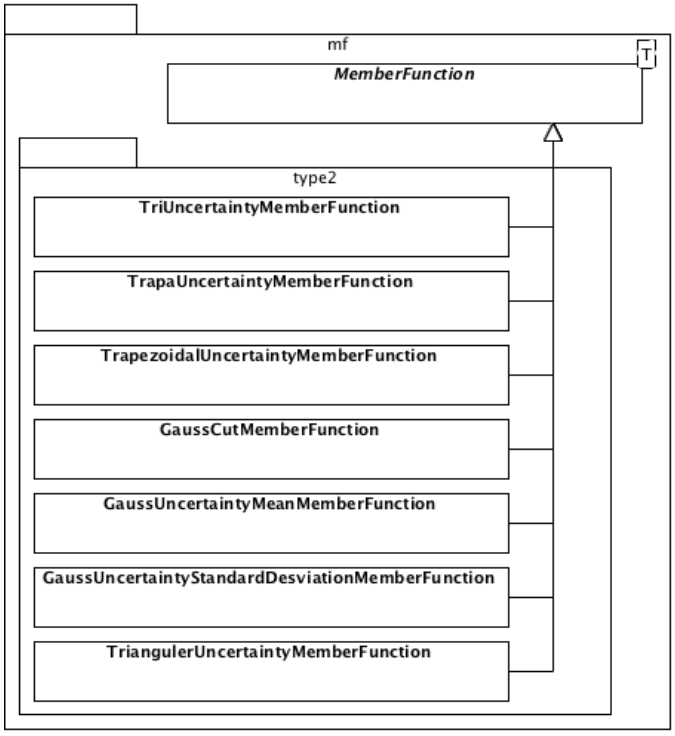

Figura 3.2: Tipos de funciones de membresía.

Tabla 3.2: Funciones de Membresía Tipo 1 en JT2FIS.

Funciones de Membresía Tipo 1	**Descripción**
BellMemberFunction	Parámetros=[a b c]
GaussMemberFunction	Parámetros=[desviación media]
TrapezoidalMemberFunction	Parámetros=[a b c d]
TriangulerMemberFunction	Parámetros=[a b c]

La clase Fis ofrece como principal característica un Sistema Difuso de Inferencia de tipo 2 por intervalos. La clase Fis, es una clase abstracta que establece el núcleo de la funcionalidad de la cual extienden la clase Mamdani y Sugeno, clases concretas que implementa diferentes enfoques.

Se puede ampliar los tipos de función de membresía extendiendo de la clase MemberFunction que es una clase abstracta. La Figura 3.4 muestra la funciones de membresía disponibles.

Tabla 3.3: Funciones de Membresía Tipo 2 en JT2FIS.

Funciones de Membresía Tipo 2	Descripción
GaussCutMemberFunction	Parámetros=[desviación media alfa]
GaussUncertaintyMeanMemberFunction	Parámetros=[desviación media1 media2]
GaussUncertaintyStandardDesviationMemberFunction	Parámetros=[desviación1 desviación2 media]
TrapaUncertaintyMemberFunction	Parámetros=[a1 b1 c1 d1 a2 b2 c2 d2 alfa]
TrapezoidalUncertaintyMemberFunction	Parámetros=[a d sa sd sn alfa]
TriangulerUncertaintyMemberFunction	Parámetros=[a c sa sc]
TriUncertaintyMemberFunction	Parámetros=[a1 b1 c1 a2 b2 c2]

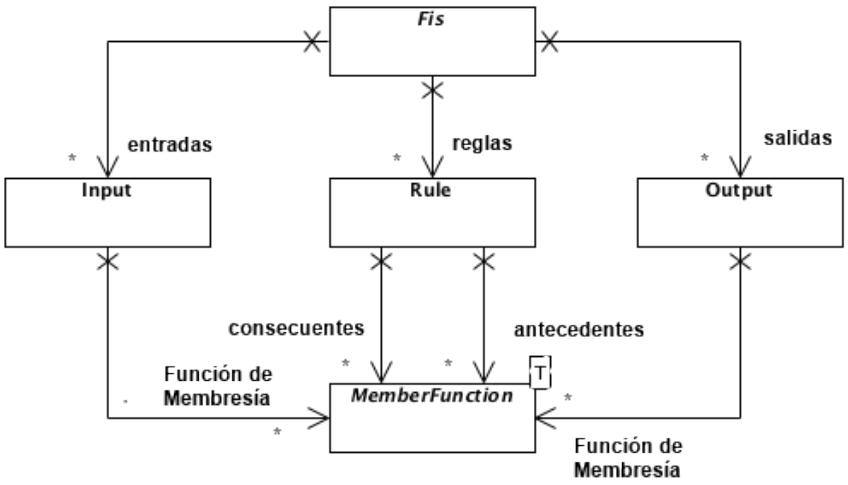

Figura 3.3: Estructura básica de clases JT2FIS.

3.1.2. Construcción de un sistema de inferencia tipo 2 por intervalos

Con el fin de crear un FIS con JT2FIS en el siguiente ejemplo, estamos considerando el siguiente procedimiento:

1. Crear una nueva instancia de la clase FIS.
2. Crear y configurar nuevas entradas; instancias de la clase Input para añadir al FIS.
3. Creación y configuración de nuevas; salidas instancias de la clase Output para añadir al FIS.

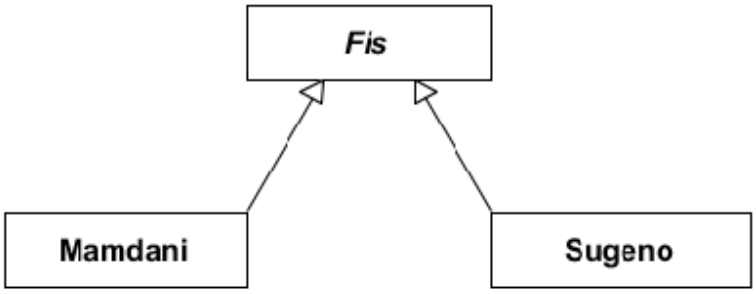

Figura 3.4: Tipos de sistemas de inferencia difuso.

4. Crear y configurar nuevas reglas; instancias de la clase Rule para añadir al FIS.

5. Evaluar inferencia con el FIS construido.

Creación de una nueva instancia FIS

Para construir un sistema de inferencia tipo 2 por Intervalos, primero tenemos que crear una instancia de la clase Mamdani. En el Listado 3.1 se muestra como hacer esto.

Listado 3.1: Creación de una nueva instancia FIS

```
Mamdani fis = new Mamdani("FIS");
```

Agregando entradas a Fis

Después de haber creado una instancia de la clase Mamdani, se puede agregar entradas creando instancias de la clase Input. Para cada instancia de Input, se puede agregar funciones de membresía estas representan las entradas de las variables lingüísticas en el sistema. Las funciones de membresía pueden ser de diferentes tipos, dependiendo de la aplicación.

Listado3.2 muestra como agregar funciones de membresía a una entrada, y como agregar una entrada a fis.

Listado 3.2: Agregando una nueva entrada a FIS

```
//Creando una nueva instancia de Input
Input input =new Input("input");
//Configurando los rangos de Input
input.setLowerRange(-20.0);
input.setSuperiorRange(20.0);
//Creando una funcion de membresia a la entrada
GaussUncertaintyMeanMemberFunction inputMF = new
   GaussUncertaintyMeanMemberFunction("InputMF");
//Configurando la funci\'on de membres\'ia para la entrada
inputMF.setSigma(-6.0);
inputMF.setMean1(-12.0);
inputMF.setMean2(-18.0);
//Agregando la funcion de membresia a la entrada
input.getMemberFunctions().add(inputMF);
//Agregando la entrada a fis
fis.getInputs().add(input);
```

Agregando salidas a FIS

Como tercer paso, se puede agregar las salidas creando instancias de la clase Output. De la misma manera que en las entradas, para cada instancia de Output, se puede agregar funciones de membresía , estas pueden representar las variables lingüísticas de salida en el sistema. Las funciones de membresía podrían ser de diferentes tipos, dependiendo de la aplicación.

En el listado3.3 muestra como agregar una función de membresía a una salida, y como agregar una salida a fis.

Listado 3.3: Agregando una nueva salida a FIS

```
//Creando una nueva instancia de Output
Output output=new Output("output");
// Configurando rangos de Output
output.setLowerRange(-1.0);
output.setSuperiorRange(1.0);
// Creando una funcion de membresia para la salida
GaussUncertaintyMeanMemberFunction outputMF=new
   GaussUncertaintyMeanMemberFunction("OutputMF");
//Configurando la funci\'on de membres\'ia de salida
outputMF.setSigma(-1.05);
outputMF.setMean1(-0.65);
outputMF.setMean2(-0.35);
//Agregando funcion de membresia a la salida
output.getMemberFunctions().add(outputMF);
//Agregando la salida a fis.
fis.getOutputs().add(output);
```

Agregando reglas a FIS

Finalmente, se puede agregar las reglas a FIS. Cada regla esta compuesta por un conjunto de antecedentes y un conjuntos de consecuentes. Los antecedentes y los consecuentes son funciones de membresía que forman parte de las entradas y las salidas.

En el listado3.4 muestra como agregar un antecedente y un consecuente a la regla, y como agregar una regla a fis.

Listado 3.4: Agregando una nueva regla a FIS

```
//Creando una nueva instancia de regla
Rule rule =new Rule();
//Agregando antecedente a la regla
MemberFunction antecedent = fis.getInputs().get(0).
   getMemberFunctions().get(0);
rule.getAntecedents().add(antecedent);
//Agregando consecuente a la regla
MemberFunction concecuent = fis.getOutputs().get(0).
   getMemberFunctions().get(0);
rule1.getConsequents().add(concecuent);
//Agregando regla a fis.
fis.getRules().add(rule);
```

Evaluando inferencia con FIS

Una vez que el sistema está construido, se puede utilizar para evaluar valores de entrada y salida. JT2FIS tiene 5 formas diferentes para evaluar un FIS. Cada forma es un método de reducción (Centroide, Centro de sumas, Alturas, Alturas Modificadas, Centro de Conjuntos). Cada método tiene parámetros que se deben de configurar en función de las necesidades de la representación del problema.

En el Listado 3.5 muestra como configurar y evaluar un fis.

Listado 3.5: Evaluando inferencia con FIS

```
// Puntos a Discretizar
int points=101;
/* Creando una lista de valores de entrada (la longitud de "
```

```
   inputList" debe de ser igual al numero de entrada del FIS) */
double inputList[]={0.5};
/* Evaluando inputList con metodo Singleton-Centroide en fis,
   retorna un arreglo de tipo Defuzzy con los valores de la
   salida. */
ArrayList<Defuzzy>result=fis.evaluateFisSingletonCentroid(
   inputList,points,Fis.AndMethod.MIN,Fis.OrMethod.MAX,Fis.
   ImpMethod.MIN,Fis.AggMethod.MAX);
// Imprimiendo los Resultados.
System.out.println("Result:"+result.get(0).toString());
```

3.2. Componente JT2FISPanel

JT2FISPanel es un componente visual JAVA que extiende la funcionalidad de un componente JPanel de la biblioteca Swing de Java. Este panel es parte de la biblioteca JT2FIS; que facilita la configuración y visualización de un sistema de inferencia difuso. Visualmente, JT2FISPanel tiene 5 funcionalidades principales para manejar un FIS:

1. Inputs.
2. Outputs.
3. Funciones de Membresía.
4. Reglas.
5. Vistas.

JT2FISPanel muestra las entradas, salidas, funciones de membresía y reglas que conforman a un sistema de inferencia difuso, en forma de un árbol; esto facilita la forma de visualizar la estructura de un FIS, propuesta para resolver un problema planteado.

En la Figura 3.5 muestra las diferentes opciones de configuración, por ejemplo, los operadores lógicos usados en los métodos de evaluación de un FIS, el tipo de desfusificación seleccionado para un problema determinado.

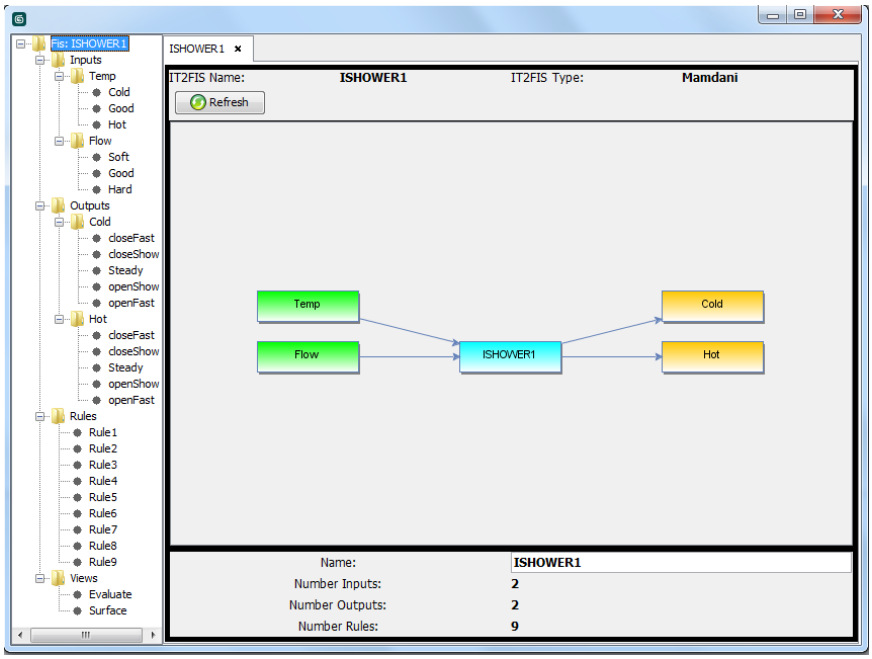

Figura 3.5: Pantalla Principal de JT2FISPanel

3.2.1. Configurando entradas con JT2FISPanel

En esta sección, se puede manejar los parámetros de una instancia de la clase Input(variables lingüísticas de entrada) para un sistema de inferencia difuso. En la figura 3.6 se muestra la configuración de un FIS llamado “ISHOWER” , tomando como ejemplo la entrada “Temp”. Se puede ver las diferentes funciones de membresía que forman parte de la entrada “Temp” se puede establecer diferentes atributos(Nombre, rango inferior y superior, etc.).

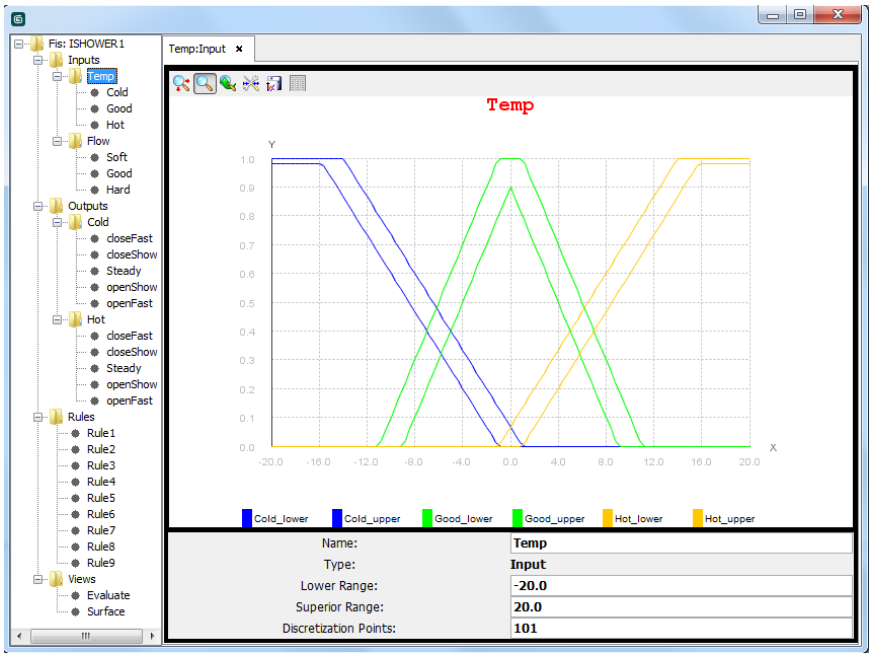

Figura 3.6: Configuración de la entrada "Temp" en el ejemplo ISHOWER.

3.2.2. Configurando salidas con `JT2FISPanel`

En esta sección, se puede manejar los parámetros de una instancia de la clase Output(variables lingüísticas de salida) para un sistema de inferencia difuso. En la figura 3.7 se muestra la configuración de un FIS llamado "ISHOWER", tomando como ejemplo la salida "Cold". Se puede ver las diferentes funciones de membresía que forman parte de la salida "Cold" y se puede establecer diferentes atributos (Nombre, rango inferior y superior, etc.).

