Trabajo de Fin de Grado

Distribución de Mercancías por Tierra – Simulación y Mejoras mediante Anylogic

Land Supply Chain - Simulation and Upgrades using Anylogic

Jesús Marín Ruiz

La Laguna, 5 de septiembre de 2017

Doña **María Belén Melián Batista**, con N.I.F. 44.311.040-E profesora Titular de Universidad adscrita al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

Don **Christopher Expósito Izquierdo**, con N.I.F. 78.851.649-J, personal docente investigador adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Distribución de Mercancías por Tierra – Simulación y Mejoras mediante Anylogic"

ha sido realizada bajo su dirección por D. **Jesús Marín Ruiz**,

con N.I.F. 03.797.140-R.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de septiembre de 2017

Agradecimientos

Mi más profundo agradecimiento a mis tutores por sus valiosas indicaciones para la realización de este trabajo, a mi esposa por la paciencia de que ha hecho gala durante la realización del mismo, a Sigfredo Melchor, cliente sobre el que se basa este trabajo, por permitirme obtener todo tipo de datos necesarios y, sobre todo, a los valiosos vídeos y tutoriales que hay disponibles en la Web, sin los cuales la realización de este trabajo no hubiera sido posible.

Licencia

Resumen

El objetivo de este trabajo ha sido el de realizar un estudio sobre posibles mejoras en la Cadena de Distribución actual de la empresa Sigfredo Melchor, S.L, dedicada al sector de la alimentación y contando con más de mil clientes a los que atender.

Esta atención se realiza mediante una flota propia de vehículos, siendo éstos de dos tipos: furgonetas para atender los pedidos urgentes y camiones para la distribución diaria.

Las furgonetas realizan entregas inmediatas, según se produzca el pedido, regresando a la central tras realizar la entrega.

Los camiones, sin embargo, realizan salidas diarias siguiendo una ruta de reparto establecida en base a los pedidos no urgentes de clientes, agrupando éstos en zonas y siguiendo el orden de menor distancia dentro de ella, y regresan a la central por dos motivos: fin de la jornada laboral, establecida en horario de 08:00 a 17:00, o fin de la mercancía a repartir.

Sigfredo, propietario de la empresa Sigfredo Melchor S.L., desea comprobar si su flota de vehículos en la isla de Tenerife es la adecuada, para lo cual necesitaría saber, en base a los pedidos de clientes, cuál es el número óptimo.

Además, desea abrir una sucursal en el sur de la isla de Tenerife para realizar una mejor atención a los pedidos urgentes de los clientes de esa zona y, al igual que en la central, desearía saber el número de vehículos necesarios.

Para estudiar estos casos, se hace uso de la herramienta Anylogic de la empresa del mismo nombre en su versión 8.1, la cual me ha permitido implementar de forma rápida y eficaz un efectivo diseño futuro a coste cero tanto para la empresa, si decide usar las mejoras que surgen de este estudio, como para mí, dada la facilidad en la realización de los estudios de optimización que surgen en el proyecto.

Palabras clave: Anylogic, Ruta, Cadena, Flota, Distribuidor, Pedido.

Abstract

The objective of this project has been to make an study about available upgrades in Sigfredo Melchor, S.L. Distribution Supply Chain company, dedicated to Food sector and having more than 1000 customers to be served.

This attention is made using its own vehicles fleet, being these of two types: vans to attend urgent orders and trucks for daily distribution.

Vans make inmediate delivery, as the order enters Central store, returning to it as soon as delivery ends.

Trucks, however, make daily distribution following delivery routes established previously based in not urgent customer orders, grouping them into island zones and following the least distance order inside it and returning to departure headquarters for two reasons: End of the working day, established from 08:00 to 17:00, or end of merchandise to be distributed.

Sigfreo, owner of Sigredo Melchor S.L. company, wishes to expand its Fleet o vehicles and has not very clear whether it is necessary and, if yes, how many additional vehicles are needed.

Besides, he wishes to open a new office on the south of the island to give a better attention to urgent order from customers in that area and, as in the main plant, would like to know the number of vans needed.

To make it, I have used Anylogic Simulator, release 8.1, which has made me certain the possibility of a quick and productive implementation of a future design at zero cost not only for me but for the company, also, if they decide to use the final advices that come from this study.

Keywords: Anylogic, Route, Chain, Fleet, Distributor, Order.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Relevancia del Reparto de Mercancías

1.1.1 Redes de Distribución

Éstas surgen principalmente por la necesidad de transportar los bienes de consumo desde su punto origen de producción o distribución hasta los clientes, ya sean finales o intermediarios.

Las decisiones sobre del diseño, planeación y operación de la cadena de suministro desempeñan un papel importante en el éxito o el fracaso de una empresa. Las fases de decisión de una cadena de suministro pueden clasificarse como diseño, planeación y operación, dependiendo del período en el cual apliquen las decisiones que se tomen (Sunil Chopra and Peter Meindl (2006). Supply Chain Management. 3° Edition. Capítulo 1. Entender qué es la cadena de suministro).

El concepto de trabajar usando suministradores y clientes es tan viejo como el comercio en sí mismo, pero la idea moderna de "cadena de suministro" es bastante reciente, probablemente no data más allá del final de los años 50 del siglo XX hasta la investigación pionera conducida por Jay Forrester y sus colegas en el MIT. Hace medio siglo, Forrester comenzó a estudiar cadenas de suministro e interrelaciones de canal entre los suministradores y los clientes e identificó un fenómeno al que posteriormente se le llamó Efecto látigo (David Blanchard (2010), Supply Chain Management Best Practices).

El Efecto Látigo se refiere a multicambios en el inventario derivados de la demanda cambiante del cliente.

Las cadenas de suministro sufren de este Efecto Látigo y del inventario de

"Trabajo en Proceso" debido a no estar sincronizados entre sí (James B. Ayers (2000) Handbook of Supply Chain Management).

1.1.2 Formas tradicionales de abordar el Reparto de Mercancías

Se puede hacer uso de una gran cantidad de medios de transporte para realizar la distribución:

- Medios aéreos utilizando, para ello, aviones que no tienen que ser necesariamente de carga, pudiendo usarse los de pasajeros para conseguir el mismo fin porque la elección de uno u otro dependerá del tamaño de la mercancía a transportar. Esta forma ha sido la última en aparecer de las mencionadas aquí pero, dada su velocidad, permite poner de forma rápida mercancía urgente en los puntos finales.
- Medios marítimos. Más baratos que los aéreos. Esto es importante ya que el coste final del producto se incrementa con el escandallo de los transportes, entre otros, lo cual hace que a transporte más barato, mejor precio final para el cliente, si el transporte lo paga el cliente, o menor costo para el distribuidor, si lo paga éste, lo que redunda en un mayor beneficio.
- Medios fluviales para transporte usando ríos como alternativa al transporte por carretera. Dependerá principalmente de los precios de uno u otro.
- Ferrocarril. Originalmente permitía cubrir de forma barata grandes distancias y conseguir tiempos más reducidos que el transporte por carretera. Éstos se fueron estrechando a medida que la tecnología conseguía mejores vehículos terrestres y de menor consumo.
- Carretera. Junto a los barcos ha sido el medio de transporte más usado hasta que la aparición de las tecnologías anteriores ha hecho que funcione de forma conjunta con ellas, siendo desplazada, en algunos casos, por estos nuevos medios, sobre todo para entregas urgentes y lejanas, p.ej.: medios aéreos.

1.1.3 Inconvenientes y Limitaciones de las formas tradicionales de Repartir Mercancía.

Cada medio de transporte anteriormente citado tiene una serie de ventajas y una serie de limitaciones. En este proyecto sólo se van a mencionar los in-

convenientes porque serán los que se intentarán mejorar en el caso de la simulación que se expone posteriormente.

Cabe, pues, citar como inconvenientes, entre otros:

- Necesidad de comprar medios para llevarlos a cabo, lo cual no siempre se puede realizar por la falta de financiación y justificación de costes.
- Por no poder o querer realizar lo anterior, se justifica la aparición de empresas que se encargan de poner la mercancía, con necesidad de ser movida, en los transportes tradicionales, recogerla en destino y llevarla hasta el punto final. Se conocen como Operadores Logísticos. Se abaratan enormemente los costes porque estos operadores sólo realizan repartos, los tienen completamente optimizados y, al no trabajar en exclusiva para el distribuidor sino para varios, los precios de transporte son bastante mejores que los que el distribuidor puede conseguir, en la gran mayoría de casos.
- Exceso de medios. Al no controlar los factores que pueden influir en los repartos, no se sabe de antemano si la compra de medios se justifica: quizás sea mejor cambiar la ubicación de una sede antes que invertir en nuevos vehículos.
- Tiempos excesivos de distribución. Al no dedicarse exclusivamente a ello, la falta de experiencia hace que los tiempos de puesta del producto en los puntos finales no sea todo lo buenos que deberían.
- El mayor inconveniente es que no se sabe con anterioridad si la inversión que se va a realizar es necesaria, si supondrá un retorno de inversión y si los puntos finales se verán influidos en los costes derivados de la nueva inversión.

1.2 Necesidad de incorporar Herramientas de Simulación

Todo lo expuesto en el apartado anterior nos convence, cada vez más, de que es necesario el uso de herramientas que permitan averiguar de antemano cómo se va a comportar una posible inversión antes de que ésta se produzca y, sobre todo, si es necesario que se lleve a cabo (Véase Figura 1.1, como ejem-

plo de vista de un programa simulador de Muelles de Carga).

1.2.1 Simulador

Según la RAE, el término simulador se aplica en Tecnología a un "*Aparato que reproduce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema*".

Esto era así y se aplicaba usualmente a sistemas concretos de hardware: aviones, autos, etc. pero el gran desarrollo que ha tenido lugar en el software, sobre todo durante los últimos años, ha permitido aplicar la simulación a otros entornos antes insospechados dada la lentitud, hasta ese momento, de los sistemas informáticos necesarios para llevarlos a cabo.

Una definición, aún más precisa y más actual, es una de las muchas usadas en la Web:

"*Recreación virtual de toda clase de escenarios, propia de los videojuegos. Los avances multimedia permiten simular, por ejemplo, desde carreras de autos a batallas romanas, pasando por todas las maniobras de un avión y completos partidos de fútbol*". [1]

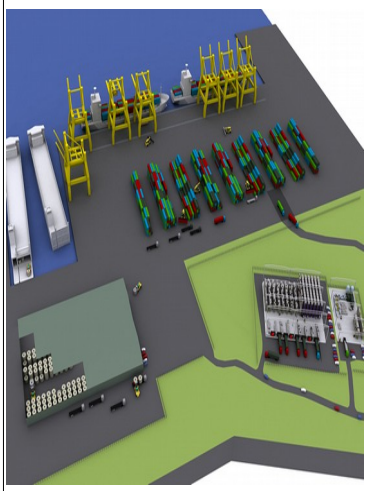


Figura 1.1: Muelle de carga

El mundo anglosajón tiene, a su vez, sus propias definiciones:

"A piece of equipment that is designed to represent real conditions, for example in an aircraft or spacecraft: People learning to fly often practise on a flight simulator"[2].

"Computer program(such as a game or animated flowchart) or a dedicated device that models (simulates) some aspects of a real life situation (such as flying an aircraft) and can be manipulated to observe the outcomes of different assumptions or actions, without exposing the experimenter to any danger or risk"[3].

1 http://www.codigodiez.mx/textosciberinfinito/diccionariodeinternet.html

2 http://www.codigodiez.mx/textosciberinfinito/diccionariodeinternet.html

3 http://www.businessdictionary.com/definition/simulator.html

El elevado coste de las nuevas implementaciones en las empresas que necesitan gestionar nuevos recursos hace que los simuladores tengan un futuro prometedor en el campo de cálculo de nuevas implementaciones de medios antes de tener que llevarlos a cabo.

1.3 Motivación personal por la temática

Mis motivaciones son varias. Como principales se han de indicar:

- Conozco Java desde la versión 2 del producto y sé de sus posibilidades así como de la gran cantidad de tiempo que se requiere en el diseño e implementación de un proyecto por lo que el uso de simuladores ya hechos en Java, y que funcionen, despierta mi curiosidad porque de tener que hacerlo desde cero, lo haría en Java, evidentemente.
- Interés por la *distribución de mercancía* ya que prácticamente casi todas las empresas que conozco la realizan de manera rudimentaria sin ningún tipo de análisis ni optimización previo.
- Modelado de casos reales usando software. Según Wikipedia[4], "*El modelado de sistemas de software es una técnica para tratar con la complejidad inherente a estos sistemas. El uso de modelos ayuda al ingeniero de software a 'visualizar' el sistema a construir. Además, los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente. Por último, las herramientas de modelado y las de Ingeniería de Software Automatizada pueden ayudar a verificar la corrección del modelo".*

  El modelado nos permite interactuar con situaciones que no podemos afrontar en el mundo real debido a varios factores:

  - complejidad
  - costo
  - destrucción de bienes
  - peligro

  y, además, como es un proceso simulado, se puede abstraer todo lo que se quiera de la realidad para quitar los datos que no son relevan-

4 https://es.wikipedia.org/wiki/Modelado_del_software

tes para el experimento.

"El modelado es más un arte que una ciencia (Anylogic7 in three days)", según Ilya Grigoriev[5] y, del mismo autor, "Modeling is about finding the way from the problem to its solution through a risk-free world we're allowed to make mistakes, undo things, go back in time and start over again". Es decir, juguemos con el "*qué pasaría si*" tantas veces como queramos sabiendo que no hay consecuencias reales y que el resultado estará totalmente optimizado sin costos adicionales al mismo: imaginemos que para construir un centro comercial es necesario expropiar y tirar una manzana de casas y que nuestro programa de simulación observa que la cantidad de clientes potenciales es inferior a la necesaria para los costes que soporta dicha construcción y puesta en marcha... El resultado será indicar la no construcción del centro y desplazarlo a otro sitio más adecuado.

5 Head of Training Services at The AnyLogic Company

Capítulo 2
Objetivos del proyecto

2.1 Resolución de problema de optimización de Reparto de Mercancía

Sigfredo Melchor S.L.[6] es una empresa canaria, con sede en Icod de los Vinos, dedicada a la distribución de productos alimenticios que dispone de una amplia cartera de más de 1.000 clientes.

Cuenta con una distribución propia de mercancía para la cual dispone de una flota de 5 furgonetas y 2 camiones frigoríficos en la isla de Tenerife. Con las furgonetas realiza repartos urgentes y con los camiones realiza repartos diarios.

En este contexto, los objetivos de este proyecto fin de grado son los siguientes:

1. Saber si la flota es óptima para los repartos que realiza cada grupo: furgonetas y camiones.
2. Analizar la creación de una sede en el sur de Tenerife que atenderá a los clientes con pedidos urgentes de esa zona. En particular se comprueba si el reparto mejora.

El objetivo fundamental que se persigue con la realización de este proyecto es crear un modelo de simulación que permita analizar el reparto de mercancías en una empresa de distribución con múltiples depósitos. Concretamente, el modelo constará de dos depósitos donde los pedidos de mercancías realizados por clientes de la isla de Tenerife llegados a cada depósito sean entregados por una flota de vehículos exclusiva para cada uno de ellos. En cada caso, el

6 http://sigfredomelchor.com

reparto de los pedidos llegados hasta cada uno de los depósitos será llevado a cabo inmediatamente por medio de los vehículos correspondientes. Opcionalmente, los pedidos pueden tener carácter de *no urgente*, lo que indica que deben ser atendidos al día siguiente de su recepción.

2.2 Introducción a la simulación

Hay dos tipos principales de modelos:

- Mental. Es el usado en la vida diaria. Todos lo utilizamos en cada momento de la misma para tomar cualquier tipo de decisión.
- Computacional (por ordenador). La llegada de los PCs como sustitutos de los Minis y su posterior abaratamiento, hizo posible poner un ordenador en casa de cada familia del planeta. Lo que hasta ese momento estaba reservado a ciertas empresas con un mínimo de poder adquisitivo, pasó a formar parte del quehacer diario de todas ellas, con una cantidad de usuarios impensable hasta ese momento.

La velocidad de los procesadores que iban apareciendo hasta llegar a los actuales consiguió también el poder usar mundos virtuales dentro de los ordenadores donde poder realizar cualquier opción que se nos ocurriera, desde Hojas de cálculo que modelaran gastos hasta complejas herramientas de simulación que nos permitieran explorar sistemas dinámicos tales como mercados de consumo o incluso campos de batalla.

Esta evolución de los ordenadores nos ha permitido poder usar dos modelados básicos:

- Analítico. A cualquier departamento de costes de cualquier organización que preguntemos por su herramienta favorita de modelado, nos dirá rápidamente que MS Excel es la que conoce, domina y recomienda por ser ampliamente soportada y fácil de usar.

  La tecnología usada por el modelado con MS Excel es simple: se introducen datos en una celda, se filtra con alguna fórmula y obtenemos un resultado en alguna otra celda.

  Este sistema, aunque simple, presenta algunos problemas dependiendo del entorno a modelar, entre otros:

  - Falta de dependencias temporales

    - Incertidumbre
    - Comportamientos no lineales

- Simulación. Los tres casos anteriores no se comportan de forma correcta aplicando fórmulas. Para ellos es necesario usar este otro tipo de modelado.

  Este otro funciona a base de diagramas de estado con control de sus cambios. Básicamente aplica un conjunto de reglas a los estados para conseguir llegar desde un estado inicial a un estado final observando en el recorrido el comportamiento del modelo según se vaya ejecutando.

  Intentar realizar este modelado usando el modelado analítico produce un modelo spaguetti inmanejable para los usuarios y que hace que se descarte su uso tarde o temprano.

  Las principales ventajas del modelo simulado son, según Ilya Grigoryev:

  - Triunfo de este tipo de modelado donde fracasan los modelados analíticos y la programación lineal.
  - Es más fácil de desarrollar un modelo simulado que uno analítico una vez que se selecciona el nivel de abstracción a aplicar.
  - La estructura del modelo simulado se corresponde de forma natural con la del sistema a modelar.
  - Se pueden medir valores y trazar cualquier entidad dentro del nivel de abstracción y añadir más valores y estadísticas en cualquier momento.
  - La habilidad de ejecutar y animar el comportamiento del sistema en su justo momento.
  - El uso de simulaciones que soporten el modelo son mucho mejores que la presentación de sólo números mediante una Hoja de Cálculo.

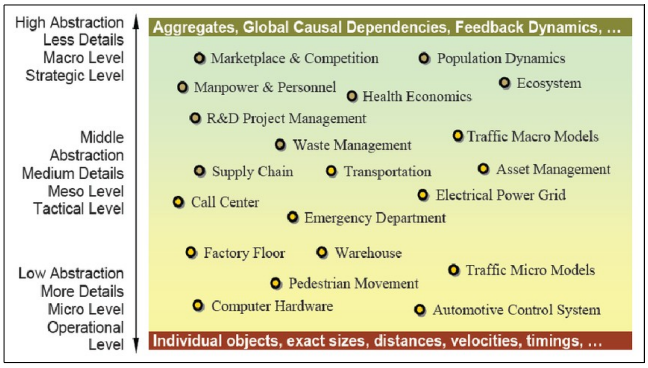


Figura 2.1: Niveles de Abstracción

La propia Figura 2.1 explicando la aplicación de las simulaciones, dependiendo del nivel de abstracción, es lo suficientemente explicativa.