3.2.3. Configurando funciones membresía con `JT2FISPanel`

En esta sección, se puede manejar los parámetros de las funciones de membresía(valores de las variables lingüísticas) de las entradas y las salidas. La figura 3.8 muestra la función de membresía "Good" para la entrada "Temp" para el ejemplo del FIS "ISHOWER" se puede observar , el valor de la variable lingüística seleccionada para la variable lingüística "Temp" y seleccionar el tipo de función de membresía. Una gráfica muestra el valor de la variable

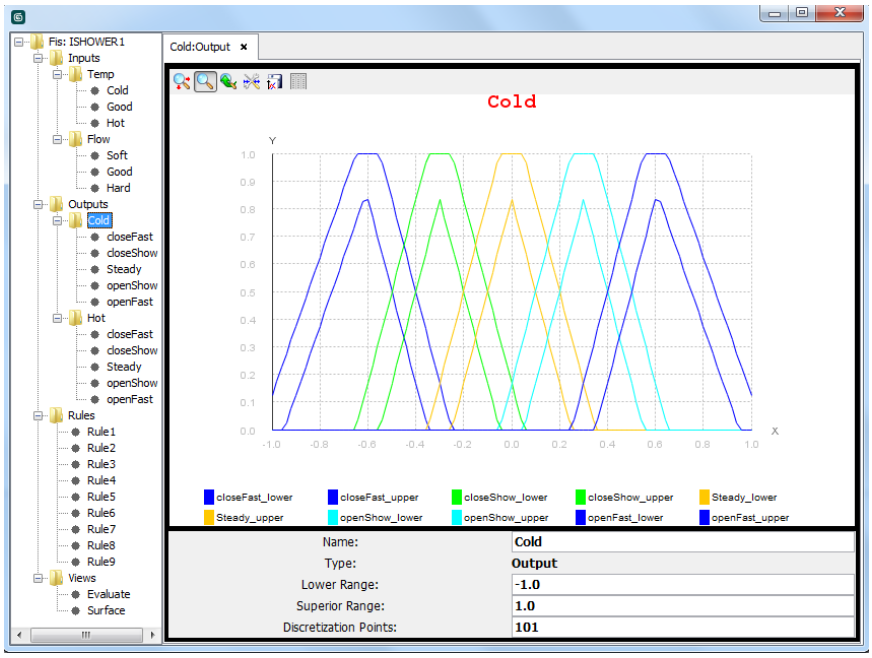

Figura 3.7: Configuración de la salida "Cold" en el ejemplo ISHOWER.

lingüística y muestra visualmente la configuración.

3.2.4. Configurando reglas con `JT2FISPanel`

En esta sección se puede administrar las reglas del sistema de inferencia difuso. La figura 3.9 muestra la regla número uno para el ejemplo FIS "ISHOWER". Se puede ver las entradas y salidas que forman la regla. En el panel inferior izquierdo, se puede ver las entradas con las funciones de membresía que la conforman; estas funciones son las que se seleccionan para formar los antecedentes de una regla. En la parte inferior derecha del panel, se puede ver las salidas con sus respectivas funciones de membresía que las conforman; estas son las que se seleccionan para formar los consecuentes de una regla. Las entradas están conectadas por un operador lógico(AND o OR). La regla formada se muestra en centro del componente.

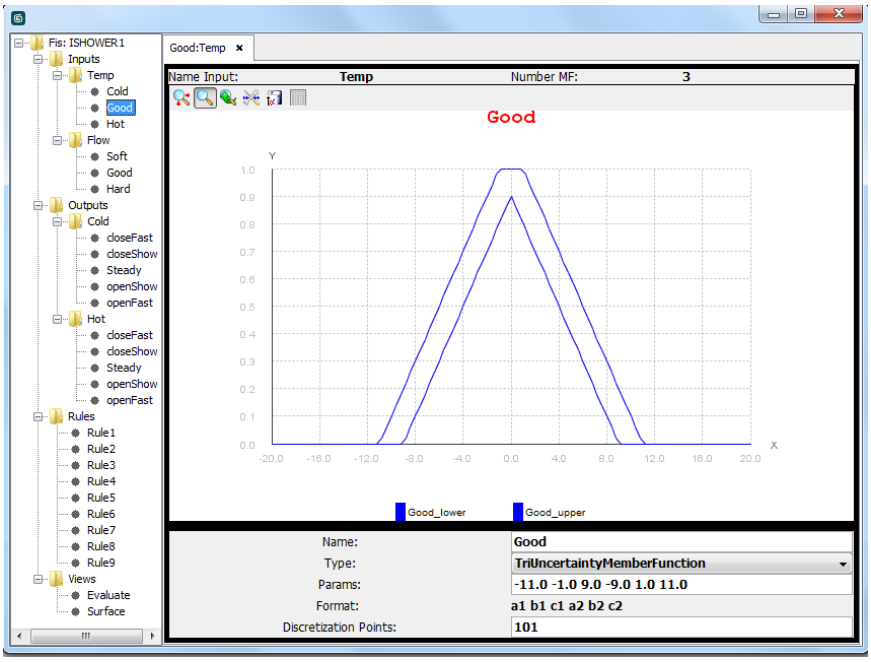

Figura 3.8: Configuración de la función de membresía "Good" para la entrada "Temp" en el ejemplo "ISHOWER".

3.2.5. Evaluando con `JT2FISPanel`

La Figura 3.10 muestra la configuración para evaluar un sistema de inferencia difuso. Puede elegir diferentes métodos para desfusificar, el número de puntos a discretizar, el método and,or, implicación y agregación para realizar la inferencia.

`JT2FISPanel` tiene 2 diferentes opciones para seleccionar los puntos que se desean evaluar. Puede agregar puntos de forma manual como se muestra en la Figura 3.11, o puede importar un archivo CSV para evaluar un conjunto de puntos automaticamente. `JT2FISPanel` puede exportar los resultados obtenidos en un archivo con formato CSV.

3.2.6. Utilizando `JT2FISPanel`

`JT2FISPanel` extiende la funcionalidad de un JPanel en Java de la biblioteca Swing. Por lo que se puede usarla en diferentes aplicaciones, utilizándola a nuestra conveniencia. En el

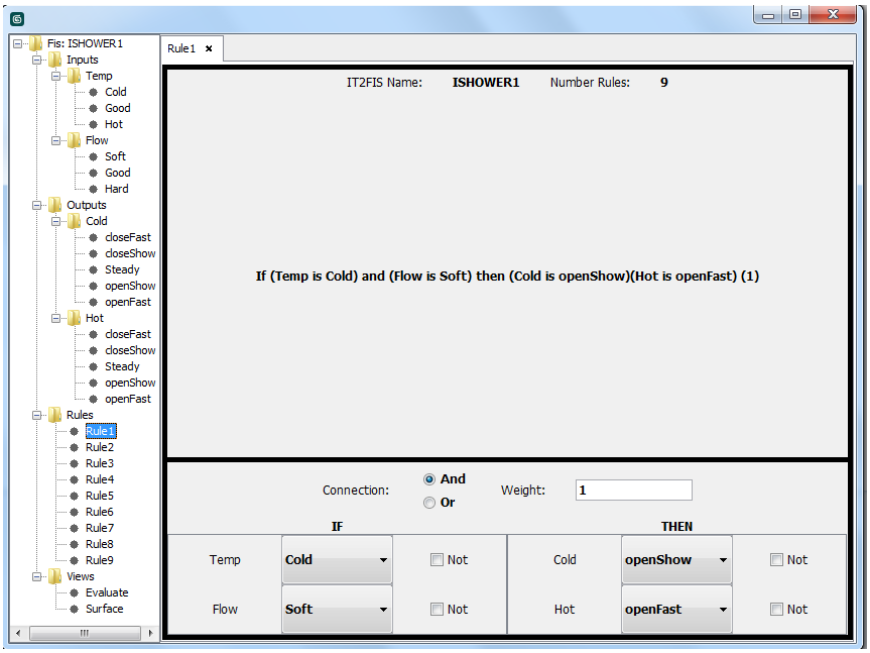

Figura 3.9: Configuración de la regla uno en el ejemplo ISHOWER.

listado 3.6, se puede ver como se agrega una instancia de `JT2FISPanel` a un JFrame.

Listado 3.6: Agregando `JT2FISPanel` a JFrame

```
// Creando una nueva instancia de JFrame.
JFrame frame = new JFrame("Example JT2FISPanel");
// Opcional: Seleccionar que sucede cuando el frame se cierra
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Creando un nueva instancia de JT2FISPanel.
JT2FISPanel panelJT2FIS = new JT2FISPanel();
//Agregando instancia de JT2FISPanel en instancia JFrame.
frame.getContentPane().add(panelJT2FIS, BorderLayout.CENTER);
// Dimension del frame.
frame.pack();
// Mostrar frame.
frame.setVisible(true);
```

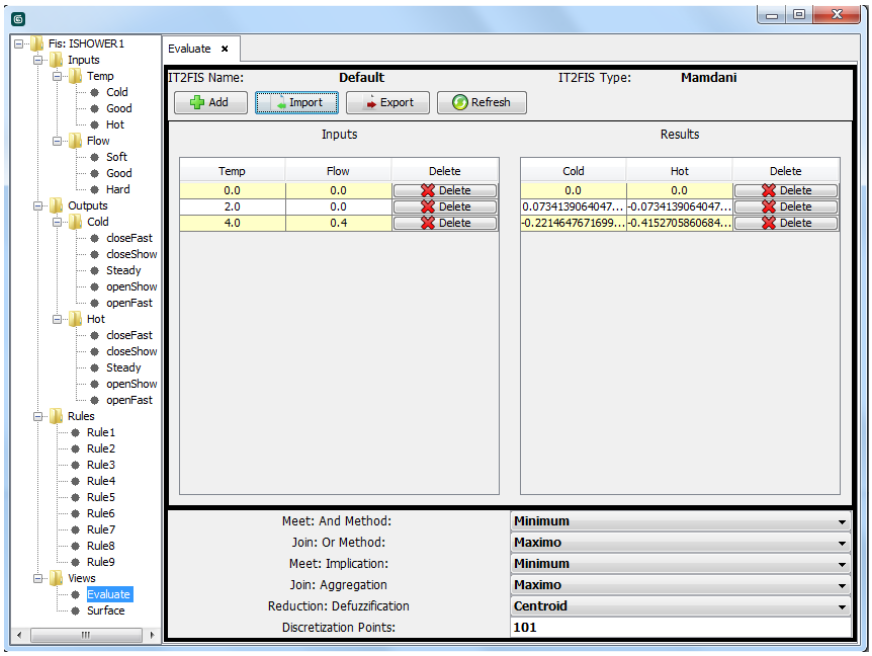

Figura 3.10: Evaluar un sistema de inferencia utilizando el ejemplo ISHOWER.

3.3. Método JT2FISClustering

JT2FISClustering es una biblioteca de clases desarrollada en Java. El propósito principal de esta biblioteca es realizar el minado de grandes conjuntos de datos para poder configurar de una manera automática y lo mas eficientemente posible diferentes Sistemas de Inferencia Tipo 2 aplicando el paradigma de programación orientada a objetos, a poyándose en la biblioteca de clases JT2FIS.

3.3.1. Diseño de JT2FISClustering

JT2FISClustering esta integrado por una estructura en paquetes que contienen una colección de clases. El contenido y la organización de estos paquetes se muestran en la Tabla 3.4.

La Figura 3.12 muestra un diagrama de paquetes UML de la biblioteca de clases JT2FISClustering.

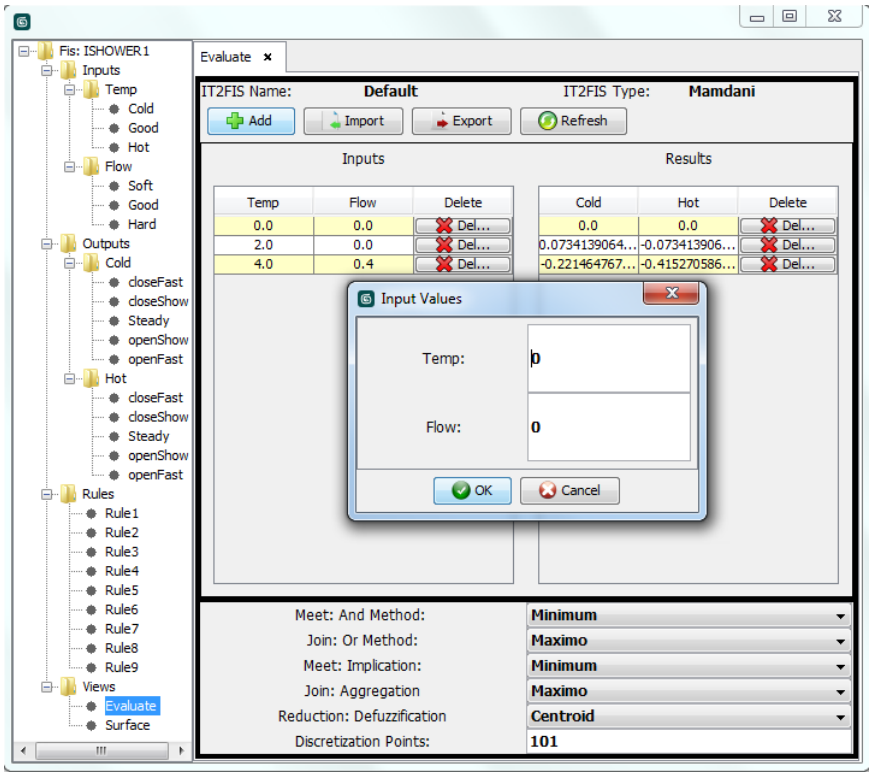

Figura 3.11: Agregando punto para evalar el ejemplo ISHOWER.

Tabla 3.4: Contenido de la Biblioteca de Clases `JT2FISClustering`

Paquete JT2FISClustering	Paquete Clustering	**Clase Cluster**
		Clase FuzzyCMeans
	Paquete Generate	**Clase GenerateMamdaniFis**
	Paquete util	**Clase MersenneTwister**
		Clase Utilities

La colección de clases esta organizada en paquetes de código, que encapsulan su comportamiento.

Una de las ventajas de esta biblioteca es la herencia una de las capacidades del paradigma de programación orientada a objetos, para integrar nuevos métodos de clusterisación. Se puede observar esto en la Figura 3.13 donde Cluster es una clase abstracta que nos permite extender tantos métodos de clusterisación como se requieran.

En la versión actual de `JT2FISClustering` se cuenta con 3 diferentes funciones de mem-

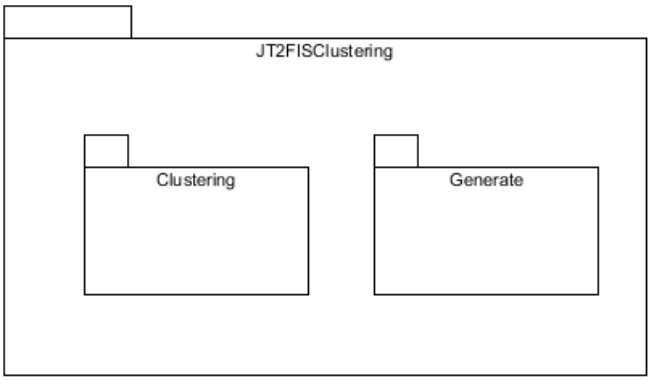

Figura 3.12: Diagrama de paquetes UML de `JT2FISClustering`.

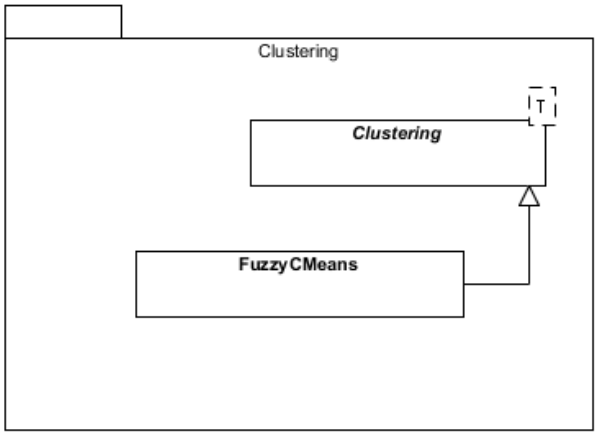

Figura 3.13: Tipos de métodos de clusterisación.

bresia, estas se pueden observar en la Tabla 3.5. Fuzzy c-Means es el método de clusterizacion por default.

Tabla 3.5: Funciones de Membresia de `JT2FISClustering`.