Para resolver el problema planteado en el apartado anterior, voy a usar Anylogic[7] como entorno de trabajo.

2.2.1 Cómo simular

Entre los diversos simuladores que hay en el mercado, cabe mencionar como principales, por un motivo u otro:

- Anylogic. Seleccionado para este trabajo.
- Simio[8]. Parece ser el que más se acerca a Anylogic. Completamente orientado a Objetos siendo estos y sus procesos definidos de forma gráfica sin necesidad de usar programación. Permite introducir riesgos en la programación de la producción mediante su sistema patentado Planificación Basada en Riesgo y Programación.
- ISG-Virtuos[9]. Adolece de algunas simulaciones importantes. Su mejor característica, según ellos mismos, es la velocidad de ejecución de los simuladores que se desarrollen.
- Powersim Studio[10]. No tiene simulación 3d. Completo entorno de desarrollo para modelado de simulaciones.
- Simulation Modeling[11]. También bastante bueno. Permite desarrollar modelos experimentales mediante ordenador que son una copia exacta de sistemas ya existentes.

7 https://www.anylogic.com

8 http://www.simio.com

9 http://www.isg-stuttgart.de

10 http://www.powersimsolutions.com/

11 http://www.rapidmodeling.com/

La relación de las características de cada uno no son objeto de este trabajo.

2.2.2 Anylogic

Su propia página web lo define como: "*Simulation software for every business challenge" (Software de simulación para cada desafío del mundo de los negocios)*.

Para ello, han dearrollado un entorno gráfico basado en la herramienta de desarrollo Eclipse[12] y el lenguaje de programación Java[13] donde se puede modelar cada situación que pueda tener lugar en la vida real: Desde máquinas concretas hasta situaciones de evolución en negocios.

Con esta valiosa ayuda, los usuarios se pueden concentrar en modelar y Anylogic realizará el software necesario para conseguir que el modelado realizado funcione tal y como se espera de él. De hecho, cada modelo que se realiza es compilado mediante Java y el resultado se puede ejecutar sin errores. Esto, sin embargo, tiene una excepción siempre que el usuario añada su propio código de programa y éste contenga errores.

Lo mejor que Anylogic proporciona es un entorno visual donde arrastrar y soltar componentes a una rejilla y la manera de cambiar su comportamiento usando las propiedades de cada uno. Éste es el trabajo principal a realizar.

El mismo entorno usado para realizar las simulaciones, te permite utilizar tres métodos, dependiendo del nivel de abstracción (Ver Figura 2.2).

- Evento Discreto (DE) para Nivel Medio y Medio-bajo.
- Basado en Agentes (AB) para prácticamente cualquier modelo.
- Dinámicas de Sistema (SD) para modelado estratégico. Específico de Niveles altos de Abstracción.

Su explicación no es el objetivo de este manual, pero siempre se puede consultar la ayuda de Anylogic para ver cómo funciona cada uno.

En las páginas que siguen, el nivel utilizado es el Basado en Agentes.

12 https://www.eclipse.org

13 https://www.oracle.com/java/index.html

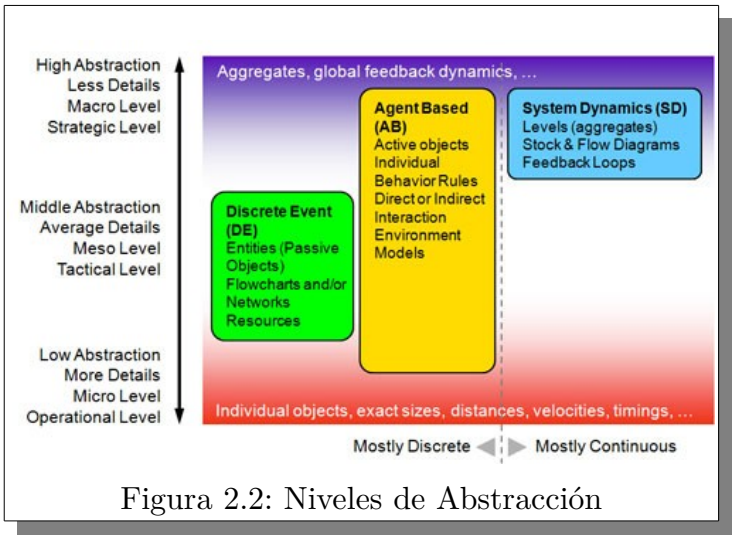


Figura 2.2: Niveles de Abstracción

2.2.3 Entorno de Trabajo Anylogic

Al usar Eclipse como entorno de trabajo, Anylogic consta de tres paneles básicos:

- A la izquierda el Árbol de proyectos (Figura 2.4) y la Paleta de componentes (Figura 2.3). En el primero, figura cada proyecto abierto en ese momento con las ramas indicando los componentes usados y el segundo está compuesto de una serie de componentes que se usan mediante arrastrar y soltar sobre la rejilla que explico a continuación.

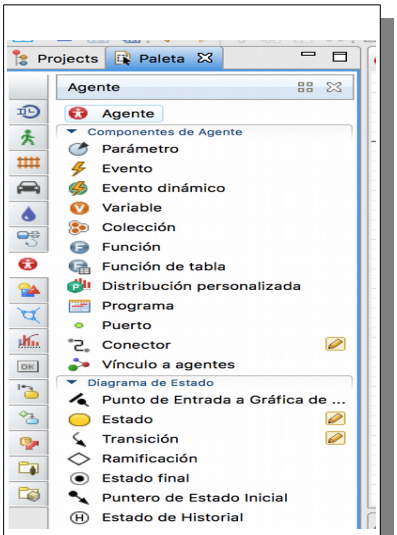


Figura 2.3: Paleta de componentes

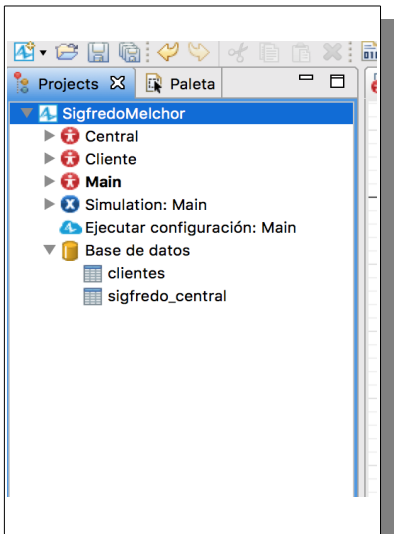


Figura 2.4: Pestaña de proyectos

- En el centro, una rejilla (Figura 2.5) para ir colocando componentes

de la paleta y visualizar los que se indiquen desde el árbol.

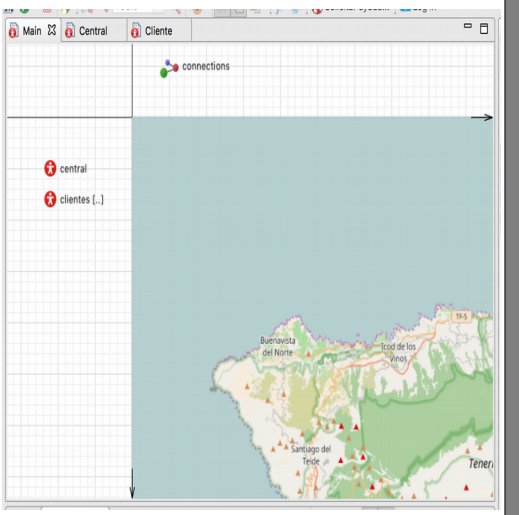


Figura 2.5: Rejilla

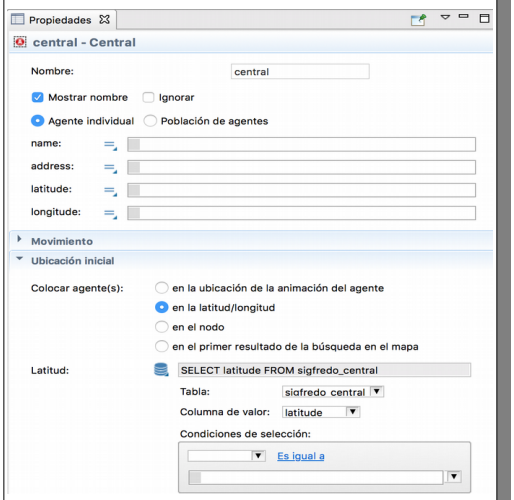


Figura 2.6: Panel de Propiedades

- A la derecha, un panel (Figura 2.6) con las propiedades de cada componente seleccionado en cada momento.

El trabajo principal consiste en depositar dentro de la rejilla central componentes seleccionados desde su panel izquierdo y cambiar sus propiedades en el panel derecho.

Capítulo 3
Problema de Distribución de Mercancías

3.1 Sigfredo Melchor S.L.

Empresa canaria dedicada a la distribución de productos de alimentación para el sector de pastelerías, panaderías, heladerías y restauración.

Consta de una amplia cartera de clientes y con presencia física en las islas de Tenerife, Gran Canaria y La Palma, distribuyendo mercancía al resto de las islas desde sus centros de Tenerife y Gran Canaria, lo que hace que sus productos lleguen a todos los usuarios de Canarias.

Para ello dispone de una flota de vehículos de dos tipos, furgonetas y camiones frigoríficos, con los que atiende a dos tipos de cliente: atención exclusiva inmediata porque el pedido es urgente y atención diaria mediante reparto por zonas de aquellos pedidos solicitados por los clientes que no se consideran urgentes y que se pueden realizar junto a otros y no de forma exclusiva tal y como se realizan los pedidos anteriores.

Las furgonetas se emplean para los pedidos urgentes: el cliente pide mercancía y se le envía una furgoneta de forma inmediata. Para esta distribución se dispone de 5 vehículos en la isla de Tenerife, 5 en la de Gran Canaria y 2 en la de La Palma.

Los camiones frigoríficos se emplean para el reparto diario: el cliente pide mercancía y cada día salen unos camiones de reparto para atenderles. Para esta distribución se dispone de 2 vehículos en la isla de Tenerife, 2 en la de Gran Canaria y 1 en La Palma.

Sigfredo no sabe si esta flota se puede considerar suficiente como para aten-

der a sus clientes y si es considerado suficiente un centro en cada isla grande: Tenerife y Gran Canaria.

El Consejo de Administración está dispuesto a realizar las inversiones necesarias tanto en vehículos como en centros de distribución pero, para ello, necesita conocer mínimamente si los cambios le reportarán una mejor atención a clientes, un recorte de gastos y un retorno de la inversión a medio plazo.

Para realizarlo, se le propone el uso de un simulador en la isla de Tenerife, en principio.

3.2 Situación actual

En la actualidad, la empresa dispone de 5 furgonetas dedicadas a servir los pedidos urgentes de los clientes:

- El cliente pide mercancía urgente a central de Sigfredo.
- Se prepara el pedido y se carga en una furgoneta.
- Se envía la furgoneta al cliente.
- Descarga la mercancía
- Regresa a la central.

  También dispone de 2 camiones de reparto diario:

- Los clientes piden mercancía no urgente a central de Sigfredo.
- Se preparan los pedidos
- Se cargan en camión teniendo en cuenta zona y distancia de cada cliente en su ficha.
- El camión realiza el reparto saliendo a las 8 de la mañana de cada día.
- Realiza la entrega a cada cliente con mercancía en el camión.
- Regresa a la central por dos motivos:
  - No queda mercancía que repartir
  - Se llega al final de la jornada laboral

3.3 Qué se ha usado

3.3.1 Tablas de Excel

Antes de comenzar a realizar el simulador, se han creado, usando la Hoja de Cálculo Swrite de LibreOffice en compatibilidad con MS Excel, los datos necesarios para un mejor desarrollo del mismo. Para ello se han creado dos tablas (Figuras 3.1 y 3.2) con columnas similares:

- Nombre de tipo String
- Dirección de tipo String
- Latitud de tipo double
- Longitud de tipo double

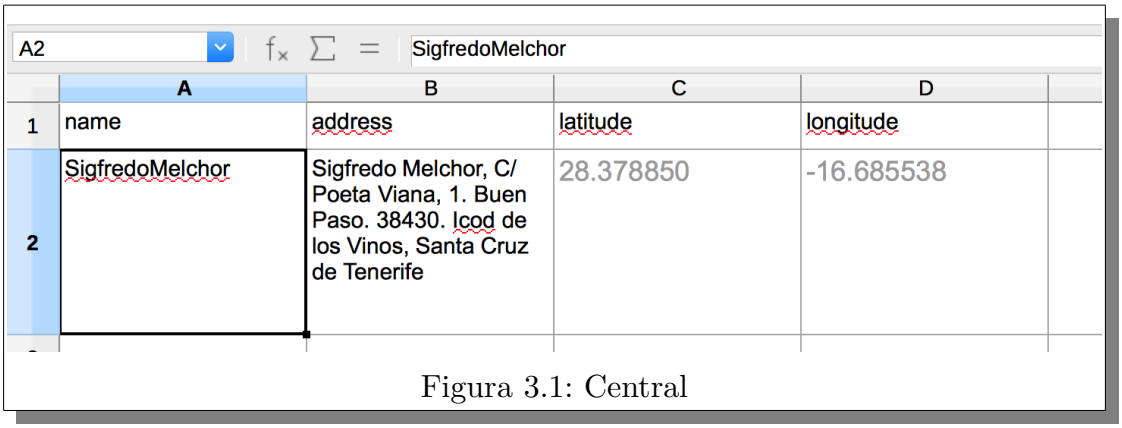
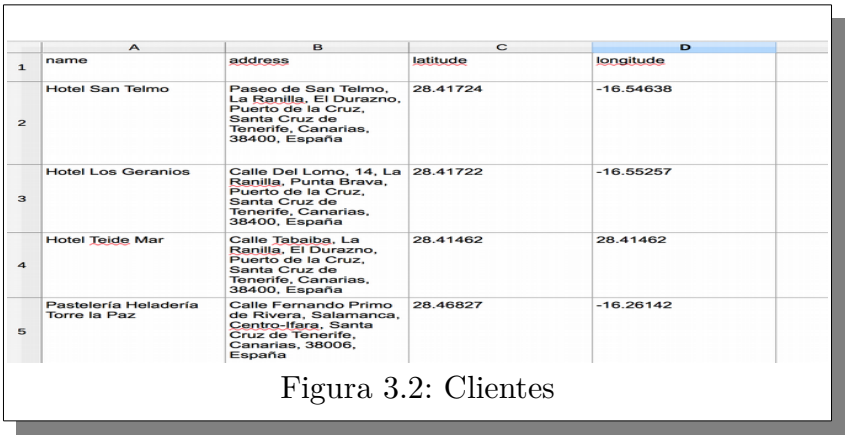


Figura 3.1: Central

El nombre y la dirección sirven para especificar la denominación de cada uno y la latitud y longitud se usan para geo-localizar cada fila en el plano. Así se puede distribuir cada valor en su sitio real dentro de la disposición del plano de la isla de Tenerife.



	A	B	C	D
1	name	address	latitude	longitude
2	Hotel San Telmo	Paseo de San Telmo, La Ranilla, El Durazno, Puerto de la Cruz, Santa Cruz de Tenerife, Canarias, 38400, España	28.41724	-16.54638
3	Hotel Los Geranios	Calle Del Lomo, 14, La Ranilla, Punta Brava, Puerto de la Cruz, Santa Cruz de Tenerife, Canarias, 38400, España	28.41722	-16.55257
4	Hotel Teide Mar	Calle Tabaiba, La Ranilla, El Durazno, Puerto de la Cruz, Santa Cruz de Tenerife, Canarias, 38400, España	28.41462	28.41462
5	Pastelería Heladería Torre la Paz	Calle Fernando Primo de Rivera, Salamanca, Centro-Ifara, Santa Cruz de Tenerife, Canarias, 38006, España	28.46827	-16.26142

Figura 3.2: Clientes

3.3.2 Procesadores de texto

Swrite de LibreOffice entrega 5.3.4.2 en formato Open de tipo odt.

3.3.3 Hojas de Cálculo

Scalc de LibreOffice entrega 5.3.4.2 en modo compatibilidad con MS Excel formato xls.

3.3.4 Capturas de pantalla

Permiten almacenar trozos de pantalla en ficheros con formato png (Ver Tabla 3.1).

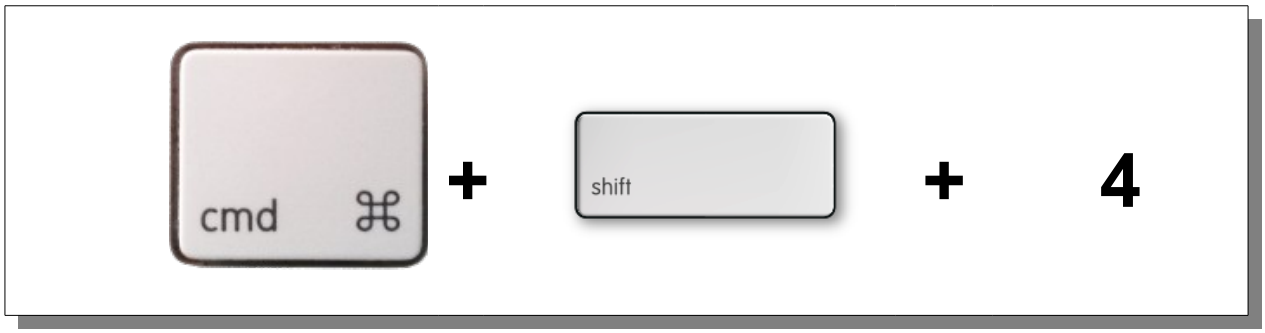


Tabla 3.1: Teclas para capturar pantallas

3.3.5 Simulador

Anylogic 8.1.0 Personal Edition.

3.3.6 Lenguaje de programación

Oracle Java SE versión 1.8 que es el que usa Anylogic y que permite dar funcionalidad adicional al simulador.

3.3.7 Hardware

Para llevar a cabo todo el desarrollo empleado y su posterior uso en Experimentos de Optimización, se ha usado el siguiente equipo:

MacBook Pro (Retina, 15-inch, Mid 2014)

- Procesador 2,5 GHz Intel Core i7 con Memoria 16 GB 1600 MHz DDR3
- Sistema Operativo Mac OS Sierra, versión 10.12.6

Capítulo 4
Creando el Simulador

4.1 Fases

El diseño de este proyecto se ha dividido en dos fases:

- Simular flota actual: furgonetas de reparto para pedidos urgentes y camiones para distribución diaria.
- Simular distribución creando una sucursal de reparto en otra ubicación de la isla.