Tipo de Clusterisación	Funciones de Membresia Tipo 2	Descripción
Fuzzy c-Means	GaussCutMemberFunction	Parámetros=[entradas salidas incertidumbre]
Fuzzy c-Means	GaussUncertaintyMeanMemberFunction	Parámetros=[entradas salidas] Params=[entradas salidas incertidumbre]
Fuzzy c-Means	GaussUncertaintyStandardDesviationMemberFunction	Parámetros=[entradas salidas] Params=[entradas salidas incertidumbre]

3.3.2. Utilizando JT2FISClustering

Con el fin de crear un FIS a través de la clusterisación de un conjunto de datos utilizando JT2FISClustering en el ejemplo siguiente, se considera el siguiente procedimiento:

1. Crear una nueva instancia de la clase del método de agrupamiento de datos que se requiera utilizar.
2. Crear una nueva instancia de la clase GenerateMamdaniFis.
3. Creación de una lista con todo el conjunto de datos para las entradas.
4. Creación de una lista con todo el conjunto de datos para las salidas.
5. Generar FIS, seleccionando el tipo de función de membresia que se requiera.

Creación de una nueva instancia del método fuzzy c-means.

Para poder elegir el método de agrupamiento de datos que se desea utilizar, se tiene que crear una instancia de la clase que contenga el tipo de agrupamiento de datos, para este ejemplo utilizaremos el método Fuzzy c-means. En el Listado 3.7 se muestra como hacer esto.

Listado 3.7: Creación de una nueva instancia FuzzyCMeans

```
// Numero de grupos(clusters) que se requieren.
int numberCluster=3;
FuzzyCMeans fcm=new FuzzyCMeans(numberCluster);
```

Creación de una nueva instancia GenerateMamdaniFis

Para construir un Sistema de Inferencia Tipo 2 por Intervalos, primero tenemos que crear una instancia de la clase GenerateMamdaniFis. El cual recibe como parametro un objeto que representa el tipo de clusterisación que se desea utilisar. En el Listado 3.8 se muestra como hacer esto.

Listado 3.8: Creación de una nueva instancia GenerateMamdaniFis

```
//Donde fcm es una instancia de la clase FuzzyCMeans.
GenerateMamdaniFis gMamFis=new GenerateMamdaniFis(fcm);
```

Lista de Entradas

Después de haber creado la instancia de la clase GenerateMamdaniFis, es necesario crear una lista(ArrayList) que contenga todos los datos de cada una de las entradas que compondrán el FIS. Para este ejemplo, se tomaron 2 entradas y 100 datos para cada entrada, los cuales fueron generados de forma aleatoria con la ayuda de la clase Random de Java, como se puede observar en el Listado 3.9

Listado 3.9: Configurando la lista de entradas con sus respectivos datos.

```
//Creando lista del conjunto de entradas.
ArrayList<ArrayList<Double>> inputList=new ArrayList<ArrayList<
   Double>>();
//Creando instancia de la clase Random de Java.
Random r=new Random();
//Variable con el numero de entradas del FIS.
int numberInputs=2;
//Variable con el numero de datos de cada entrada.
```

```
int numberDatas=100;
for(int i=0;i<numberInputs;i++){
        //Lista que contiene todos los datos de una sola
           entrada.
        ArrayList<Double> input=new ArrayList<Double>();
        for(int j=0;j<numberDatas;j++){
                //Agregando dato, generado aleatoriamente.
                input.add(r.nextDouble());
        }
        //Agregando lista de datos de una entrada a la lista
           del conjunto de entradas.
        inputList.add(input);
}
```

Lista de Salidas

Después es necesario crear una lista(ArrayList) que contenga todos los datos de cada una de las salidas que compondrán el FIS. Para este ejemplo, se tomaron 2 salidas y 100 datos para cada entrada, los cuales fueron generados de forma aleatoria con la ayuda de la clase Random de Java, como se puede observar en el Listado 3.10

Listado 3.10: Configurando la lista de salidas con sus respectivos datos.

```
//Creando lista del conjunto de salidas.
ArrayList<ArrayList<Double>> outputList=new ArrayList<ArrayList
   <Double>>();
//Creando instancia de la clase Random de Java.
Random r=new Random();
```

```
//Variable con el numero de salidas del FIS.
int numberOutputs=2;
//Variable con el numero de datos de cada salida.
int numberDatas=100;
for(int i=0;i<numberOutputs;i++){
        //Lista que contiene todos los datos de una sola salida
           .
        ArrayList<Double> output=new ArrayList<Double>();
        for(int j=0;j<numberDatas;j++){
                //Agregando dato, generado aleatoriamente.
                output.add(r.nextDouble());
        }
        //Agregando lista de datos de una salida a la lista del
            conjunto de salidas.
        outputList.add(output);
}
```

Generar FIS

Una vez que se tengan creadas la lista de entradas y salidas, se puede generar el FIS. `JT2FISClustering` actualmente solo crea sistemas de inferencia difuso tipo 2 de tipo Mamdani. Los tipos de función de membresia que puede tener se pueden observar en la Tabla 3.5. `JT2FISClustering` proporciona la opción de poder elegir la incertidumbre de forma manual o dejando la opción que `JT2FISClustering` la calcule automáticamente.En el Listado 3.11 se muestra como generar un FIS, en este ejemplo se eligió la función de membresia "GaussUncertaintyMeanMemberFunction".

Listado 3.11: Generando sistema de inferencia difuza tipo Mamdani.

```
//Variable para establecer incertidumbre
double uncertainty=0.8;
//Llamando el metodo generar Fis tipo Mamdani con funcion de
   membresia ''GaussUncertaintyMeanMemberFunction'' eligiendo
   incertidumbre.
Mamdani fis=gMamFis.
   generateMamdaniFisGaussUncertaintyMeanMemberFunction(
   inputList, outputList,uncertainty);
System.out.println(fis.toString());


//Llamando el metodo generar Fis tipo Mamdani con funcion de
   membresia ''GaussUncertaintyMeanMemberFunction'' con
   incertidumbre calculada.
Mamdani fis2=gMamFis.
   generateMamdaniFisGaussUncertaintyMeanMemberFunction(
   inputList, outputList);
System.out.println(fis2.toString());
```

3.4. Componente JT2FISClusteringPanel

JT2FISClusteringPanel es un componente visual JAVA que extiende la funcionalidad de un componente JPanel de la biblioteca Swing de Java. Este panel es parte de la biblioteca JT2FISClustering; que facilita la creación y visualización de un sistema de inferencia difuso

a partir de un método de clusterisación.

La Figura 3.14 muestra la interfaz gráfica de usuario de `JT2FISClusteringPanel`. En esta el usuario puede establecer diferentes métodos de generación de funciones de membresia y métodos de clusterisación con el fin de aplicarlo en el proceso de generación de un sistema de inferencia difuso según el propósito que se requiera.

Fuzzy c-Means es el método de clusterisación por default. En la Tabla 3.5 de la sección anterior se puede observar un listado de las funciones de membresia de tipo 2 disponibles en `JT2FISClustering`.

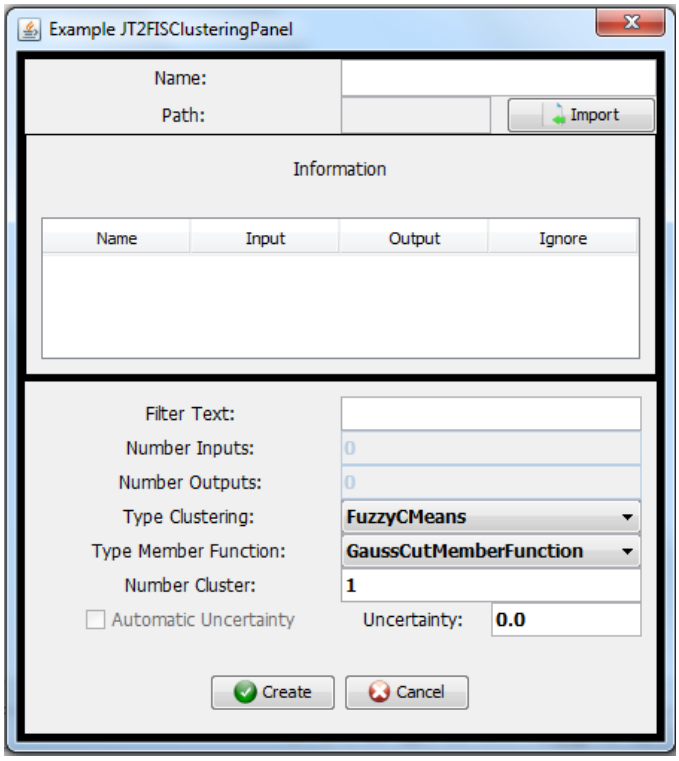

Figura 3.14: Interfaz Gráfica de Usuario de `JT2FISClusteringPanel`.

`JT2FISClusteringPanel` genera un FIS a partir de un archivo CSV. El usuario puede seleccionar las entradas y salidas de la columnas de datos contenidas en el archivo CSV para configurar el proceso de generación tal y como se muestra en la Figura 3.15. El archivo CSV tiene un único requisito y es que el primer renglón debe de contener los nombres de las

entradas y salidas, ya que JT2FISClusteringPanel toma este primer renglón para configurar los nombres que llevaran las entradas y salidas del FIS.

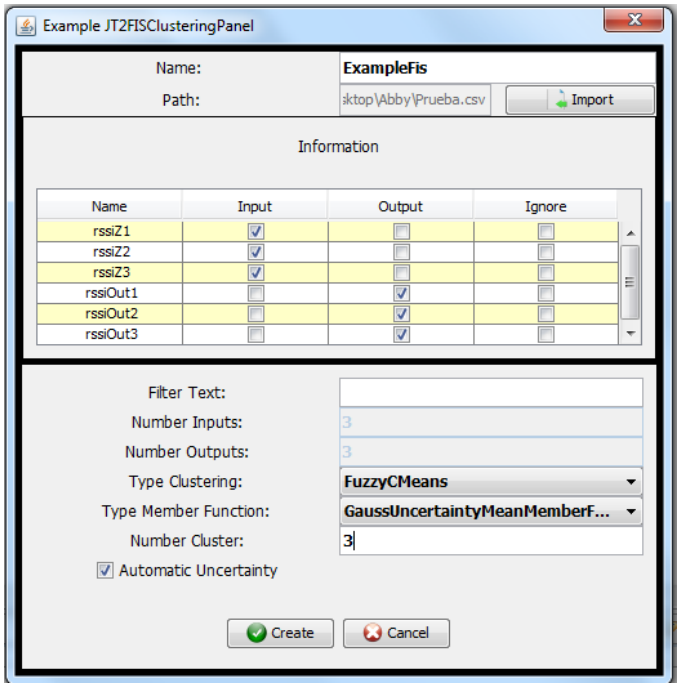

Figura 3.15: Configurando JT2FISClusteringPanel para Crear FIS.

3.4.1. Utilizando JT2FISClusteringPanel

JT2FISClusteringPanel extiende su funcionalidad de la clase JPanel en Java de la biblioteca Swing. Esto permite agregarlo en cualquier proyecto de una forma sencilla ya sea en un JFrame,JDialog o lo que se requiera según el objetivo de la persona que lo desee utilizar. En el listado 3.12 se muestra un ejemplo de como se puede agregar una instancia de JT2FISClusteringPanel a una instancia de la clase JDialog de la biblioteca Swing.

Listado 3.12: Llamando a JT2FISClusteringPanel.

```
//Creando una nueva instancia de JDialog.
final JDialog dialog=new JDialog();
dialog.setTitle("Ejemplo␣JT2FISClusteringPanel");
```

```
dialog.setBounds(100,100,450,500);
//Creando una nueva instancia de JT2FISClusteringPanel.
final JT2FISClusteringPanel clusterPanel= new
   JT2FISClusteringPanel();
//Agregando la instancia de JT2FISClusteringPanel en la
   instancia de la clase JDialog.
dialog.add(clusterPanel);
//Mostrar JDialog
dialog.setVisible(true);
//Agregando acciones de JT2FISClusteringPanel.
clusterPanel.setActionListener(new
   OnActionsJT2FISClusteringListener() {
  //Accion cancelar.
  public void onCancel(int selectOption) {
    dialog.setVisible(false);
        dialog.dispose();
  }
  //Accion crear.
  public void OnApprove(int selectOption, Fis fis) {
        //Checar si el FIS fue creado correctamente.
    if(selectOption==JT2FISClusteringPanel.APPROVE_OPTION){
      //Imprimir informacion de la instancia FIS.
          System.out.println(clusterPanel.getFis().toString());
          dialog.setVisible(false);
          dialog.dispose();
        }
```

```
  }
});
```

3.5. Aplicaciones JT2FISPanel y JT2FISClusteringPanel

Utilizando los componentes JT2FIS en aplicaciones visuales es útil para facilitar la interacción del usuario con un FIS. JT2FISPanel podría utilizarse para mostrar la estructura de un FIS y ayudar a configurar el sistema de inferencia difuso. Los componentes pueden ser incorporados en cada aplicación de escritorio Java Swing. La Figura 3.16 muestra un proceso de minería de datos utilizando JT2FISClusteringPanel y JT2FISPanel en aplicaciones visuales.

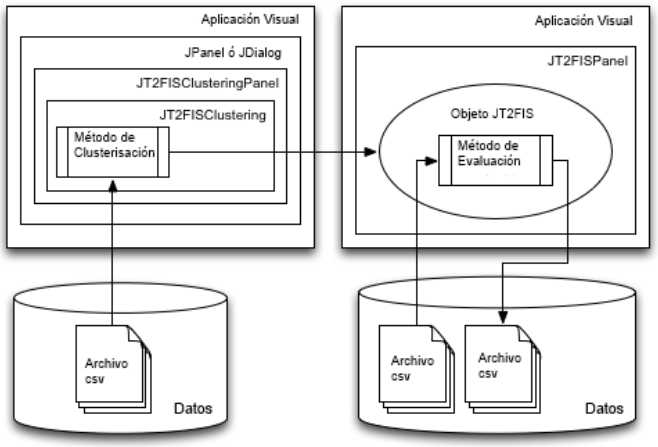

Figura 3.16: Proceso de minería de datos usando JT2FISClusteringPanel y JT2FISPanel en aplicaciones visuales.

El JT2FISClusteringPanel es beneficioso para facilitar la creación de un FIS a través proceso de minería de datos. Usando métodos de agrupamiento, JT2FISClusteringPanel utiliza datos reales para construir una configuración de un FIS. El FIS generado puede ser enviado a JT2FISPanel para poder ser visualizado. Con la vinculación de ambos componentes, se puede producir y analizar un FIS antes de utilizarlo en las soluciones de software

inteligente.

3.6. La Extensión JT2FISNetLogo

JT2FISNetLogo es una extensión que permite la comunicación entre en lenguaje de programación Java y el entorno de modelado multi-agente NetLogo[68]. Su objetivo es agregar la funcionalidad en NetLogo de poder utilizar sistemas de inferencia difusos tipo 2 en diferentes simulaciones, con la ayuda de la biblioteca de clases JT2FIS; por lo tanto JT2FISNetLogo es una interfaz que permite la comunicación entre JT2FIS y NetLogo.

3.6.1. Diseño de JT2FISNetLogo

JT2FISNetLogo esta integrado por una estructura en paquetes que contienen una colección de clases. JT2FISNetLogo depende totalmente de la biblioteca de clases JT2FIS por lo que su contenido y organización de estos paquetes es totalmente igual, su organización se muestra en la Tabla 3.1. Solo existe una pequeña diferencia, tiene un paquete extra llamado fuzzyextension que controla la comunicación entre JT2FIS y NetLogo.

La Figura 3.17 muestra un diagrama de paquetes UML de la biblioteca de clases JT2FISNetLogo. La colección de clases esta organizada en paquetes de código, que encapsulan su comportamiento.

En la version actual de JT2FISNetLogo se cuenta con siete diferentes funciones de membresía tipo 2 disponibles. En la Tabla 3.6 se listan las funciones de membresía tipo 2 con las que cuenta la biblioteca.

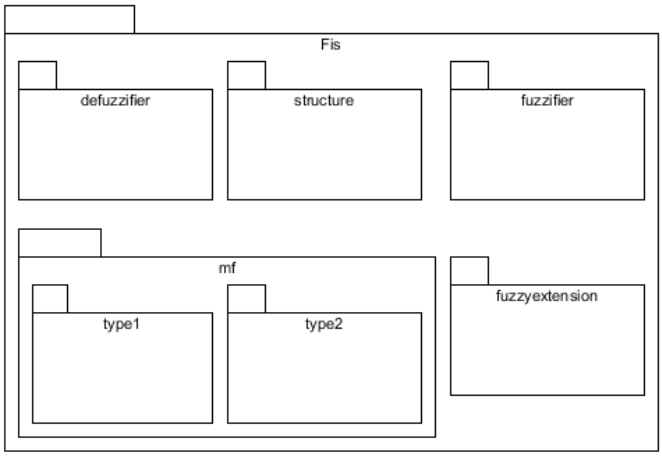

Figura 3.17: Diagrama de paquetes UML de `JT2FISNetLogo`.

Tabla 3.6: Funciones de Membresía Tipo 2 en `JT2FISNetLogo`.

Funciones de Membresía Tipo 2	**Descripción**
GaussCutMemberFunction	Parámetros=[nombre desviación media alfa]
GaussUncertaintyMeanMemberFunction	Parámetros=[nombre desviación media1 media2]
GaussUncertaintyStandardDesviationMemberFunction	Parámetros=[nombre desviación1 desviación2 media]
TrapaUncertaintyMemberFunction	Parámetros=[nombre a1 b1 c1 d1 a2 b2 c2 d2 alfa]
TrapezoidalUncertaintyMemberFunction	Parámetros=[nombre a d sa sd sn alfa]
TriangulerUncertaintyMemberFunction	Parámetros=[nombre a c sa sc]
TriUncertaintyMemberFunction	Parámetros=[nombre a1 b1 c1 a2 b2 c2]

3.6.2. Construcción de un FIS tipo 2 en NetLogo

Con el fin de crear un FIS con `JT2FISNetLogo` en el siguiente ejemplo, estamos considerando el siguiente procedimiento:

1. Configuración previa de NetLogo.
2. Crear una nueva instancia de la clase FIS.
3. Crear y configurar nuevas entradas; instancias de la clase Input para añadir al FIS.
4. Creación y configuración de nuevas; salidas instancias de la clase Output para añadir al FIS.
5. Crear y configurar nuevas reglas; instancias de la clase Rule para añadir al FIS.

6. Evaluar inferencia con el FIS construido.

Configuración previa de NetLogo.

Cada extensión NetLogo consiste en una carpeta con el mismo nombre que la extensión, en minúsculas. Por lo que se debe crear una carpeta con el mismo nombre que la extensión, esta debe de contener un archivo JAR con el mismo nombre de la carpeta. Por ejemplo puede crear una carpeta llamada jt2fisnetlogo con un archivo dentro llamado jt2fisnetlogo.jar.

Para instalar una extensión NetLogo para el uso de cualquier modelo, se debe poner la carpeta de la extensión en la carpeta "extensions" del directorio de NetLogo. O bien, puede simplemente mantener la carpeta de la extensión en la misma carpeta que el modelo que utiliza.

Para utilizar una extensión en un modelo,es necesario agregar la palabra reservada extensions al inicio del código, antes de declarar cualquier raza o variable, como se muestra en el Listado 3.13.

Listado 3.13: Incluyendo extension en el modelo.

```
extensions[jt2fisnetlogo]
```

Después de extensiones viene una lista de nombres de extensión entre corchetes. Por ejemplo:

Creación de una nueva instancia FIS.

Para construir un Sistema de Inferencia Tipo 2 por Intervalos, primero tenemos que crear una instancia de la clase Mamdani. En el Listado3.14 se muestra como hacer esto.

Listado 3.14: Creación de una nueva instancia FIS

```
;Creando un nuevo FIS
set fis jt2fisnetlogo:generateMamdaniFis "FIS"
```

Agregando entras a FIS.

Después de haber creado una instancia de la clase Mamdani, se puede agregar entradas. Para cada entrada, se puede agregar funciones de membresía estas representan las entradas de las variables lingüísticas en el sistema. Las funciones de membresía pueden ser de diferentes tipos, dependiendo de la aplicación.

En el Listado 3.15 muestra como agregar funciones de membresía a una entrada, y como agregar una entrada a un FIS.

Listado 3.15: Agregando una nueva entrada a FIS

```
;Creando una nueva entrada
set input jt2fisnetlogo:newInput "Input" 0 242
;Creando nuevas funciones de membresia
set inputMf1 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction "
   inputMf1" 25.54 59.42 60.62
set inputMf2 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction "
   inputMf2" 51.48 230.1 234.7
set inputMf3 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction "
   inputMf3" 30.18 119.3 121.8
set inputMf4 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction "
   inputMf4" 33.11 7.23 7.37
;Agregando las funciones de membresia  a la entrada
jt2fisnetlogo:addMemberFunctionToInput input inputMf1
```

```
jt2fisnetlogo:addMemberFunctionToInput input inputMf2
jt2fisnetlogo:addMemberFunctionToInput input inputMf3
jt2fisnetlogo:addMemberFunctionToInput input inputMf4
;Agregando la entrada a FIS
jt2fisnetlogo:addInput fis input
```

Agregando salidas a FIS.

Como siguiente paso, se puede agregar las salidas de la misma manera que en las entradas, para cada salida, se puede agregar funciones de membresía , estas pueden representar las variables lingüísticas de salida en el sistema. Las funciones de membresía podrían ser de diferentes tipos, dependiendo de la aplicación.

En el Listado 3.16 muestra como agregar una función de membresía a una salida, y como agregar una salida a un FIS.

Listado 3.16: Agregando una nueva salida a FIS

```
;Creando una nueva salida
set output jt2fisnetlogo:newOutput "Output" 0 253
;Creando nuevas funciones de membresia
set outputMf1 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction
   "outputMf1" 28.7 61.3 62.53
set outputMf2 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction
   "outputMf2" 56.42 234.41 239.15
set outputMf3 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction
   "outputMf3" 31.48 122.6 125.08
set outputMf4 jt2fisnetlogo:gaussUncertaintyMeanMemberFunction
```

```
   "outputMf4" 34.42 7.77 7.93
;Agregando las funciones de membresia a la salida
jt2fisnetlogo:addMemberFunctionToOutput output outputMf1
jt2fisnetlogo:addMemberFunctionToOutput output outputMf2
jt2fisnetlogo:addMemberFunctionToOutput output outputMf3
jt2fisnetlogo:addMemberFunctionToOutput output outputMf4
;Agregando la salida a FIS
jt2fisnetlogo:addOutput fis output
```

Agregando reglas a FIS.

Finalmente, se puede agregar las reglas a un FIS. Cada regla esta compuesta por un conjunto de antecedentes y un conjuntos de consecuentes. Los antecedentes y los consecuentes son funciones de membresía que forman parte de las entradas y las salidas.

En el Listado 3.17 muestra como agregar un antecedente y un consecuente a la regla, y como agregar una regla a un FIS.

Listado 3.17: Agregando una nueva regla a FIS

```
;Creando una nueva regla.
set rule1 jt2fisnetlogo:newRule 1 "AND"
;Agregando antedecentes a la regla.
jt2fisnetlogo:addAntecedent rule1 inputMf1
;Agregando consecuentes a la regla.
jt2fisnetlogo:addConsequent rule1 outputMf1
;Agregando la regla a FIS.
jt2fisnetlogo:addRule fis rule1
```

Evaluando inferencia con FIS.

Una vez que el sistema está construido, se puede utilizar para evaluar valores de entrada y salida. JT2FIS tiene 5 formas diferentes para evaluar un FIS. Cada forma es un método de reducción (Centroide, Centro de sumas, Alturas, Alturas Modificadas, Centro de Conjuntos). Cada método tiene parámetros que se deben de configurar en función de las necesidades de la representación del problema.

En el Listando 3.18 muestra como configurar y evaluar un FIS.

Listado 3.18: Evaluando inferencia con FIS

```
;Establecer puntos a discretizar
set points=101
;Configurar Metodo And 0=minimo 1=producto
set andMethod=0
;Configurar Metodo Or 0=maximo 1=probor
set orMethod=0
;Configurar Metodo Implicacion 0=minimo 1=producto
set impMethod=0
;Configurar Metodo Agregacion 0=maximo 1=suma 2=probor
set aggMethod=0
;Evaluar FIS con metodo de reduccion Centroide.
set outputList jt2fisnetlogo:evaluateFisSingletonCentroid fis
   [2] points andMethod orMethod impMethod aggMethod
```

Capítulo 4

Herramientas IMIX

En este capítulo se introducen la herramientas de modelado denominadas IMIX Tools, en particular, la herramienta Wíinik. Estas herramientas estás inspiradas principalmente en el enfoque de complejidad, agencia distribuida y en inteligencia computacional. Es una colección de prototipos experimentales que tienen la intención de evaluar diferentes estrategias de diseño enfocadas en el programador de aplicaciones y en el usuario.

Debido a que las herramientas son prototipos experimentales enfocadas mas en la creatividad que en un proceso de desarrollo formal de software basado en requerimientos, está fuera del alcance de este trabajo la descripción formal de los mismos. Aún así, este trabajo de tesis describe en particular la herramienta denominada Wíink, que fue desarrollada con la noble intención de mostrar la utilización del marco de trabajo a través de un prototipo inicial, y de la integración de algunas ideas.

4.1. Herramienta Wíinik

Wíinik, es una herramienta de diseño de simulaciones basados en agentes. El propósito de esta herramienta es dibujar las relaciones entre los agentes distribuidos y configurar una base de reglas como un atributo para un agente cognitivo. Usamos un sistema de inferencia difuso tipo 2 que podría ser utilizado como un sistema de toma de decisiones.

Una característica importante es la capacidad de exportar diseños a Netlogo que es un entorno de simulación orientado a agentes utilizado amplia mente para las ciencias sociales. Esta herramienta se centra en el modelado de simulaciones con herramientas de software visual con el fin de ayudar a los investigadores a diseñar sistemas multi-agentes en un nivel más alto de abstracción.

4.1.1. Arquitectura Wíinik

La arquitectura de Wíinik utiliza una arquitectura convencional de 3 capas como se muestra en la figura 4.1. La primera capa es la presentación de la interfaz gráfica del usuario. Consideramos una interfaz intuitiva, que ayude lo mas posible a crear modelos complejos de agentes. La capa de lógica de negocio contiene dos paquetes, el paquete *Agent* que implementa la lógica de la estructura de los agentes, para construir los agentes en Jade y NetLogo, y el paquete *FIS*, este implementa la lógica del sistema de inferencia difuso usando la biblioteca JT2FIS. La capa de acceso a datos contiene los paquetes que implementan la lógica de la estructura de comunicación y representación de la información.

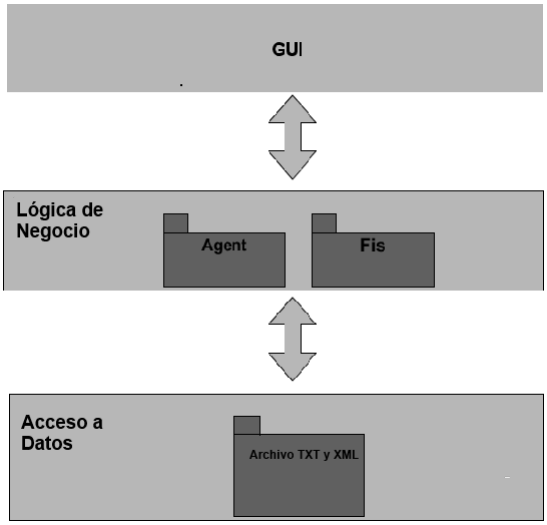

Figura 4.1: Arquitectura Wíinik

4.1.2. Elementos Wíinik

Wíinik es una herramienta de programación visual. Podemos representar un modelo a través de un simple diagrama con el cual podemos construir una estructura de agentes que representa a todo el sistema. Para la creación de un modelo podemos usar diferentes elementos gráficos: agentes, contenedores y relaciones.

Los contenedores son agentes que representa un nivel de grupo superior y todos los agentes contenidos en el interior están directamente relacionados con ellos. Los contenedores tienen las mismas propiedades que los agentes y podían comunicarse de la misma manera. De esta manera, los contenedores son agentes ilustrados con diferentes símbolos.

Los agentes están representados por un símbolo llamado "actor", que es el principal elemento para construir el modelo. Las relaciones se utilizan para representar las conexiones entre los agentes o contenedores.

En la Figura 4.2 se muestra la interfaz gráfica de usuario, y los elementos que conforman a Wíinik.

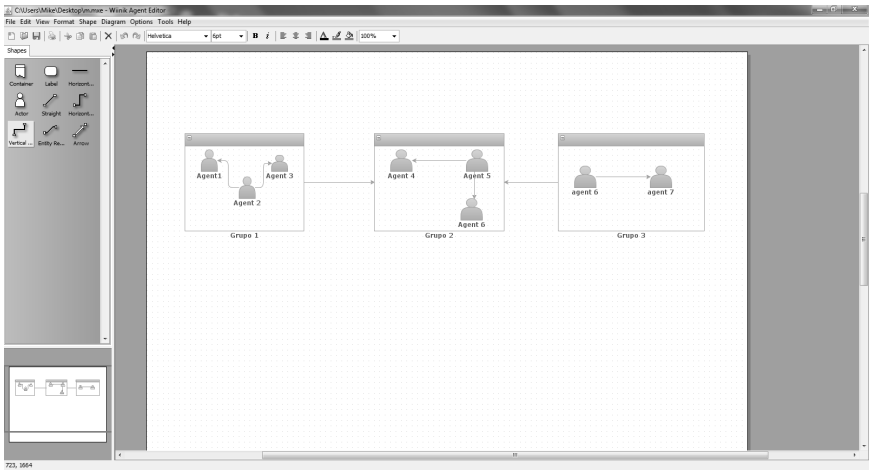

Figura 4.2: Interfaz gráfica de Usuario de Wiinik.

4.1.3. Características Wíinik

El trabajo visual de W ’iinik simplifica la conceptualización de las cosas reales, ayudando a construir la relación de una estructura más compleja. Las funcionalidades mas destacadas son:

1. Comunicación con JADE
2. Configurando FIS
3. Exportar a NetLogo

Estas características son descritas a detalle en las siguientes secciones.

4.1.4. Comunicación con JADE

Uno de los objetivos de Wíinik es ser capaz de construir una estructura de agentes con Jade. Actualmente la herramienta es capaz de establecer una comunicación con un agente en la plataforma Jade y enviar un modelo descrito en XML a un agente constructor. Este agente se encarga de construir la estructura de los agentes descritos en el modelo en la plataforma.

La Figura 4.3 muestra al agente WiinikGUI, este representa la aplicacion GUI en la plataforma jade. El agente WiinikGUI envia el modelo a otro agente llamado "Construct", este recive el modelo y lo construye el sistema.

Actualmente, sólo se establece una comunicación entre ellos y envía un archivo XML, con la que se pretende en el futuro interpretarlo para que un agente("Construct") sea capas de construir el modelo dentro de la plataforma JADE.

Para comunicarse con JADE es necesario seguir una serie de sencillos pasos. Plataforma Jade debe ser inicializada y luego tenemos que conectarnos con ella. Para lograr esto, es necesario ir a:

Tools > JADE > Connect.

Esto creará dentro de nuestra plataforma un agente llamado WiinikGUI como se muestra en la Figura 4.3, que se encarga de recoger el modelo y enviarlo.

Después de eso, tenemos que crear nuestra propia RMA , llamado WiinikRMA. Esto a través de: Tools > JADE > RMA

Esto nos permite crear una nueva clase de agente. El agente constructor en este caso llamado "Construct", como se muestra en la Figura 4.4, que será en el futuro quien va a construir todo el modelo en la plataforma JADE.

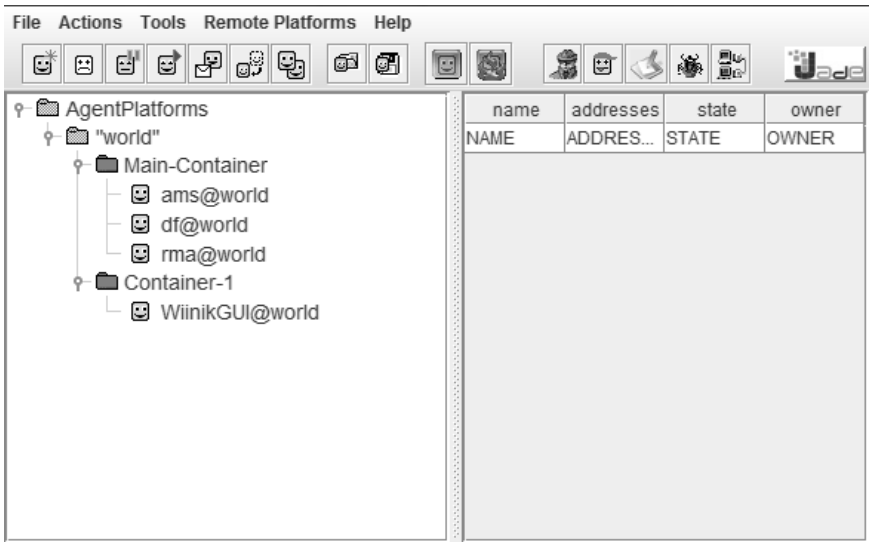

Figura 4.3: Conectando Wíinik con plataforma JADE

Finalmente enviamos el modelo a través de:

Tools > JADE > Message.