4.2 Creando el modelo

Una vez lanzado Anylogic, pulsamos Ctrl+N para crear el nuevo modelo (Figura 4.1), le damos un nombre, en este caso *SigfredoDSC* (Sigfredo Distribution Supply Chain) y cambiamos la propiedad *Unidades de Tiempo* a horas.

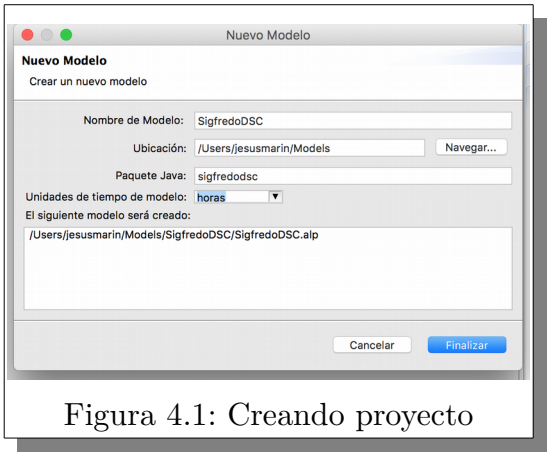


Figura 4.1: Creando proyecto

Por defecto, se crea el agente *Main* y se muestra en el panel central.

Este proyecto se va a basar en Agentes y el uso de plantillas *GIS*[14] (Geographic Information System) sobre las cuales voy a distribuir todos los agentes (furgonetas, camiones, clientes, central,...).

El sistema GIS permite visualizar, anali-

14 National Geographic defines GIS as "*a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. By relating seemingly unrelated data, GIS can help individuals and organizations better understand spatial patterns and relationships*".

zar e interpretar los resultados que vaya produciendo el simulador usando valores reales para los desplazamientos de los vehículos. De hecho, se puede ver cómo los camiones y furgonetas se desplazan a los clientes siguiendo las carreteras disponibles.

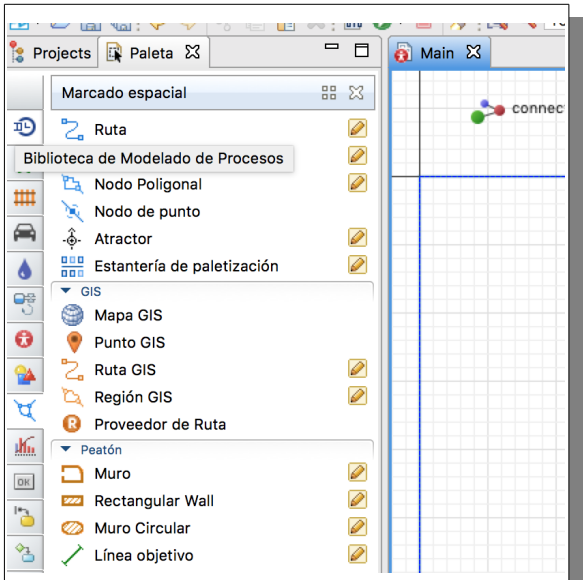


Figura 4.2: Paleta de componentes

Del panel izquierdo, seleccionamos la pestaña *Paleta* (Figura 4.2) *y*, dentro de ésta, la lista de componentes *Marcado espacial*.

Seleccionamos, con el botón izquierdo del ratón, el componente Mapa GIS y, sin soltar el botón, lo arrastramos y soltamos en la rejilla del agente *Main* ubicado en el panel central, haciendo que coincida con las líneas delimitadoras de la rejilla que allí está, quedando posicionado como sigue en la Figura 4.3:

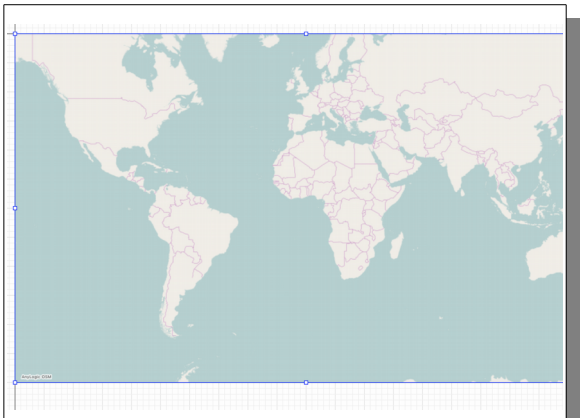


Figura 4.3: Mapa en la rejilla

Con esto hemos conseguido disponer de un mapa interactivo en nuestro proyecto para trabajar como base para nuestros agentes.

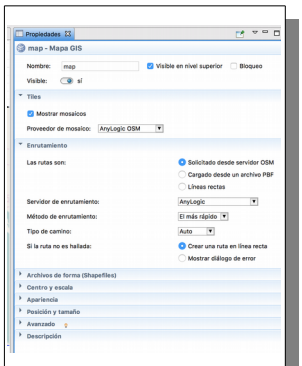


Figura 4.4: Propiedades de mapa GIS

De las propiedades del mapa, así como de todos los demás componentes que se vayan mostrando a lo largo de este proyecto, se puede cambiar cualquier valor para adaptar su funcionalidad a las necesidades que vayan surgiendo, ya sea mediante cambio manual o mediante programación java.

Para este proyecto son adecuados los valores que se ofrecen por defecto (Figura 4.4): Proveedor, tipo de camino, etc.

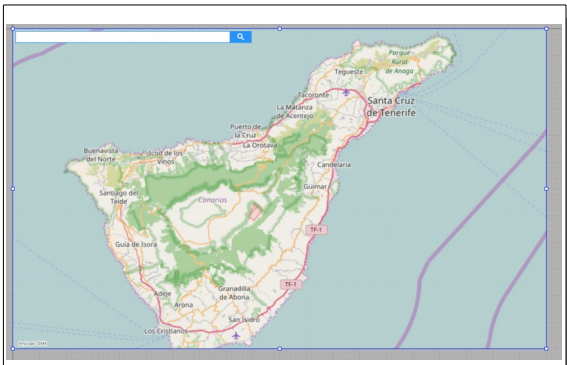


Figura 4.5: Isla de Tenerife ampliada desde mapa GIS inicial

Una vez colocado el mapa GIS en la rejilla, hay que buscar la isla de Tenerife en el mismo. Para ello, pulsamos doble clic y usando las barras de desplazamiento del ratón y la tecla de Control, ampliamos la zona de las Islas Canarias hasta lograr el tamaño adecuado (Figura 4.5). Si queremos desplazar el mapa, por ejemplo para centrarlo en la rejilla, hacemos doble clic sin soltar el dedo en el segundo clic y usando el desplazamiento del ratón, lo movemos hasta la posición adecuada.

4.2.1 Creando la Central

Para crear la central dentro de este mapa, creamos un agente al que vamos a llamar *Central*. Para ello, seleccionamos la paleta de componentes, *Agentes* y arrastramos y soltamos el componente agente a la izquierda de la rejilla del agente *Main*.

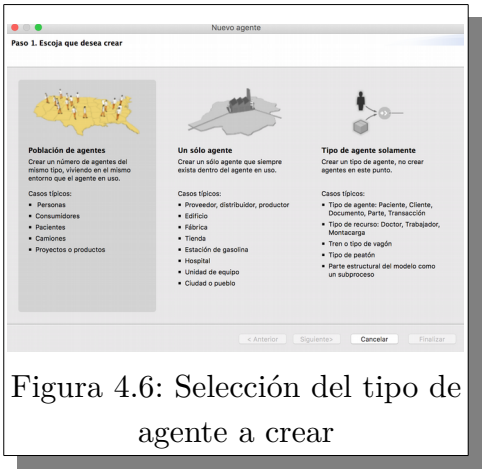


Figura 4.6: Selección del tipo de agente a crear

En esta ventana que nos aparece en la Figura 4.6, seleccionamos la creación de un solo agente y Anylogic nos muestra otra ventana (Figura 4.7) para poner el nombre con dos opciones: crearlo desde cero o usando base de datos.

En este caso, usamos base de datos (Figura 4.8), al disponer de los datos suministrados por el cliente.

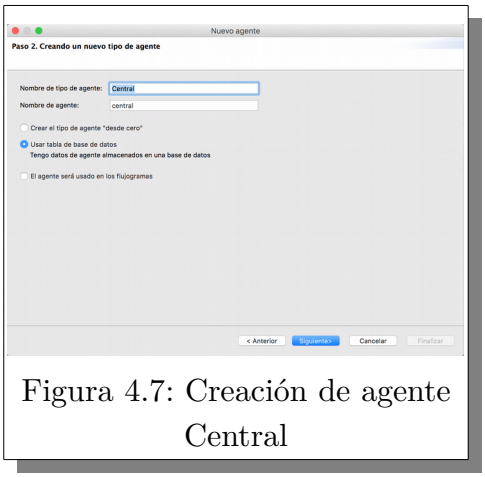


Figura 4.7: Creación de agente Central

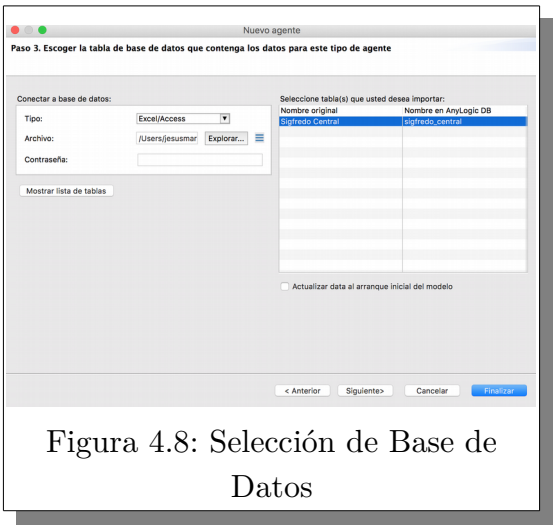


Figura 4.8: Selección de Base de Datos

Seleccionamos el Tipo *Excel* y el fichero que la contiene en *Archivo* y pulsamos siguiente.