Esto hace que el agente recoga todos los modelos y lo envié a través de un mensaje en formato XML. El agente constructor construirá el modelo en plataforma.

El agente constructor responderá con un mensaje si fue posible o no construirlo. La Figura 4.5 muestra esta característica.

4.1.5. Configurando FIS

La herramienta Wíink tiene una interfaz gráfica de usuario que facilita la interacción del usuario para crear un sistema de inferencia difuso tipo 2. Para la configuración, el usuario simplemente necesita hacer doble clic sobre el agente que desea configurar su sistema de inferencia.Wíink implementa el panel JT2FISPanel que utiliza la biblioteca JT2FIS para configurar y evaluar los sistemas de inferencia difusos que se desean crear para el agente seleccionado. La descripción y la funcionalidad de este panel esta descrita en el sub-tema 3.2

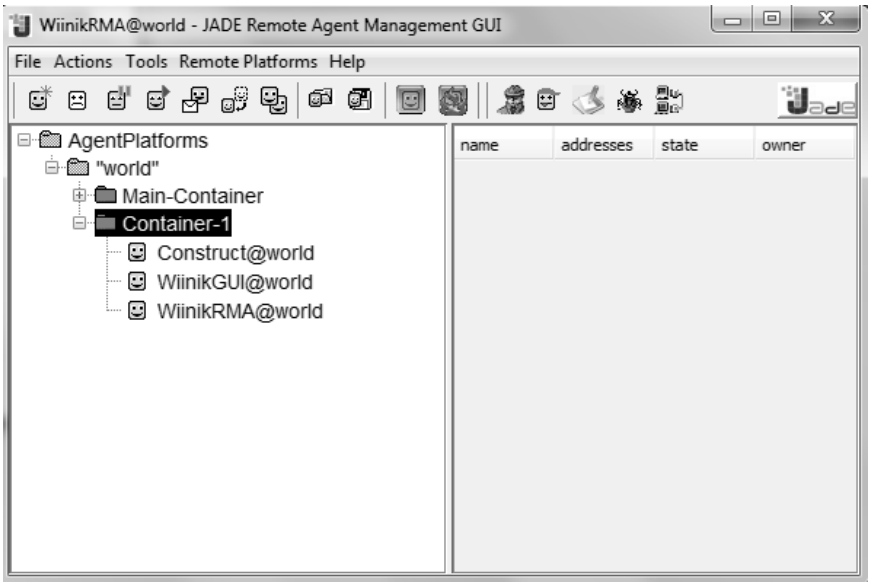

Figura 4.4: Crear un nuevo agente constructor.

de esta unidad.

4.1.6. Exportar a NetLogo

Para exportar a NetLogo, es necesario tener la extensión "fuzzy", que se ha mencionado anteriormente en el sub-tema 3.4 de esta unidad. Los pasos para exportar el código NetLogo es el siguiente:

Tools > NetLogo > Export.

La Figura 4.6 muestra este procedimiento.

Esto producirá un archivo con el código generado NetLogo, donde se encuentran los sistemas de inferencia de cada uno de los agentes. El usuario puede aplicar este código de una manera sencilla a diferentes simulaciones. También exportará las relaciones existentes entre los agentes.

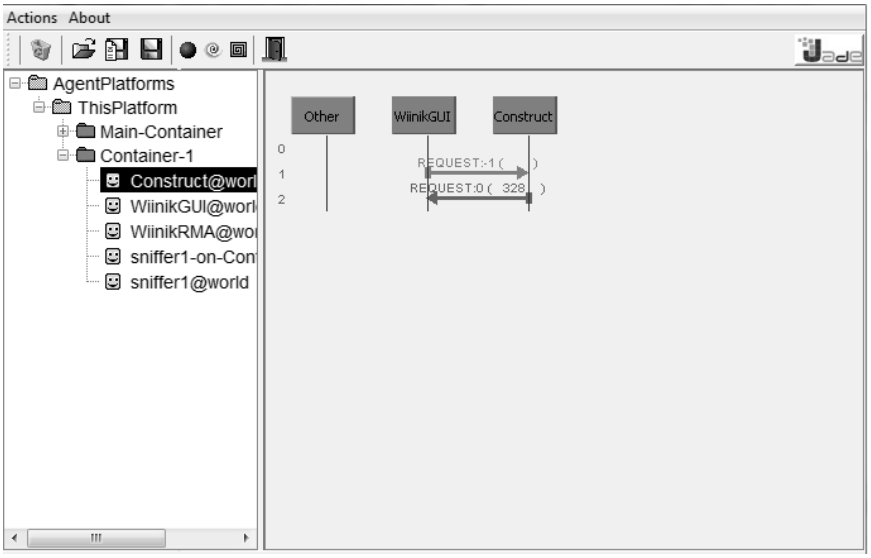

Figura 4.5: Enviando un modelo desde Wíinik al agente "Construct".

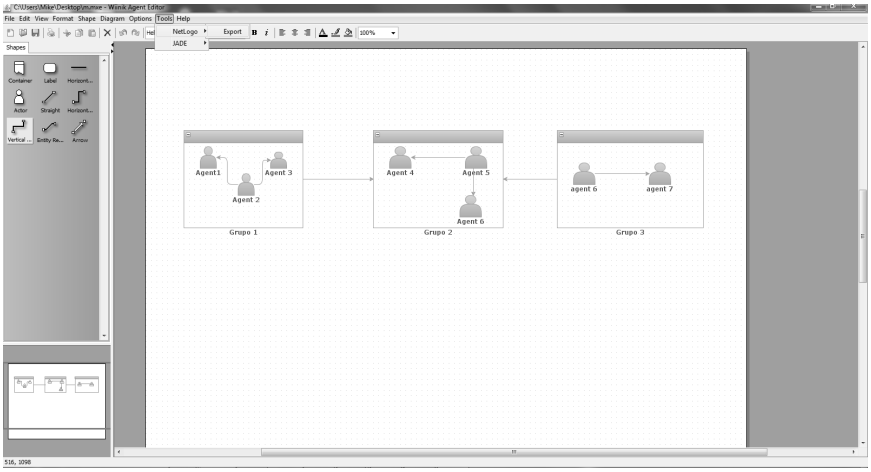

Figura 4.6: GUI para exportar a NetLogo

Capítulo 5

Pruebas y resultados

En este capítulo se describe en detalle los casos de estudio considerados en este proyecto. También se describen los resultados obtenidos en los casos de prueba y el resultado del análisis comparativo con otras herramientas similares.

Las pruebas que se llevaron a cabo fueron pruebas de rendimiento principalmente. Debido a que JT2FIS Framework es la biblioteca base, se aplicó la mayoría del esfuerzo en demostrar que es implementable en aplicaciones de software donde es importante la ligereza y el rendimiento. En simulación es importante aspirar a modelos con cantidades grandes de entidades capaces de procesar mucha información o ejecutar muchos procesos, entonces se vuelve clave contar con la confianza de que las bibliotecas pueden ofrecer el rendimiento necesario sin sacrificar demasiado la precisión.

Al desarrollar una extensión para NetLogo, se incorporó el uso de estos sistemas en ambientes de simulación existentes. Para este caso, sabiendo que NetLogo no es una herramienta que sobresalga por su rendimiento, el esfuerzo se enfocó en la facilidad para extender su funcionalidad a sistemas de inferencia difusos de manera clara para el usuario. Se desarrolló un ejemplo que fuera fácil de seguir por los usuarios adeptos a esta herramienta basado en un

caso clásico que viene por defecto en los ejemplos de origen de la herramienta.

5.1. Caso de estudio 1: Comparando resultados y rendimiento de JT2FIS con Matlab® Interval Type-2 Fuzzy Toolbox usando el caso de estudio de control de temperatura y flujo del agua.

Para probar la biblioteca JT2FIS , se usó el caso de estudio del control de temperatura y flujo del agua llamado "ISHOWER". Este ejemplo es un problema propuesto proporcionado por Matlab® como ejemplo para mostrar como utilizar la herramienta de lógica difusa. Castro et. al. [52, 53] extiende esta herramienta para lógica difusa de tipo 2 ("Matlab Interval Type-2 Fuzzy Toolbox"), así que se utilizó esta poderosa herramienta para compararlo contra la biblioteca JT2FIS.

En la Tabla 5.1 se puede observar las entradas y salidas que se usan en el ejemplo "ISHOWER", así como el tipo de función de membresía y los parámetros para cada una de ellas.Configuramos del mismo modo el sistema de inferencia difuso tipo 2 usando JT2FIS y la herramienta "Matlab Interval Type-2 Fuzzy Toolbox".

El sistema implementa las reglas que se muestran en la Tabla 5.2.

5.1.1. Resultados del caso de estudio ISHOWER

Con la configuración anterior del sistema de inferencia, usando 101 puntos para discretizar y el tipo de reducción centroide, evaluamos el control de temperatura y flujo del agua implementando JT2FIS y "Matlab Interval Type-2 Fuzzy Toolbox". En la Tabla 5.3 se muestra que

Tabla 5.1: Entradas y Salidas del ejemplo ISHOWER

Entrada/Salida	Función de Membresía	Parámetros
Temp (Cold)	TrapaUncertaintyMemberFunction	a1=-31,b1=-31,c1=-16,d1=-1, a2=-29,b2=-29,c2=-14,d2=1.0,alfa=0.98
Temp (Good)	TriUncertaintyMemberFunction	a1=-11,b1=-1,c1=9,a2=-9,b2=1,c2=11
Temp (Hot)	TrapaUncertaintyMemberFunction	a1=-1,b1=14,c1=29,d1=29, a2=1,b2=16,c2=31,d2=31,alfa=0.98
Flow (Soft)	TrapaUncertaintyMemberFunction	a1=-3.1,b1=-3.1,c1=-0.9,d1=-0.1, a2=-2.9,b2=-2.9,c2=-0.7,d2=0.1,alfa=0.98
Flow (Good)	TriUncertaintyMemberFunction	a1=-0.45,b1=-0.05,c1=0.35,a2=-0.35,b2=0.05,c2=0.45
Flow (Hard)	TrapaUncertaintyMemberFunction	a1=-0.1,b1=0.7,c1=2.9,d1=0.1, a2=0.9,b2=3.1,c2=3.1,d2=0.1,alfa=0.98
Cold (OpenSlow)	TriUncertaintyMemberFunction	a1=-1.05,b1=-0.65,c1=-0.35,a2=-0.95,b2=-0.55,c2=-0.25
Cold (CloseSlow)	TriUncertaintyMemberFunction	a1=-0.65,b1=-0.35,c1=-0.05,a2=-0.55,b2=-0.25,c2=0.05
Cold (CloseFast)	TriUncertaintyMemberFunction	a1=-0.35,b1=-0.05,c1=0.25,a2=-0.25,b2=0.05,c2=0.35
Cold (OpenFast)	TriUncertaintyMemberFunction	a1=-0.05,b1=0.25,c1=0.55,a2=0.05,b2=0.35,c2=0.65
Cold (Steady)	TriUncertaintyMemberFunction	a1=0.25,b1=0.55,c1=0.95,a2=0.35,b2=0.65,c2=1.05
Hot (OpenSlow)	TriUncertaintyMemberFunction	a1=-1.05,b1=-0.65,c1=-0.35,a2=-0.95,b2=-0.55,c2=-0.25
Hot (CloseSlow)	TriUncertaintyMemberFunction	a1=-0.65,b1=-0.35,c1=-0.05,a2=-0.55,b2=-0.25,c2=0.05
Hot (CloseFast)	TriUncertaintyMemberFunction	a1=-0.35,b1=-0.05,c1=0.25,a2=-0.25,b2=0.05,c2=0.35
Hot (OpenFast)	TriUncertaintyMemberFunction	a1=-0.05,b1=0.25,c1=0.55,a2=0.05,b2=0.35,c2=0.65
Hot (Steady)	TriUncertaintyMemberFunction	a1=0.25,b1=0.55,c1=0.95,a2=0.35,b2=0.65,c2=1.05

no hay diferencias entre las herramientas, en las cuales se obtuvieron las mismas respuestas.

Evaluamos el rendimiento incrementando el número de puntos de discretizacion en orden de incremento de complejidad de procesamiento de entradas y salidas(fuzificación y defuzificación). En la Tabla 5.4 se muestra los resultados donde podemos observar que JT2FIS es mas rapido que Matlab® Interval Type-2 Fuzzy Logic Toolbox en estas condiciones.

Tabla 5.2: Reglas del ejemplo ISHOWER

Antecedent (IF)	Consequent (THEN)
(Temp is Cold) and (Flow is Soft)	(Cold is OpenSlow)(Hot is OpenFast)
(Temp is Cold) and (Flow is Good)	(Cold is CloseSlow)(Hot is OpenSlow)
(Temp is Cold) and (Flow is Hard)	(Cold is CloseFast)(Hot is CloseSlow)
(Temp is Good) and (Flow is Soft)	(Cold is OpenSlow)(Hot is OpenSlow)
(Temp is Good) and (Flow is Good)	(Cold is Steady)(Hot is Steady)
(Temp is Good) and (Flow is Hard)	(Cold is CloseSlow)(Hot is CloseSlow)
(Temp is Hot) and (Flow is Soft)	(Cold is OpenFast)(Hot is OpenSlow)
(Temp is Hot) and (Flow is Good)	(Cold is OpenSlow)(Hot is CloseSlow)
(Temp is Hot) and (Flow is Hard)	(Cold is CloseSlow)(Hot is CloseFast)

5.2. Caso de Estudio 2: Comparando Resultados y Rendimiento de JT2FIS con Matlab® Interval Type-2 Fuzzy Toolbox y Juzzy Toolkit usando una variación del Caso de Estudio de Control de Temperatura y Flujo del Agua.

Wagner[56] introdujo Juzzy, una herramienta para sistemas de inferencia difusa para tipo 2 para el lenguaje de programación Java. Camparamos JT2FIS con la herramienta Juzzy, para esto usamos una variación del caso de estudio de control de temperatura y flujo de agua. Se uso esta variación para comparar también Matlab® Interval Type-2 Fuzzy Toolbox.

Para el caso de estudio 2, implementamos la misma configuración para un sistema de inferencia difusa tipo 2 en JT2FIS, Matlab® Interval Type-2 Fuzzy Toolbox y Juzzy Toolkit, usamos los parámetros que se muestran en la Tabla 5.5.

La herramienta Juzzy no tiene disponible funciones de membresía de tipo trapezoidales para sistemas de inferencia difusa de tipo 2, usamos una nueva configuración utilizando

Tabla 5.3: Comparando las salidas de JT2FIS con las salidas de Matlab® Interval Type-2 Fuzzy Logic Toolbox .

Inputs		Outputs JT2FIS			Outputs Interval Type-2 Fuzzy Logic Toolbox		
Temp	Flow	Cold	Hot	Tiempo(ms)	Cold	Hot	Tiempo(ms)
-16	-0.8	0.3	0.6328	1.24	0.3	0.6328	8.6
-12	-0.6	0.3	0.6342	1.24	0.3	0.6342	8.6
-8	-0.4	0.2211	0.5184	1.24	0.2211	0.5184	8.6
-4	-0.2	-0.0098	0.2545	1.24	-0.0098	0.2545	8.6
0	0	0	0	1.24	0	0	8.6
4	0.2	0.0098	-0.2545	1.24	0.0098	-0.2545	8.6
8	0.4	-0.22113	-0.5184	1.24	-0.22113	-0.5184	8.6
12	0.6	-0.2999	-0.6342	1.24	-0.2999	-0.6342	8.6
16	0.8	-0.3	-0.6328	1.24	-0.3	-0.6328	8.6
20	1	-0.3	-0.6328	1.24	-0.3	-0.6328	8.6

Tabla 5.4: Comparando el tiempo(ms) de JT2FIS con el tiempo(ms) de Matlab® Interval Type-2 Fuzzy Logic Toolbox con diferentes números de puntos de discretización para Input1=20 e Input2=1.

Número de Puntos	JT2FIS Tiempo(ms)	Interval Type-2 Fuzzy Logic Toolbox Tiempo(ms)
101	1.24	8.6
1001	4.96	13.7
10001	45.42	71.2

funciones de membresia de tipo “Triangular con Incertidumbre”.

Utilizamos 101 puntos y el tipo de reducción centroide para evaluar el sistema implementado con el mismo conjunto de datos para las entradas. La Tabla 5.6 muestra el resultado donde podemos observar que no existe diferencia ente las herramientas JT2FIS y Matlab® Interval Type-2 Fuzzy Toolbox, obteniendo el mismo resultado, pero se puede notar pequeñas diferencias entre JT2FIS y la herramienta Juzzy en estas condiciones.

Evaluamos el rendimiento incrementando el número de puntos de discretización incrementando la complejidad del proceso de entradas y salidas (fusificación y defusificación). La

Tabla 5.5: Entradas y salidas del ejemplo ISHOWER con funciones de membresía triangulares.

Entradas/Salidas	Función de Membresia	Parámetros
Temp (Cold)	TriUncertaintyMemberFunctionn	a1=-20.0,b1=-20.0,c1=1.06,a2=-20.0,b2=-20.0,c2=-1.0
Temp (Good)	TriUncertaintyMemberFunction	a1=-11.0,b1=0.0,c1=11.0,a2=-9.0,b2=0.0,c2=9.0
Temp (Hot)	TriUncertaintyMemberFunction	a1=-1.0,b1=20.0,c1=20.0,a2=1.0,b2=20.0,c2=20.0
Flow (Soft)	TriUncertaintyMemberFunction	a1=-1.0,b1=-1.0,c1=0.1,a2=-1.0,b2=-1.0,c2=-0.1
Flow (Good)	TriUncertaintyMemberFunction	a1=-0.5,b1=-0.0,c1=0.5,a2=-0.3,b2=0.0,c2=0.3
Flow (Hard)	TriUncertaintyMemberFunction	a1=-0.1,b1=1.0,c1=1.0,a2=0.1,b2=1.0,c2=1.0
Cold (OpenSlow)	TriUncertaintyMemberFunction	a1=-1.05,b1=-0.6,c1=-0.25,a2=-0.95,b2=-0.6,c2=-0.35
Cold (CloseSlow)	TriUncertaintyMemberFunction	a1=-0.65,b1=-0.3,c1=0.05,a2=-0.55,b2=-0.3,c2=-0.05
Cold (CloseFast)	TriUncertaintyMemberFunction	a1=-0.35,b1=0.0,c1=0.35,a2=-0.25,b2=0.0,c2=0.25
Cold (OpenFast)	TriUncertaintyMemberFunction	a1=-0.05,b1=0.3,c1=0.65,a2=0.05,b2=0.3,c2=0.55
Cold (Steady)	TriUncertaintyMemberFunction	a1=0.25,b1=0.6,c1=1.05,a2=0.35,b2=0.6,c2=0.95
Hot (OpenSlow)	TriUncertaintyMemberFunction	a1=-1.05,b1=-0.6,c1=-0.25,a2=-0.95,b2=-0.6,c2=-0.35
Hot (CloseSlow)	TriUncertaintyMemberFunction	a1=-0.65,b1=-0.35,c1=-0.05,a2=-0.55,b2=-0.25,c2=0.05
Hot (CloseFast)	TriUncertaintyMemberFunction	a1=-0.35,b1=0.0,c1=0.35,a2=-0.25,b2=0.0,c2=0.25
Hot (OpenFast)	TriUncertaintyMemberFunction	a1=-0.05,b1=0.3,c1=0.65,a2=0.05,b2=0.3,c2=0.55
Hot (Steady)	TriUncertaintyMemberFunction	a1=0.25,b1=0.6,c1=1.05,a2=0.35,b2=0.6,c2=0.95

Tabla 5.7 muestra los resultados donde se puede observar que la herramienta Juzzy es mas rápida que JT2FIS y Matlab® Interval Type-2 Fuzzy Logic Toolbox en estas condiciones.

5.3. Caso de Estudio 3: Comparando Rendimiento de JT2FIS con Matlab® Interval Type-2 Fuzzy Toolbox y la herramienta Juzzy usando diferente conjunto de reglas.

Como alternativa, para comparar el rendimiento entre JT2FIS,Matlab® Interval Type-2 Fuzzy Toolbox y la herramienta Juzzy, usamos un conjunto de reglas para configurar y probar los sistemas de inferencia, incrementado el número de reglas en cada prueba en orden de incremento de complejidad del procesamiento de las reglas.

Para el caso de estudio 3, configuramos un sistema de inferencia difuza tipo 2 en JT2FIS,

Tabla 5.6: Comparando las salidas de JT2FIS contra Matlab® Interval Type-2 Fuzzy Logic Toolbox y la herramienta Juzzy.

Entradas		Salidas JT2FIS			Salidas Interval Type-2 Fuzzy Logic Toolbox			Salidas Juzzy		
Temp	Flow	Cold	Hot	Tiempo(ms)	Cold	Hot	Tiempo(ms)	Cold	Hot	Tiempo(ms)
-16	-0.8	0.3	0.6336	1.24	0.3	0.6336	8.6	0.2999	0.6346	0.67
-12	-0.6	0.3	0.6363	1.24	0.3	0.6363	8.6	0.2999	0.6374	0.67
-8	-0.4	0.1822	0.4890	1.24	0.1822	0.4890	8.6	0.1824	0.4905	0.67
-4	-0.2	-0.0052	0.2337	1.24	-0.0052	0.2337	8.6	-0.0053	0.2356	0.67
0	0	-3.4694E-17	-3.4694E-17	1.24	-3.4694E-17	-3.4694E-17	8.6	-5.5511E-17	-5.5511E-17	0.67
4	0.2	0.0052	-0.2337	1.24	0.0052	-0.2337	8.6	0.0053	-0.2356	0.67
8	0.4	-0.1822	-0.4890	1.24	-0.1822	-0.4890	8.6	-0.1824	-0.4905	0.67
12	0.6	-0.2999	-0.6363	1.24	-0.2999	-0.6363	8.6	-0.2999	-0.6374	0.67
16	0.8	-0.3	-0.6336	1.24	-0.3	-0.6336	8.6	-0.2999	-0.6346	0.67
20	1	-0.3	-0.6324	1.24	-0.3	-0.6324	8.6	-0.2999	-0.6334	0.67

Tabla 5.7: Comparando el tiempo(ms) de JT2FIS contra el tiempo(ms) de Matlab® Interval Type-2 Fuzzy Logic Toolbox y el tiempo(ms) de la herramienta Juzzy Toolkit con diferentes puntos de discretizacion para para input1=20 and input2=1.

Número de Puntos	JT2FIS Tiempo(ms)	Interval Type-2 Fuzzy Logic Toolbox Tiempo(ms)	Juzzy Toolkit Tiempo(ms)
101	1.24	8.6	0.67
1001	4.96	13.7	1.23
10001	45.42	71.2	8.48

Matlab® Interval Type-2 Fuzzy Toolbox y la herramienta Juzzy usando los mismos parámetros que se muestran en la Tabla 5.8. Agregamos una entrada en cada incremento de conjunto de reglas y configuramos las reglas relativamente para cada ronda.

Usamos 101 puntos y el tipo de reducción centroide, evaluamos todos los sistemas implementados con los diferentes conjuntos de reglas. La Tabla 5.9 muestra los resultados donde se puede observar las diferencias entre JT2FIS,Matlab® Interval Type-2 Fuzzy Toolbox y la herramienta Juzzy. JT2FIS mostró tener mejor rendimiento cuando el conjunto de reglas se fue incrementando.

Tabla 5.8: Configuración de entradas y salidas para el conjunto de reglas del caso de estudio.

Entadas/Salidas	Función de Membresia	Parámetros
Entrada (MF1)	GaussUncertaintyStandardDesviationMemberFunction	desviación 1=0.1623, desviación 2=0.2623 media=0.0
Entrada (MF2)	GaussUncertaintyStandardDesviationMemberFunction	desviación 1=0.1623, desviación 2=0.2623 media=0.5
Entrada (MF3)	GaussUncertaintyStandardDesviationMemberFunction	desviación 1=0.1623, desviación 2=0.2623 media=1.0
Salida (MF1)	TriUncertaintyMemberFunction	a1=-0.5833,b1=-0.08333,c1=0.4167,a2=-0.4167,b2=-0.08333,c2=0.5833
Salida (MF2)	TriUncertaintyMemberFunction	a1=-0.08333,b1=0.4167,c1=0.9167,a2=0.08333,b2=0.4167,c2=1.083
Salida (MF3)	TriUncertaintyMemberFunction	a1=0.4167,b1=0.9167,c1=1.417,a2=0.5833,b2=0.9167,c2=1.583

Tabla 5.9: Comparando tiempo(ms)de JT2FIS contra el tiempo(ms) de Matlab® Interval Type-2 Fuzzy Logic Toolbox y el tiempo(ms) de la herramienta Juzzy con diferente número de reglas para input1=20 and input2=1.

Número de Entradas	Número de Reglas	JT2FIS Tiempo(ms)	Interval Type-2 Fuzzy Logic Toolbox Tiempo(ms)	Juzzy Toolkit Tiempo(ms)
2	9	0.72	4	0.74
3	27	1.09	5.5	2.03
4	81	1.95	10	10.77
5	243	5.807	30	148.19
6	729	16.44	70	3621.2

5.4. Caso de Estudio 4: Comparando Resultados y Rendimiento de JT2FISClustering con genfis3 Matlab® Usando Diferentes Conjuntos de Datos Aleatorios.

Para probar la biblioteca JT2FISClustering, usamos un conjunto de datos creados aleatoria mente a través del método de generación de números pseudoaleatorios Mersenne Twister[75] para compararlo contra el método genfis3 proporcionado por Matlab®, este último método genera sistemas de inferencia difusa tipo 1, por lo que ajustamos a JT2FISClustering con incertidumbre cero.

Para este caso de estudio utilizamos 2 entradas, 2 salidas y 3 reglas para 500,5000,50000 y 500000 datos, para este ejemplo escogimos la función de membresía gaussiana con incer-

tidumbre en la media("GaussUncertaintyMeanMemberFunction") para todas las entradas y salidas; en la Tabla 5.10 se muestran los tiempos que tardaron cada uno de los métodos en generar el sistema de inferencia, se puede observar que la biblioteca JT2FISClustering es mas rápida que el método genfis3 proporcionado por Matlab®. En los resultados no se observa ninguna diferencia a cerca de la configuración generada como se puede observar en la Tabla 5.11.

Tabla 5.10: Comparando tiempo(ms)de JT2FISClustering contra el tiempo(ms) de genfis3 Matlab®.

Número de Datos	**JT2FISClustering Tiempo(ms)**	**genfis3 Tiempo(ms)**
500	9.62	16.18
5000	94.86	172.81
50000	197.21	277.84
500000	402.98	666.93

5.5. Caso de Estudio 5: Comparando Rendimiento de JT2FISClustering con genfis3 Matlab® Usando Diferente Número de Reglas.

Como alternativa, para comparar el rendimiento entre JT2FISClustering y el método genfis3 proporcionado por Matlab®, usamos un conjunto de 500 datos para cada entrada y salida, creados aleatoriamente a través del método de generación de números pseudoaleatorios Mersenne Twister[75], al igual que en el caso de estudio 4 ajustamos a JT2FISClustering con incertidumbre cero.

Para este caso de estudio incrementamos el número de reglas en cada iteración, utilizamos 1 salida para todas las iteraciones y empezamos con 2 entradas, en cada iteración agregamos

una entrada en cada incremento de conjunto de reglas. En este caso de estudio escogimos la función de mebresía gaussina con incertidumbre en la media("GaussUncertaintyMeanMemberFunction") para todas las entradas y salidas; en la Tabla 5.12 se muestran los tiempos que tardaron cada unos de los métodos en generar el sistema de inferencia, se puede observar que la biblioteca JT2FISClustering es mas rápida que el método genfis3 en cada iteración.

5.6. Caso de Estudio 6: Extendiendo NetLogo usando JT2FISNetLogo.

Hemos ampliado la funcionalidad de NetLogo, introduciendo un sistema de inferencia difuso tipo 2. Para probar la extensión JT2FISNetLogo para NetLogo, se usó como caso de estudio del modelo de votación("Voting") que viene propuesto como modelo de ejemplo en NetLogo[76]. Este modelo, es un autómata celular que simula la distribución de votos haciendo que cada parche(patch) tome un "voto"de sus ocho vecinos, este puede cambiar su voto de acuerdo a los resultados de sus vecinos.

Siguiendo las sugerencias de los autores del modelo original de ejemplo, para este caso de uso se extendió el modelo agregando una regla que permita cambiar el resultados de los votos incorporando a las preferencias de cada autómata (representado por un cuadro -patch- en el área de simulación).

El sistemas de preferencias de los autómatas en su conjunto es descrito por un sistema de inferencia difuso tipo 2 por intervalos que es utilizado por en el modelo como un sistema de toma de decisiones.

La configuración inicial del sistema de inferencia difuso tipo 2 se pueden observar en la Tabla 5.13.

El sistema implementa las siguientes reglas:

1. If (BlueVotes is LowBlueVotes) and (GreenVotes is HighGreenVotes) then (BlueVotePreference is LowBluePreference)

2. If (BlueVotes is HighBlueVotes) and (GreenVotes is LowGreenVotes) then (BlueVotePreference is HighBluePreference)

En este caso de estudio, hemos agregado la posibilidad de cambiar los parámetros de las funciones de membresía con las herramientas visuales proporcionadas por NetLogo (vea la figura 5.1).

Al ejecutar la simulación en el modo en donde se considera las preferencias de los autómatas, se utiliza el sistema de inferencia difuso para evaluar cada caso en particular (cada autómata en particular) y decidir de acuerdo a las reglas descritas la decisión del autómata. A diferencia del modelo original, donde los autómatas ejecutan reglas lineales sencillas basada en sus 8 vecinos, en la nueva aproximación se incorpora la percepción del autómata respecto a lo que está sucediendo no solo alrededor de él, sino en todo el universo de agentes para decidir su voto.

La percepción está representada por los conjuntos difusos del sistema, que sirven para convertir los valores concretos (crisp values) en valores difusos. El sistema difuso evalúa valores concretos de entrada, fuzzificando, aplicando las reglas y defuzzificando para obtener como resultado un valor que representa una toma de decisión (de facto).

El modelo se vuelve entonces flexible para representar diferentes poblaciones de autómatas y su sistema de toma de decisiones (o sea, autómatas con diferentes configuraciones en su sistema de toma de decisiones). Aunque el sistema aplica las mismas reglas en el sistema de toma de decisiones, la interpretación de las entradas pueden variar dependiendo de la

configuración de las funciones de membresía de entradas y salidas, a lo que podríamos decir que puede variar, debido a que los autómatas podrían percibir de distinta manera su entorno.

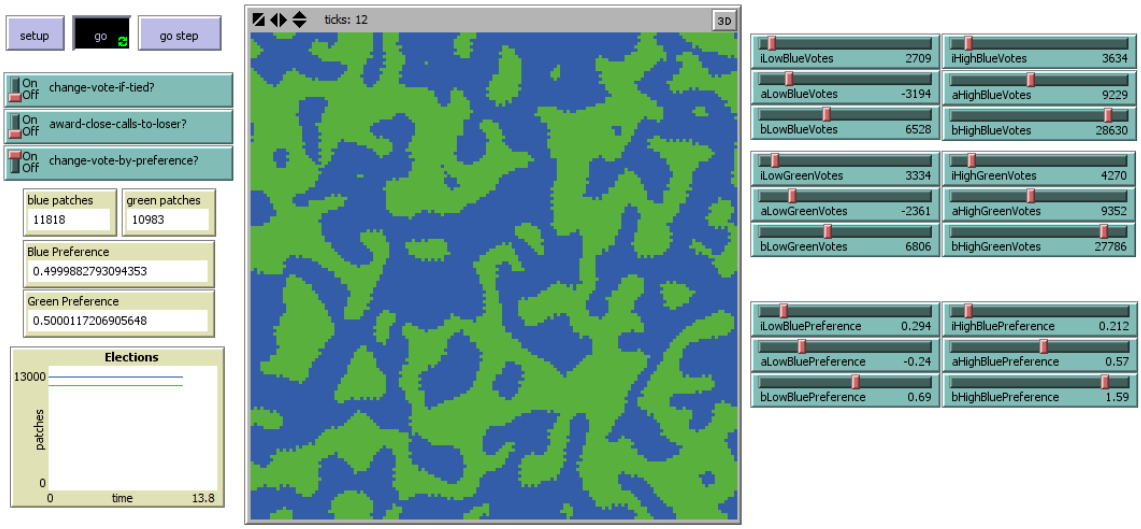

Figura 5.1: Simulación del modelo de votación en NetLogo.