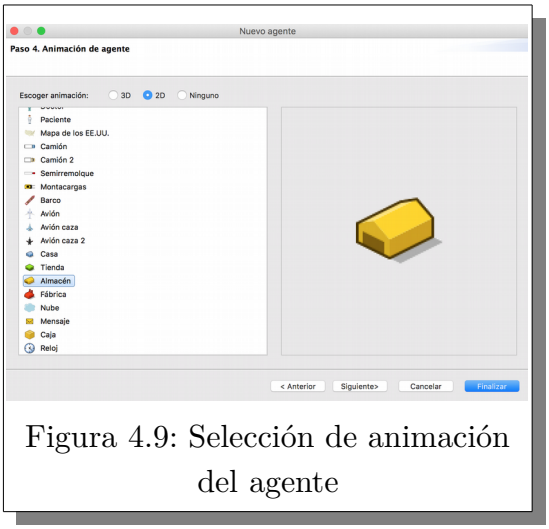


Figura 4.9: Selección de animación del agente

En esta ventana (Figura 4.9), seleccionamos la imagen que va a hacer referencia al agente, dentro del tipo 2d (se puede cambiar posteriormente) y pulso siguiente para llegar a la ventana (Figura 4.10) de selección de campos del fichero xls a procesar en el proyecto.

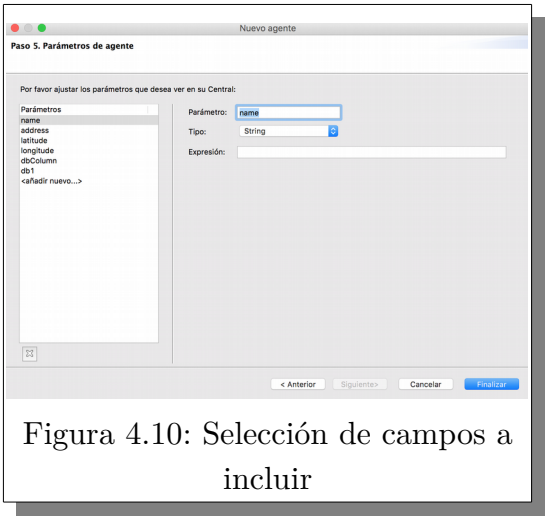


Figura 4.10: Selección de campos a incluir

En esta ventana de la Figura 4.10 se permite indicar los nombres de los campos a procesar e incluso cambiar el tipo propuesto por Anylogic.

El fichero xls con los datos leídos, se muestra en la siguiente tabla 4.1:

id	name	address	latitude	longitude
1	SigfredoMelchor	Sigfredo Melchor, C/ Poeta Viana, 1. Buen Paso. 38430. Icod de los Vinos, Santa Cruz de Tenerife	28.378850	-16.685538

Tabla 4.1: Datos de Central Distribuidora

Una vez nos encontremos en la pantalla principal, otra vez, seleccionamos en el árbol del proyecto: Main, Agentes, central y nos aparece la pantalla de propiedades de la Central. Ahí cambiamos las coordenadas del componente para que se lean desde la tabla de base de datos cargada anteriormente y seleccionando, en cada uno, la *latitud/longitud* e indicando que se usen los campos *latitude* y *longitude* que son los nombres de cada columna de la tabla sigfredo_central, según se puede ver en la Figura 4.11.

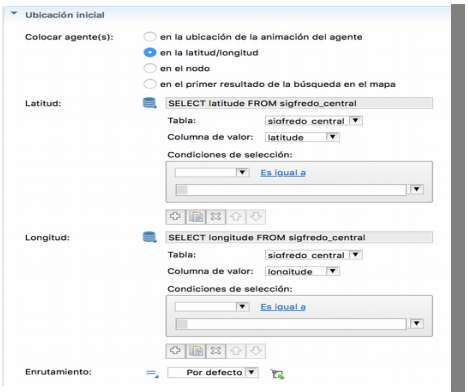


Figura 4.11: Indicar latitud y longitud desde tabla sigfredo_central

Con esto, conseguimos que el componente se ubique dentro del mapa en la posición que indican sus coordenadas.

Como normalmente la escala de la imagen del componente queda demasiado grande para el mapa donde se ubica, es necesario re-escalarlo para que se adapte mejor al mapa de fondo.

Para ello, hacemos doble clic en el componente dentro de *main* o en el árbol de proyectos y, en la ventana que aparece, clic sobre la imagen.

Esto nos muestra el panel de propiedades (Figura 4.12) donde podemos ajustar el tamaño al 40% o 50% del propuesto simplemente con cambiar la escala de X e Y a 0.40 o 0.50.

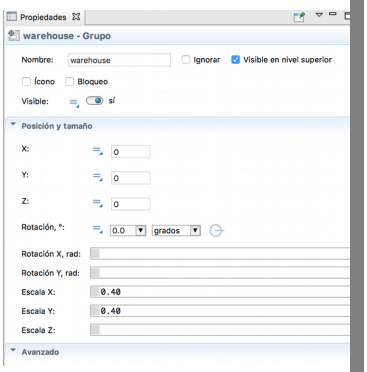


Figura 4.12: Escalado de imagen de la central

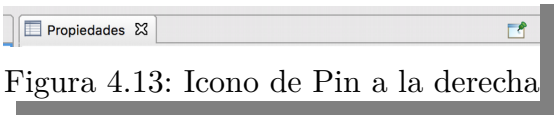


Figura 4.13: Icono de Pin a la derecha

Nota.- En la versión 8.x de Anylogic, las propiedades asociadas a un solo clic del componente, a veces, no aparecen,

pero si hacemos doble clic en el panel de propiedades sobre el icono superior *pin* de la derecha (ver Figura 4.13) y luego clic sobre el componente en cuestión, aparece el panel correcto.

Tanto la ubicación indicada en la latitud y longitud como su escala no se reflejan correctamente sobre el mapa hasta que ejecutemos el simulador mediante F5 o haciendo clic en el icono *Play* de la barra de herramientas.

4.2.2 Creando los Clientes

Ahora crearemos los clientes. Los diferentes pasos a seguir son similares a la creación de la Central.

Para ello:

- seleccionamos Paleta de componentes,
- Agente y lo arrastramos y soltamos en *Main*,
- seleccionamos Población de Agentes,
- al nuevo tipo lo llamamos Cliente,
- seleccionamos Base de datos desde fuente externa (fichero xls con clientes),
- le asociamos imagen
- y finalizamos.

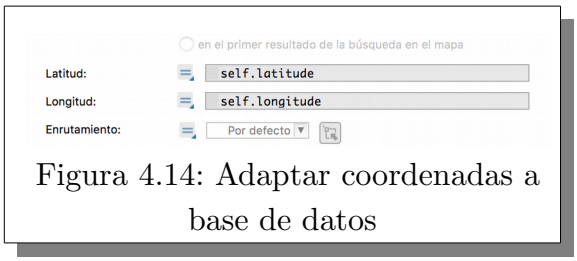


Figura 4.14: Adaptar coordenadas a base de datos

Hacemos clic sobre el componente dentro de *Main* y cambiamos su latitud y longitud como sigue en la Figura 4.14. No contiene valores similares a los usados en la Central, por ser los de ahora pertenecientes a una población.

Hacemos clic sobre imagen del componente dentro del Tipo del mismo y lo re-escalamos a 40% o 50% según queramos verlo.

Nota.- Si al ejecutar el simulador me avisa de que la Base de Datos no es única, esto indica que hay que añadir una clave única a la BD importada. Para ello, seleccionamos en el árbol del proyecto la rama Base de datos y dentro de ella la Tabla en cuestión, marcando en Propiedades la columna que establecemos como única, *id* por ejemplo. Si la columna seleccionada contiene duplicados, Anylogic me lo indicará: seleccionemos entonces otra columna o

eliminemos los duplicados.

id	name	address	latitude	longitude	distanceTo1	timeTo1	zone
1	Hotel San Telmo	Paseo de San Telmo, La Ranilla, El Durazno, Puerto de la Cruz, Santa Cruz de Tenerife, Canarias, 38400, España	28.41724	-16.54638	NE 19.7	NE 25	NE

Tabla 4.2: Datos de Clientes

Para su uso en la ruta de reparto, cada cliente dispone de una columna *distanceTo1,* según se puede ver en la Tabla 4.2, que contiene la zona de ubicación y la distancia desde la central. Esto nos permite cargar una Ruta de distribución en cada camión con los clientes destinatarios clasificados de menor a mayor cercanía con respecto a la central.

4.2.3 Rutas

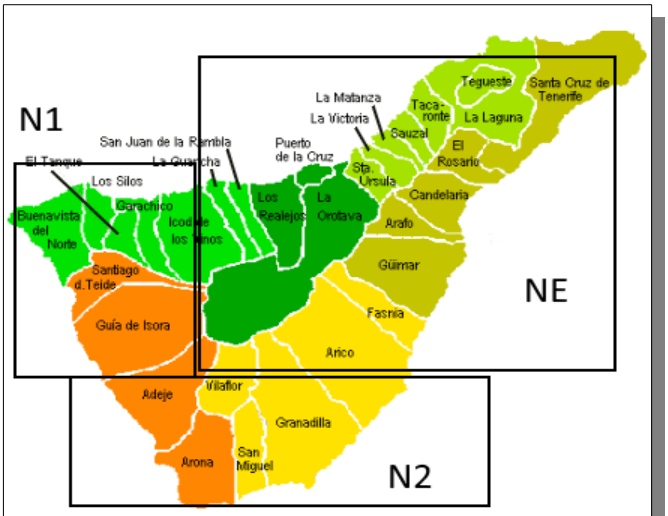


Figura 4.15: Zonas de Tenerife

Se ha dividido a la isla de Tenerife en tres grandes zonas, según se puede observar en la Figura 4.15, tomando como punto de partida la Central que se encuentra en Icod:

- N1 con los clientes al oeste.
- N2 con los clientes al sur.
- NE con los clientes al este.

Como de cada cliente se dispone de la columna *distanceTo1* con

la zona de ubicación y la distancia hasta la central, si clasificamos la ruta de distribución de los clientes a visitar, mediante esta columna, obtenemos una ruta de visitas clasificada de más a menos cercanía con respecto a la central.

Esto nos va a permitir visitar primero a los clientes de la zona N1, luego a los de la N2 y finalmente a los de la zona NE, obteniendo una distribución no 100% óptima, pero sí bastante idónea.

4.2.4 Creando furgonetas

Ahora vamos a crear las furgonetas disponibles en Sigfredo para los repartos urgentes. Los diferentes pasos a seguir son similares a la creación de la Central.

Para ello:

- seleccionamos Paleta de componentes, Agente
- y lo arrastramos y soltamos en Main,
- seleccionamos Población de Agentes, al nuevo tipo lo llamamos Furgoneta,
- indicamos que se cree desde cero,
- le asociamos imagen,
- añadimos parámetro con nombre *aCliente* de tipo *Cliente*
- y finalizamos.

Este parámetro nos va a permitir indicar posteriormente el cliente destino del envío.

Ahora lo re-escalamos de forma similar a como hemos hecho con la Central y los Clientes a un 0.40 o al porcentaje que nos resulte más cómodo a la vista.

Para indicar el número de furgonetas de que disponemos, creamos un parámetro dentro de *Main* de forma similar a como hemos creado agentes hasta ahora:

- Seleccionamos paleta de componentes
- Componentes de agentes
- Parámetros

Lo llamamos *cantidadFurgonetas* y le asignamos el tipo int y 5 de valor inicial.

El tener el número disponible de furgonetas almacenado en un parámetro de *Main,* nos permitirá cambiarlo durante los experimentos que vamos a realizar.

Para ubicar las furgonetas en la Central, inicialmente, creamos un Punto GIS en las coordenadas de la central, lo llamamos GISPointCentral y las furgonetas de *Main* las ubicamos en el nodo GISPointCentral.

4.2.5 Creando Pedidos de Clientes

Para conseguir que los clientes empiecen a realizar pedidos urgentes a la Central, es necesario crear un Tipo de Agente nuevo sin instanciarlo en *Main*, de momento.

Procedemos como hasta ahora:

- Seleccionamos paleta de componentes
- Componentes de agentes
- Arrastrar y soltar Agente a *Main*
- Seleccionar Tipo de Agente solamente
- Nombrarlo como *Pedido*
- Marcar *Crear agente desde cero*
- No seleccionar animación alguna
- Añadir parámetro nuevo de tipo *Cliente* y nombre *pedidoCliente*

4.2.6 Creando marca visual de "Cliente haciendo pedido"

Para conseguir visualizar en el mapa aquellos clientes que se encuentran en situación de *Pedido realizado*, es necesario crear un pequeño círculo ovalado que ubicaremos en cada cliente y que se encenderá y apagará dependiendo de si el cliente está en proceso de Pedido o ya le ha llegado la mercancía.

Para ello:

- Seleccionamos la paleta de componentes.

- Presentación
- Oval y lo arrastramos y soltamos en Main
- Lo reducimos lo suficiente como para que se puede colocar sobre la imagen de los clientes
- Cambiamos el color a *tomato* para *relleno* dejando *línea* como está.

Este componente se visualizará mediante gráficos de estados de los clientes que se explicarán a continuación.

Como posteriormente hay que definir los Pedidos semanales de clientes, aprovechamos y creamos otro componente oval para este tipo de pedidos y le damos color *deep sky blue*.

4.2.7 Creando Gráficos de Estado de los Clientes

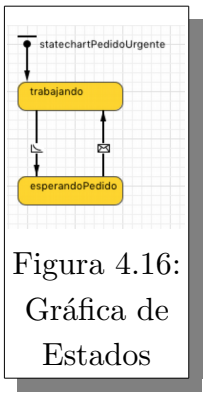


Figura 4.16: Gráfica de Estados

Para controlar los pedidos de los clientes al distribuidor, es necesario crear unos Gráficos de Estado que nos permitan hacer transiciones de uno a otro desde el estado inicial de Trabajando, transitando a *Esperando pedido* mediante *Realizado pedido* hasta el estado final de Trabajando, de nuevo, al transitar mediante *Pedido recibido* (Ver Figura 4.16).

Para ello:

- Seleccionamos la paleta de componentes
- Diagrama de Estado
- Seleccionamos *Punto de Entrada*, lo arrastramos y soltamos sobre la rejilla de los clientes y lo llamamos *statechartPedidoUrgente*.
- Arrastramos y soltamos un Estado, lo llamamos *trabajando* y lo enlazamos con el anterior.
- Arrastramos y soltamos una transición. Indicamos que la desencadena una Tasa de *main.pedidosUrgentes* por semana.
- Arrastramos y soltamos un Estado, lo llamamos *esperandoPedido* y en Acción de Entrada introducimos el código

```
PedidoUrgente pedido =
    new PedidoUrgente(this, true);
send(pedido,main.central);
```

que crea un pedido de este cliente y lo envía como mensaje al objeto central de main, poniendo true como valor del segundo parámetro para indicar que es urgente.

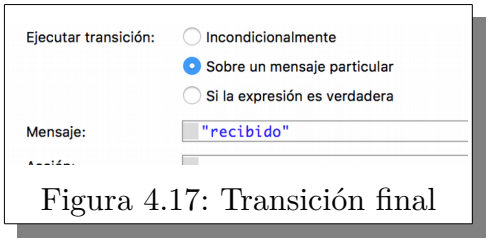


Figura 4.17: Transición final

- Arrastramos y soltamos otra transición que va a recibir el mensaje de "*recibido*", según se ve en la Figura 4.17, el cual será el que ponga ésta en funcionamiento. Luego indicamos que se ejecute la transición sobre un mensaje en particular. Esta transición unirá el estado anterior con el primero, es decir *esperandoPedido* con *trabajando*.

4.2.8 Creando Diagrama de Modelado del Proceso de Pedidos Urgentes

El diagrama que viene a continuación nos va a permitir "*usar combinaciones de agentes, recursos y procesos para crear un modelo central de nuestro sistema de mundo real" (Ilya Grigoriev en Anylogic 7 in Three days, pág. 144),* Figura 4.18.

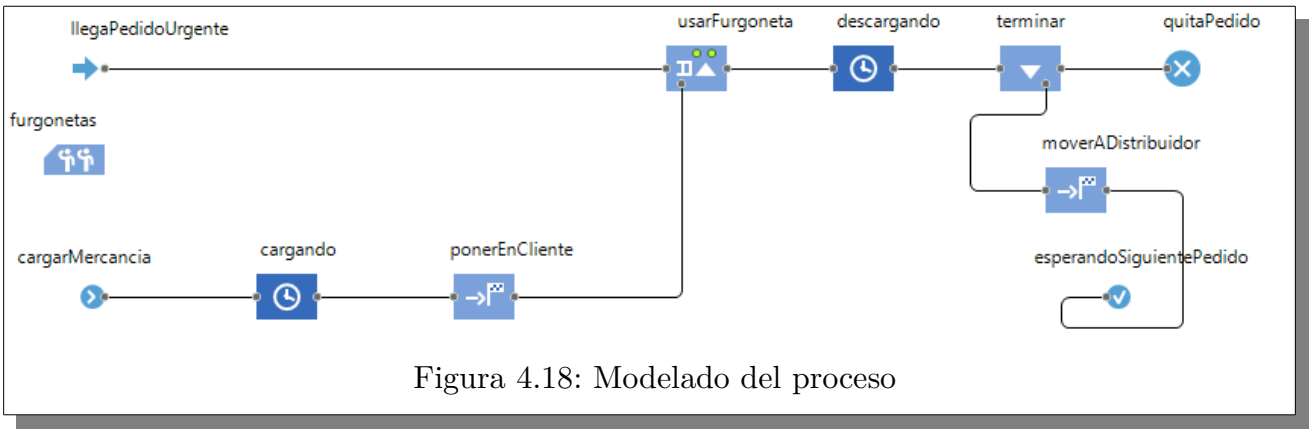


Figura 4.18: Modelado del proceso

La figura anterior muestra una visión global del proceso de Pedidos Urgentes de clientes, desde su recepción en la central y separación de la mercancía hasta su carga en la furgoneta, desplazamiento a cliente, entrega y regreso a central.

El punto de partida es el mensaje recibido en la central enviado por un cliente. Este mensaje se envía desde el estado *esperandoPedido* del diagrama

de estados del agente *Cliente*

```
send(pedido,main.central);
```

y se recibe por la conexión de la central: componente *connections de la Figura 4.19*.