En la Figura 5.1 se pueden observar los resultados y la interfaz gráfica de usuario utilizada para este ejemplo. Con este pequeño caso ilustrativo del modelo de votación de NetLogo se valida la operatividad de la propuesta de ampliación de NetLogo a través de la extención JT2FISNetLogo.

Tabla 5.11: Comparando resultados de JT2FISClustering contra genfis3 de Matlab® para 3 entradas y 3 salidas.

Número de Datos	Entrada/Salida	**JT2FISClustering** Parámetros	**genfis3** Parámetros
500	Entrada Uno(in1)	sigma=0.1757 media 1,2=0.6495	sigma=0.1757 media=0.6495
		sigma=0.1701 media 1,2=0.4174	sigma=0.1701 media=0.4174
		sigma=0.1748 media 1,2=0.4187	sigma=0.1748 media=0.4187
	Entrada Dos(in2)	sigma=0.16 media 1,2=0.5118	sigma=0.16 media=0.5118
		sigma=0.1896 media 1,2=0.2218	sigma=0.1896 media=0.2218
		sigma=0.1892 media 1,2=0.791	sigma=0.1892 media=0.791
	Salida Uno(out1)	sigma=0.16 media 1,2=0.5118	sigma=0.16 media=0.5118
		sigma=0.1896 media 1,2=0.2218	sigma=0.1896 media=0.2218
		sigma=0.1892 media 1,2=0.791	sigma=0.1892 media=0.791
	Salida Dos(out2)	sigma=0.1825 media 1,2=0.3553	sigma=0.1825 media=0.3553
		sigma=0.1793 media 1,2=0.5945	sigma=0.1793 media=0.5945
		sigma=0.1784 media 1,2=0.5882	sigma=0.1784 media=0.5882
5000	Entrada Uno(in1)	sigma=0.1686 media 1,2=0.5001	sigma=0.1686 media=0.5001
		sigma=0.1688 media 1,2=0.5278	sigma=0.1688 media=0.5278
		sigma=0.1688 media 1,2=0.4904	sigma=0.1688 media=0.4904
	Entrada Dos(in2)	sigma=0.1689 media 1,2=0.4981	sigma=0.1689 media=0.4981
		sigma=0.1695 media 1,2=0.4704	sigma=0.1695 media=0.4704
		sigma=0.1693 media 1,2=0.541	sigma=0.1693 media=0.541
	Salida Uno(out1)	sigma=0.1683 media 1,2=0.5067	sigma=0.1683 media=0.5067
		sigma=0.1684 media 1,2=0.5146	sigma=0.1684 media=0.5146
		sigma=0.1686 media 1,2=0.4635	sigma=0.1686 media=0.4635
	Salida Dos(out2)	sigma=0.1688 media 1,2=0.5178	sigma=0.1688 media=0.5178
		sigma=0.1689 media 1,2=0.5237	sigma= 0.1689 media=0.5237
		sigma=0.1695 media 1,2=0.4549	sigma=0.1695 media=0.4549
5000	Entrada Uno(in1)	sigma=0.1688 media 1,2=0.5003	sigma=0.1688 media=0.5003
		sigma=0.1688 media 1,2=0.4989	sigma=0.1688 media=0.4989
		sigma=0.1688 media 1,2=0.501	sigma=0.1688 media=0.501
	Entrada Dos(in2)	sigma=0.1682 media 1,2=0.5017	sigma=0.1682 media=0.5017
		sigma=0.1682 media 1,2=0.499	sigma=0.1682 media=0.499
		sigma=0.1682 media 1,2=0.5009	sigma=0.1682 media=0.5009
	Salida Uno(out1)	sigma=0.1689 media 1,2=0.4985	sigma=0.1689 media=0.4985
		sigma=0.1689 media 1,2=0.4991	sigma=0.1689 media=0.4991
		sigma=0.1689 media 1,2=0.4991	sigma=0.1689 media=0.4991
	Salida Dos(out2)	sigma=0.1685 media 1,2=0.4992	sigma=0.1685 media=0.4992
		sigma=0.1685 media 1,2=0.5005	sigma=0.1685 media=0.5005
		sigma=0.1685 media 1,2=0.5008	sigma=0.1685 media=0.5008
5000	Entrada Uno(in1)	sigma=0.1689 media 1,2=0.5006	sigma=0.1689 media=0.5006
		sigma=0.1689 media 1,2=0.5006	sigma=0.1689 media=0.5006
		sigma=0.1689 media 1,2=0.4997	sigma=0.1689 media=0.4997
	Entrada Dos(in2)	sigma=0.1687 media 1,2=0.4993	sigma=0.1687 media=0.4993
		sigma=0.1687 media 1,2=0.4992	sigma=0.1687 media=0.4992
		sigma=0.1687 media 1,2=0.5	sigma=0.1687 media=0.5
	Salida Uno(out1)	sigma=0.1688 media 1,2=0.5003	sigma=0.1688 media=0.5003
		sigma=0.1688 media 1,2=0.5001	sigma=0.1688 media=0.5001
		sigma=0.1688 media 1,2=0.5002	sigma=0.1688 media=0.5002
	Salida Dos(out2)	sigma=0.1683 media 1,2=0.4993	sigma=0.1683 media=0.4993
		sigma=0.1683 media 1,2=0.4998	sigma=0.1683 media=0.4998
		sigma=0.1683 media 1,2=0.501	sigma=0.1683 media=0.501

Tabla 5.12: Comparando tiempo(ms)de JT2FISClustering contra el tiempo(ms) de genfis3 Matlab® con diferente número de reglas.

Número de Entradas	Número de Reglas	JT2FISClustering Tiempo(ms)	genfis3 Tiempo(ms)
2	9	19.2	31.56
3	27	88.64	165.92
4	81	179.88	385.92
5	243	510.17	1179.7
6	729	1155.31	3305.1

Tabla 5.13: Entradas y Salidas del ejemplo "Voting".

Entrada/Salida	Función de Membresía	Parámetros
BlueVotes(LowBlueVotes) Entrada Uno	gaussUncertaintyMeanMemberFunction	sigma=2709, media 1=-3194, media 2=6528
BlueVotes (HighBlueVotes) Entrada Uno	gaussUncertaintyMeanMemberFunction	sigma=3634, media 1=9229, media 2=28630
GreenVotes(LowGreenVotes) Entrada Dos	gaussUncertaintyMeanMemberFunction	sigma=3334, media 1=-2361, media 2=6806
GreenVotes (HighGreenVotes) Entrada Dos	gaussUncertaintyMeanMemberFunction	sigma=4270, media 1=9352, media 2=27786
BlueVotePreference(LowBluePreference) Salida Uno	gaussUncertaintyMeanMemberFunction	sigma=0.294, media 1=-0.24, media 2=0.69
BlueVotePreference(HighBluePreference) Salida Uno	gaussUncertaintyMeanMemberFunction	sigma=0.212, media 1=0.57 , media 2=1.59

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este documento se ha presentado un conjunto de herramientas, cuya finalidad es ayudar a crear modelos basados en agentes de sistemas complejos utilizando agentes cognitivos y sistemas multi-agente.

JT2FIS es una biblioteca de clases orientada a objetos que se puede utilizar para crear aplicaciones inteligentes basadas en sistemas de inferencia difusos tipo 2. Se presentó una arquitectura con un diseño orientado a objetos, que ayuda a representar estos sistemas de una forma mas sencilla que otras herramientas como MatLab® Interval Type-2 Fuzzy Toolbox y la herramienta Juzzy, y que le permite al programador crear sistemas de inferencia de una manera sencilla y poca compleja, teniendo conocimientos básicos sobre los conceptos de sistemas de inferencia difusa.

JT2FIS proporciona un conjunto de clases que se pueden utilizar para construir sistemas inteligentes basados en sistemas difusos tipo 2. Actualmente hay algunas bibliotecas de lógica difusa disponibles en Java, pero la mayoría de ellos están restringidos para sistemas difusos

tipo 1. Esta biblioteca de clases se ha propuesto para cubrir la necesitad de una biblioteca de sistemas difusos tipo 2 en Java. Tradicionalmente en diferentes investigaciones se usa la herramienta MatLab® Interval Type-2 Fuzzy Toolbox, pero ha sido difícil transferir algunos diseños e implementaciones de MatLab® ha aplicaciones comerciales. Por otra parte, Java es un lenguaje de programación muy utilizado por los académicos y profesionales, y está completamente aceptado por la industria de software por la conveniencia de la re utilización, la legibilidad de la portabilidad y la codificación del sistema.

La biblioteca de clases propuesta está completamente centrada en el diseño orientado a objetos y en la usabilidad de la programación en Java. Esta permite crear sistemas de inferencia difusa construida como una estructura de objetos y su diseño permite ampliar sus funcionalidades y características. Los diferentes objetos que componen la estructura pueden utilizarse por separado, sin el sistemas de inferencia completo. JT2FIS no depende de bibliotecas de terceros por lo que se puede utilizar e incrustar en diferentes tipos de aplicaciones Java, incluyendo aplicaciones distribuidas.

De acuerdo a los casos de estudio, realizados con JT2FIS y las comparaciones hechas con las herramientas MatLab® Interval Type-2 Fuzzy Toolbox y Juzzy toolkit, es esta última posiblemente el más cercano a JT2FIS. Juzzy es un conjunto de herramientas Java para la creación de sistemas de inferencia difusa tipo 2 y tiene la ventaja de que puede construir sistemas difusos generalizados, pero JT2FIS fue inspirado en la herramienta MatLab® Interval Type-2 Fuzzy Toolbox y hasta este momento se demostró la consistencia de los resultados. Otra diferencia importante es que Juzzy viene con algunas herramientas útiles integradas para configurar el sistema y graficar los resultados, por otro lado, JT2FIS es una biblioteca de clases de Java puro que implementa sólo la funcionalidad de sistemas de inferencia difusa de tipo 2, por lo que las nuevas aplicaciones se pueden desarrollar desde cero utilizando como código núcleo JT2FIS incluyendo componentes visuales de Java o las herramientas para graficar resultados.

A partir de JT2FIS se creó JT2FISPanel un componente visual de Java para la creación de aplicaciones Java inteligentes que utilizan sistemas de inferencia difuza tipo 2. El diseño de JT2FISPanel es orientado a objetos y esta basado en el componente visual JPanel de Swing de Java. JT2FISPanel permite una fácil configuración de un sistema de inferencia, que permite agregar entradas, salidas y reglas que conforman un FIS, de una forma visual, sencilla y dinámica. Esta muestra un árbol donde se muestra todos los objetos que integran un FIS, para tener un mejor control y edición del mismo, de una forma mas sencilla y natural para las personas que lo deseen utilizar.

Actualmente no existen editores que permitan configurar sistemas de inferencia difusa tipo 2 en un lenguaje como Java, uno de los editores existentes para esto es proporcionada por la herramienta MatLab® Interval Type-2 Fuzzy Toolbox, pero esta hace complicada realizar aplicaciones comerciales con ellas o integrarlas a nuevos proyectos, ya que no está diseñada para incluirla en nuevos proyectos si no como una herramienta exclusiva de configuración de sistemas de inferencia, por lo que lo hace muy complicado tratar de hacer uso de ella en aplicaciones comerciales o que requieran el apoyo de esta para resolver un problema en particular.

JT2FISPanel por el contrario está diseñada como un componente que sea capaz de ser utilizado de una forma sencilla en cualquier proyecto que se desee, este componente extiende de la clase JPanel de Swing de Java, por lo que le da las características de poderlo usar dentro de componentes como JFrame o JDialog de la librería Swing de Java. Esto permite agregarlo en cualquier proyecto que se desee, ya que se utiliza como si fuera un simple JPanel de Java. En este documento se expuso y ejemplifico como poder utilizar JT2FISPanel dentro de un JFrame y se mostró lo fácil que es utilizarlo para configurar un sistema de inferencia difuso tipo 2.

En ocasiones existen problemas, en donde los investigadores no saben o no tienen idea de

qué parámetros poder utilizar o como configurar inicialmente las funciones de pertenencia de las entradas y salidas de un sistema de inferencia difuso, muchos investigadores optan por sacar simples cálculos como medias y desviaciones de los datos obtenidos en sus investigaciones de campo para configurar estas funciones de pertenencia, pero muchas veces no se obtienen los resultados deseados, ya que no tienen en mente que una función de pertenencia describe una agrupación de datos que conforma un problema, estas agrupaciones deben compartir ciertas características y para determinar la semejanza entre estos datos se requieren técnicas mucho más complejas que solo cálculos probabílisticos como lo pueden ser los métodos de minería de datos que permitan agrupar estos datos.

Con esto en mente, se optó por crear la biblioteca de clases JT2FISClustering, esta es una biblioteca de clases capaz de configurar un sistema de inferencia difusa de una forma rápida y sencilla teniendo un conjunto pertinente de datos. Para poder lograr esto, JT2FISClustering hace uso de técnicas de minado de datos, como el método Fuzzy c-means que es un método rápido, capaz de encontrar agrupaciones de datos con características semejantes y configurar rápidamente un FIS de tipo Mamdani con funciones de pertenencia de tipo gaussiana. Actualmente, dentro de los sistemas de inferencia difusos tipo 2, los métodos existentes de minería de datos aun no proporcionan una confianza del 100 % para poder decir que son configuraciones lo suficientemente correctas para resolver un problema, pero proporcionan un primer vistazo rápido y sencillo de como poder configurar nuestro FIS.

MatLab® proporciona un método para poder lograr esto llamado ‘genfis3”, uno de los problemas con este método es que se obtiene como resultado un sistema de inferencia difuso Mamdani de tipo 1, por lo que no expresa la incertidumbre ni los factores de riesgo existentes en los problemas reales que se pueden presentar. JT2FISClutering por el contrario ha sido modificado para poder obtener sistemas de inferencia difuso Mamdani de tipo 2. Además con los casos de estudios presentados en el capítulo 5 se puede observar, que ambas bajo las mismas condiciones obtienen los mismos resultados, pero que el rendimiento de

JT2FISClustering es mucho mejor en cuestión de velocidad, tanto en el cambio de número de datos a procesar, como en el cambio de número de reglas.

JT2FISClustering esta diseñado aplicando los conceptos del paradigma de programación orientada a objetos, por lo que proporciona mucha facilidad de agregar nuevas características a la misma, haciendo más sencillo para los programadores agregar nuevos métodos de agrupación de datos o nuevas funcionalidades, además al estar desarrollada en Java tiene la característica de portabilidad y al no depender de ninguna biblioteca de terceros, se puede usar de forma más simple en cualquier proyecto donde se desee hacer uso de ella.

Para facilitar al usuario el uso de la biblioteca de clases JT2FISClustering, se realizó un componente visual llamado JT2FISClusteringPanel para la creación de aplicaciones inteligentes en Java que utilizan sistemas de inferencia difusa tipo 2. Esta presenta un diseño orientado a objetos basado en JPanel de la biblioteca Swing de Java.

JT2FISClusteringPanel utiliza las bibliotecas de clases JT2FIS y JT2FISClustering que son el núcleo de este componente. JT2FISClusteringPanel permite trata los datos recolectados del problema de una forma sencilla, eficiente y eficaz. Este componente proporciona una alternativa para configurar un FIS de una forma sencilla y sin complicaciones. Esta utiliza un archivo CSV para importar los datos que se tienen del problema, estos son presentados en forma de tabla, donde se pueden visualizar los nombres de cada columna que conforma el conjunto de datos del problema, para poder ser elegidos como entrada, salida o ignorarlo para la configuración del FIS.

El objetivo de JT2FISClusteringPanel es tratar la información de la forma mas sencilla posible y sin complicaciones para los usuarios, basta con importar un archivo CSV, seleccionar que datos se quieren como entradas, salidas o cuales ignorar, para después seleccionar un método de agrupamiento y esto dará como resultado una configuración rápida de un FIS. Otra de las ventajas de JTFISClusteringPanel es que al estar basada en el componente JPanel

de la biblioteca Swing de Java, facilita incluir él componente en cualquier proyecto, ya sea en un JFrame, JDialog u otros componentes, ya que se comporta como un panel que puede ser incrustado en cualquier proyecto que se desee. En los casos de estudio ejemplificamos como puede usarse dentro de un JDialog.

Todas estas bibliotecas y componentes visuales fueron diseñadas con el objetivo de ayudar a crear modelos basados en agentes de sistemas complejos utilizando agentes cognitivos y sistemas multi-agente, pero para poder lograr esto se necesita de una herramienta que sea capaz de facilitarle a los usuario la creación de modelos y poder agregar todas las funcionalidades antes mencionadas. Para esto se desarrollo Wiinik.

Wíinik es una herramienta prototipo experimental que tiene como objetivo ayudar a los investigadores a diseñar estructuras de agentes en las simulaciones basadas en agentes. La herramienta está inspirada en el marco conceptual de las agencias distribuidas, lo cual permite expresar un problema en términos de niveles, donde cada nivel esta conformada por diferentes agentes capaces de comunicarse entre si dentro de su nivel, y fuera del mismo con otros agentes de nivel superior. Cada nivel puede ser visto como un agente de un nivel superior, que puede llegar afectar en el comportamiento de todos los agentes que pertenezcan a él.

Wíinik implementa las bibliotecas y componentes anteriormente expuestos, ya que lo que se propone es que los agentes puedan contener una base de conocimientos que se implementan con un sistema de inferencia difuso que permite la incorporación de la percepción y la incertidumbre en los modelos. Actualmente esta herramienta no es un entorno de simulación, pero puede ser utilizada para crear modelos preliminares que pueden implementarse en simulaciones, capaces de incluir en cada agente una base de conocimientos.

Wíinik ayuda a diseñar estructuras de agentes pero al no ser un entorno de simulación tenemos que hacer uso de otros entornos para poder crear simulaciones, nosotros proponemos

utilizar NetLogo ya que es una de las herramientas mas utilizadas en creación de modelos y permite agregar nuevas funcionalidades a través de extensiones, otra ventaja es que esta desarrollada completamente en Java.

NetLogo no tiene una funcionalidad de agregar a los agentes un sistema de inferencia difuso que les permita convertirse en agentes inteligentes, por lo que hacemos uso de la biblioteca de clases JT2FISNetLogo para poder agregar esta funcionalidad.

JT2FISNetLogo es una extensión que permite configurar sistemas de inferencia difusos dentro del ambiente NetLogo, está biblioteca permite una comunicación entre NetLogo y la biblioteca de clases JT2FIS, para poder utilizar agentes que tengan sistemas de inferencia difusa que les permitan utilizarlos para la toma de decisiones. JT2FISNetLogo tiene una arquitectura con un diseño orientada a objetos igual que JT2FIS que permite una mejor comunicación entre ambas librerías y facilita los cambios o agregar nuevas funcionalidades en ambas si es que son necesarios en un futuro.

6.2. Trabajo futuro

A continuación se presenta una conjunto de recomendaciones para continuar con este trabajo. Las herramientas desarrolladas pueden ser mejoradas en muchas direcciones, por lo que sugerimos una lista de mejoras y nuevas funcionalidades.

`JT2FIS`

- Incrementar los métodos de defusificación.
- Implementar tipo `no-singleton` para los sistemas de inferencia difuso.
- Implementar tipo `Sugeno` para los sistemas de inferencia difuso.

- Agregar sistemas de inferencia difuso tipo 2 generalizado.
- Importar modelos generados por la herramienta `Matlab® Interval Type-2 Fuzzy Toolbox`
- Exportar sistema de inferencia difuso utilizando `FML` (Fuzzy Markup Language).
- Migrar a diferentes lenguajes por ejemplo: C#, Python.
- Mejorar rendimiento.
- Refactorizar código.
- Hacer manual de usuario.

`JT2FISPanel`

- Realizar pruebas de usabilidad.
- Agregar nuevos modos de graficar los resultados obtenidos por el sistema de inferencia difuso (gráficas 3D o de superficie).
- Implementar las nuevas funcionalidades de JT2FIS.
- Mejorar interfaz gráfica de usuario.
- Hacer manual de usuario.

`JT2FISClustering`

- Agregar métodos de agrupamientos de datos.
- Refactorizar código.

- Mejorar rendimiento.
- Hacer manual de usuario

`JT2FISClusteringPanel`

- Realizar pruebas de usabilidad.
- Agregar conexiones directas a base de datos.
- Implementar las nuevas funcionalidades de JT2FISClustering.
- Mejorar interfaz gráfica de usuario.
- Hacer manual de usuario.

`JT2FISNetLogo`

- Incrementar casos de estudio para probar la extension.
- Implementar las nuevas funcionalidades de JT2FIS.
- Refactorizar código.
- Hacer manual de usuario

`Wiinik`

- Realizar pruebas de usabilidad.
- Mejorar la funcionalidad de exportar a NetLogo.
- Agregar la funcionalidad para exportar a Jade.

- Escalar la funcionalidad de una herramienta de diseño a un simulador
- Mejorar interfaz gráfica de usuario.
- Hacer manual de usuario.

Bibliografía

[1] Manuel Castañón-Puga, Antonio Rodríguez-Díaz, Guillermo Licea, and Eugenio Dante Suarez. Social systems simulation person modeling as systemic constructivist approach. *Soft Computing for Hybrid Intelligent Systems*, 2008.

[2] Yong-Kui Zhang Yea-Chung Ding. The simulation of urban growth applying sleuth ca model to the yilan delta in Taiwan. *Alam Bina*, 2007.

[3] Dora-Luz Flores, Manuel Castañón-Puga, and Carelia Gaxiola-Pacheco. A complex social system simulation using type-2 fuzzy logic and multiagent system. In *Simulation*. 2011.

[4] Bogart Yail Márquez, Manuel Castañón-Puga, Juan Ramón Castro, and Eugenio Dante Suarez. Methodology for the Modeling of Complex Social System Using Neuro-Fuzzy and Distributed Agencies. *cyberjournals.com*, 2011.

[5] Michele Piunti, Andrea Santi, and Alessandro Ricci. Programming soa/ws systems with bdi agents and artifact-based environments. *Of MALLOW federated Workshops on Agents*, 2009.

[6] Mauro Dragone, Howell Jordan, David Lillis, and Rem W. Collier. Separation of concerns in hybrid agent and component system. *Networks*, 2010.

[7] Claudio Cioffi-Revilla. A methodology for complex social simulations. *Journal of Artificial Societies and Social Simulation*, 13(1):7, 2010.

[8] Scott Moss. Alternative approaches to the empirical validation of agent-based models. *Journal of Artificial Societies and Social Simulation*, 11(1):5, 2008.

[9] David Hales, Juliette Rouchier, and Bruce Edmonds. Model-to-model analysis. *Journal of Artificial Societies and Social Simulation*, 6(4), 2003.

[10] Ph.D Claudio Cioffi-Revilla. On social complexity : A manifesto for computational social science. *The Journal of Artificial Societies and Social Simulation*, 2010.

[11] Robert Axelrod. Advancing the art of simulation in the social sciences. *Complexity*, 3(2):21–40, November 1997.

[12] Arturo Guillén. "the party is over", la crisis global y la recesión generalizada. *Economía, UNAM*, 6(016):23–43, 2009.

[13] Roberto I. Escalante-Semerena. Desarrollo rural, regional y medio ambiente. *Economía, UNAM*, 3(008):70–94, 2009.

[14] Mario Miguel Carrillo-Huerta. La teoría neoclásica de la convergencia y la realidad del desarrollo regional en méxico. *IIEc-UNAM*, 32, 2001.

[15] Norma Samaniego. La crisis, el empleo y los salarios en méxico. *Economía, UNAM*, 6(016):57–67, 2009.

[16] Robert G. Sargent. Verification and validation of simulation models. In *Simulation*, pages 37–48. 2003.

[17] Christoph Adami. What is complexity? *BioEssays*, 24(12):1085–1094, 2002.

[18] Francis Heylighen. Five questions on complexity. 2008.

[19] Warren Weaver. *Science And Complexity*, chapter 1' "The Scientists Speak,". 1948, New York City, 1948.

[20] Edgar Morin. Restricted complexity, general complexity. 2008.

[21] Julie Thompson Klein. Interdisciplinarity and complexity: An evolving relationship. volume 6, pages 2–10, México, 2004.

[22] Giulio Tononi and Gerald M. Edelman. Consciousness and complexity. *Science AAAS*, 2010.

[23] W. Ross Ashby. Requisite variety and its implications for the control of complex systems. *Cybernetica*, 1, 1958.

[24] Steven C. Bankes. Tools and techniques for developing policies for complex and uncertain systems. *PNAS*, 9:7263–7266, 2002.

[25] Melanie Mitchell and Mark Newman. Complex systems theory and evolution. *Encyclopaedia of Evolution (M. Pagel, editor)*, 2002.

[26] Stephan Halloy. A theoretical framework for abundance distributions in complex systems, 1999.

[27] Jason Brownlee. Complex adaptive systems. Technical report, Complex Intelligent Systems Laboratory, Centre for Information Technology Research, Faculty of Information Communication Technology, Swinburne University of Technology, 2007.

[28] Russ Abbott. Emergence explained: Getting epiphenomena to do real work. Emergence Explained: Getting epiphenomena to do real work, 2006.

[29] E. Ahmed, A. S. Elgazzar, and A. S. Hegazi. An overview of complex adaptive systems. *Mansoura*, 2005.

[30] Stephen Lansing. Complex adaptive systems, 2003.

[31] John H. Holland. Complex adaptive systems. *Daedalus*, 121:17–30, 1992.

[32] Jürgen Jost. *External and internal complexity of complex adaptive systems*, volume 123, pages 69–88. Springer Berlin / Heidelberg, Berlin, 2004.

[33] Adolfo López, Cesáreo Hernández, Javier Pajares, and Antonio Aguilera. Sistemas multi-agente en ingeniería de organización. técnicas computacionales de simulación de sistemas complejos, 2002.

[34] Jorge Gómez. Metodologías para el desarrollo de sistemas multi-agente. 2003.

[35] Michael Wooldridge and Nicholas Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 1995.

[36] Nicholas R. Jennings and Michael J. Wooldridge. *Agent Technology: Foundations, Applications and Markets.* Springer-Verlag New York, Inc, New York, 1998.

[37] Vicente Botti and Vicente Julian. Estudio de métodos de desarrollo de sistemas multiagente. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 18, 2003.

[38] Eugenio Dante Suarez, Antonio Rodríguez-Díaz, and Manuel Castañón-Puga. *Fuzzy Agents*, volume 154 of *Studies in Computational Intelligence*, pages 269–293. Springer, Berlin, 2007.

[39] Eugenio Dante Suarez and Manuel Castañón Puga. Distributed agency. *International Journal of Agent Technologies and Systems*, 5(3):32–52, 2013.

[40] Allen Downey. *Computational Modeling and Complexity Science: PYTHON - 2008 Edition.* CreateSpace, 2009.

[41] Alexander Artikis, Olivier Boissier, Rafael Bordini, Cristiano Castelfranchi, Rosaria Conte, Ulisses Cortés, Mehdi Dastani, Frank Dignum, Shaheen Fatima, Nicoletta Fornara, Christian Lemaitre, Victor Lesser, Fabiola López, Michael Luck, Eric Matson, Pablo Noriega, James Odell, Eugénio Oliveira, Andrea Omicini, Sascha Ossowski, Anna Perini, Paolo Renna, Antônio Carlos da Rocha Costa, Juan A. Rodríguez-Aguilar, Paul Scerri, Jaime S. Sichman, Maarten Sierhuis, Katia Sycara, Liz Sonenberg, Wamberto Vasconcelos, Javier Vazquez-Salceda, and Mario Verdicchio. *Handbook of research on multi-agent systems : semantics and dynamics of organizational models*. Hershey Pa. : Information Science Reference, New York, 2009.

[42] Karl Sigmund. Complex adaptive systems and the evolution of reciprocation, 1998.

[43] Itzhak Benenson and Paul Torrens. *Geosimulation: Automata-based Modeling of Urban Phenomeno*. Geosimulation: Automata-based Modeling of Urban Phenomeno. John Wiley and Sons, London, 2004.

[44] John Miler and Scott Page. *Complex Adptative Systems, An introduction to computational models of social life*. Princeton university press edition, 2007.

[45] Nigel Gilbert. *Computational social science: Agent-based social simulation*, pages 115–134. Bardwell, Oxford, 2007.

[46] Stuart Russell and Peter Norvig. *Inteligencia Artificial. Un Enfoque Moderno*. Pearson Prentice Hall, 2 edition, 2004.

[47] Micha Georgeff and Amy Lansky. Reactive reasoning and planning, 1987.

[48] Bogart Yail Márquez, Manuel Castañón Puga, Juan Ramón Castro, and Eugenio D. Suarez. Methodology for the Modeling of Complex Social System Using neuro-Fuzzy and Distributed Agencies. *Journal of Selected Areas in Software Engineering (JSSE)*, March:1–8, 2011.

[49] Lofti A. Zadeh. *Fuzzy sets*, volume 8. Information and Control, 1965.

[50] Lofti A. Zadeh. Fuzzy logic. *Computer*, 21(4):83–93, 1988.

[51] J. S. Jang, C. T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. MATLAB Curriculum Series. Prentice Hall, Upper Saddle River, NJ, 1997.

[52] Juan Ramón Castro, Oscar Castillo, and Luis Guillermo Martínez. Interval type-2 fuzzy logic toolbox. *Engineering Letters*, 15(1), 2007.

[53] Oscar Castillo, Patricia Melin, and Juan R. Castro. Computational intelligence software for interval type-2 fuzzy logic. *Computer Applications in Engineering Education*, 21(4):737–747, 2013.

[54] Mario García-Valdez, Guillermo Licea-Sandoval, Arnulfo Alaníz-Garza, and Oscar Castillo. Object oriented design and implementation of an inference engine for fuzzy systems. *Engineering Notes*, 15(1), August 2007.

[55] Pablo Cingolani and Jesus Alcala-Fdez. jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. *2012 IEEE International Conference on Fuzzy Systems*, pages 1–8, June 2012.

[56] C. Wagner. Juzzy a java based toolkit for type-2 fuzzy logic. In *the IEEE Symposium Series on Computational Intelligence*.

[57] J Han and M. Kamber. *Data Mining: Concepts and Techiques*. Morgan Kaufmann, 2 edition, 2006.

[58] Two-Crows. Introduction to Data Mining and Knowledge Discovery. *Two Crows Corporation*, 1999.

[59] S. Bozkir and E. Sezer. FUAT - A fuzzy clustering analysis tool. *Expert Systems with Applications, FUZZYSS11: 2nd International Fuzzy Systems Symposium*, 40:40–3: 842–849, 2013.

[60] J. Bezdek. FCM: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, page 10–2–3 and 191–203, 1984.

[61] X. Yin, L. Khoo, and Y. Chong. A fuzzy c-means based hybrid evolutionary approach to the clustering of supply chain. *Computers and Industrial Engineering*, page 66–4 and 768–780, 2013.

[62] Michael Drennan. The human science of simulation: a robust hermeneutics for artificial societies. *Journal of Artificial Societies and Social Simulation*, 8(1), 2005.

[63] Robert Lempert. Agent-based modeling as organizational and public policy simulators, 2002.

[64] Maurice Yolles. Organizations as complex systems: An introduction to knowledge cybernetics, 2006.

[65] Kenneth Boulding. General systems theory the skeleton of science. *Management Science*, 6(3):127–139, 1956.

[66] Jesse Doherty, Laurie Hendren, and Soroush Radpour. Kind analysis for matlab. *SIGPLAN Not.*, 46(10):99–118, October 2011.

[67] *MATLAB user's guide.*

[68] U. Wilensky. Netlogo. *Center for Connected Learning and Computer-based Modeling*, 1999.

[69] S. Railsback, S. Lytinen, and S. Jackson. Agent-based simulation platforms: Review and development recommendations. *Simulation*, 82(9):609–623, 2006.

[70] *MASON: A Java Multi-Agent Simulation Library.*, 2003.

[71] Liviu Panait Keith Sullivan Sean Luke, Claudio Cioffi-Revilla and Gabriel Balan. Mason: A multi-agent simulation environment. *In Simulation: Transactions of the society for Modeling and Simulation International.*, 2005.

[72] *MASON: A New Multi-Agent Simulation Toolkit.*, 2004.

[73] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. Jade: A white paper. *EXP in search of innovation*, 3(3):6–19, 2003.

[74] Manuel Castañón-Puga, Juan Ramón Castro, Josué Miguel Flores-Parra, Carelia Guadalupe Gaxiola-Pacheco, Luis-Guillermo Martínez-Méndez, and Luis Enrique Palafox-Maestre. Jt2fis a java type-2 fuzzy inference systems class library for building object-oriented intelligent applications. In Félix Castro, Alexander Gelbukh, and Miguel González, editors, *Advances in Soft Computing and Its Applications*, volume 8266 of *Lecture Notes in Computer Science*, pages 204–215. Springer Berlin Heidelberg, 2013.

[75] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998.

[76] U. Wilensky. Netlogo voting model. *Center for Connected Learning and Computer-Based Modeling*, 1998.