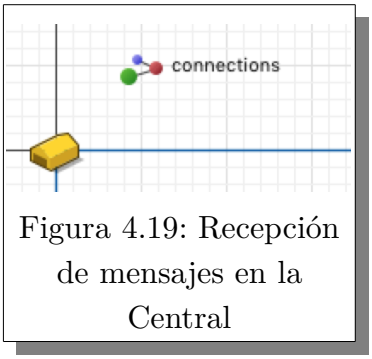


Figura 4.19: Recepción de mensajes en la Central

siendo la acción realizada la que se indica en *Sobre el mensaje recibido* de la siguiente figura de las propiedades de connections de la Figura 4.20:

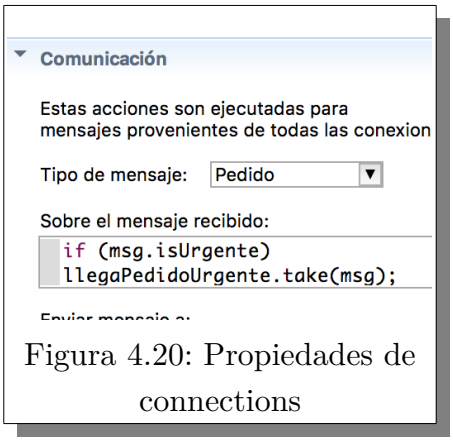


Figura 4.20: Propiedades de connections

donde, como se ve, llama al método *take* de *llegaPedidoUrgente* con el parámetro *msg* que es la manera de hacer referencia al mensaje recibido del tipo *Pedido* y que ha sido enviado por el cliente desde el diagrama de estados.

En realidad *msg* es la instanciación de la clase *Pedido* y, como tal, permite el uso de todos los métodos públicos de la citada clase.

Para averiguar los métodos disponibles, escribir *msg.* en *Sobre el mensaje recibido* y pulsar Alt+Espacio (Mac) o Ctrl+Espacio (Windows), para mostrar una lista de los métodos y propiedades.

Como la clase *Pedido* dispone de la propiedad *isUrgente*, la podemos usar para controlar si el pedido lo es.

4.2.9 Descripción de cada componente

llegaPedidoUrgente (Figura 4.21)

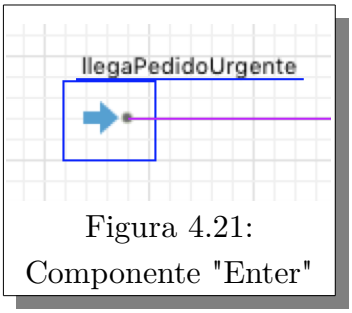


Figura 4.21: Componente "Enter"

Es del tipo *Enter*. Según la ayuda del propio componente *Permite insertar agentes ya existentes en un punto en particular del flujo del modelo de proceso. Este objeto se usa típicamente para transferir agentes ya creados dentro del flujo del proceso o, al igual que su objeto complementario, Exit, para implementar ruta habitual en el modelo del proceso.*

El método take es la forma de insertar el agente con este objeto y sus propiedades son las indicadas en la Figura 4.22.

Figura 4.22: Propiedades del componente "llegaPedidoUrgente"

furgonetas (Figura 4.23)

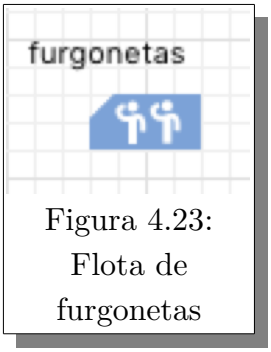


Figura 4.23: Flota de furgonetas

Este componente se usa para controlar la flota de furgonetas disponible.

Su objeto es del tipo *ResourcePool* (Relación de Recursos). Según la Ayuda de Anylogic *Define un conjunto de unidades de recurso que pueden ser recogidas mediante Seize, entregadas con Release, ensambladas mediante Assembler y bloques de diagrama de flujo de Servicio* y sus propiedades son las descritas en la Figura 4.24.

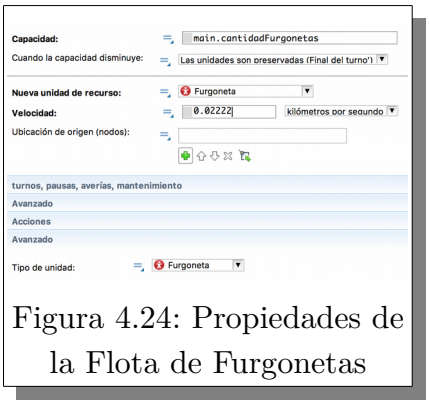


Figura 4.24: Propiedades de la Flota de Furgonetas

Obsérvese cómo la Capacidad se carga desde el parámetro *cantidadFurgonetas* de *Main*, el tipo de Recurso es *Furgoneta* y la velocidad es 0,02222 Km/s que equivale a 80 Km/h o 22,22 m/s. También se puede poner 80 km/h pero en la versión española no figura esta medida y sí kilómetros por segundo, que puede ser un error de traducción.

Para poder ver luego en el experimento los desplazamientos con claridad, será necesario bajar la velocidad 10 veces, al menos.

cargarMercancia (Figura 4.25)

Figura 4.25: Inicio de Tarea de Recurso

Este objeto es del tipo *ResourceTaskStart.* Según la ayuda de Anylogic *Define el inicio de la rama de flujo modelando el proceso de tareas para las unidades de recurso (normalmente es un proceso de preparación de recursos)* y su propiedades son las indicadas en la Figura 4.26 donde sólo se cambia el Tipo de unidad a Furgoneta.

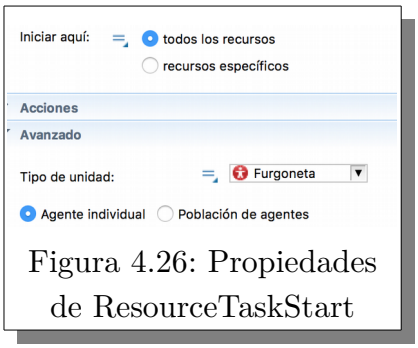


Figura 4.26: Propiedades de ResourceTaskStart

cargando (Figura 4.27)

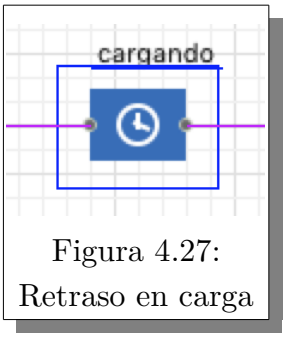


Figura 4.27: Retraso en carga

Este objeto es del tipo *Delay*. Según la ayuda de Anylogic *Crea un retraso en los agentes durante un período determinado de tiempo. El valor de retraso se evalúa dinámicamente, puede ser estocástico*[15] *y puede depender del agente así como de otras condiciones. Opcionalmente se puede calcular como la longitud de la ruta de animación del Retardo dividida entre la Velocidad del agente. Se pueden retardar a la vez múltiples agentes (hasta la capacidad de Retardo suministrada) y de forma independiente.*

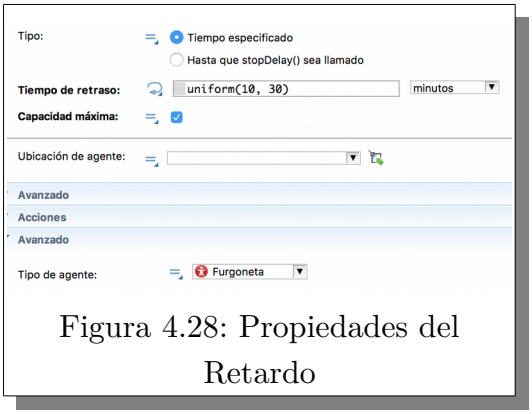


Figura 4.28: Propiedades del Retardo

Sus propiedades son las indicadas en la Figura 4.28. El tipo de retardo es *Tiempo especificado* y su valor es *uniform(10, 30) minutos* que indica un retraso entre 10 y 30 minutos de media.

ponerEnCliente (Figura 4.29)

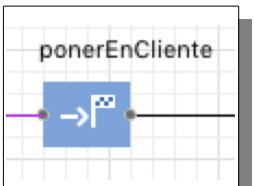


Figura 4.29: Componente "MoveTo"

Este componente es del tipo MoveTo. Según la ayuda de Anylogic *Mueve el agente a una nueva localización. Si algún recurso se encuentra adherido al agente, se mueve también. La velocidad será la del agente a pesar de la velocidad de los recursos adheridos. El tiempo que emplea el agente en este objeto es igual a la longitud de la ruta más corta desde la localización actual del agente hasta el destino dividida entre la velocidad del agente. El agente es animado a lo largo de la ruta.*

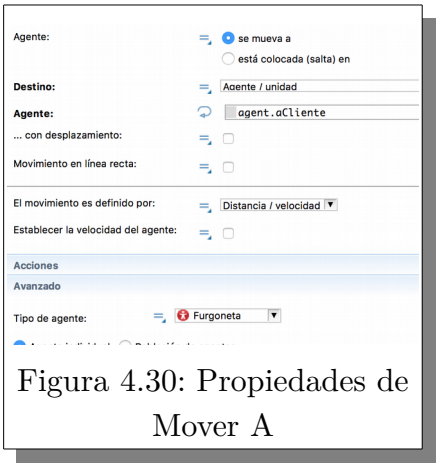


Figura 4.30: Propiedades de Mover A

Sus propiedades son las indicadas en la Figura 4.30.

El tipo es *Furgoneta*, el Agente *se mueve* a Agente y *agent.aCliente* nos indica a dónde.

15 Según la RAE: Perteneciente o relativo al azar.

Es decir, la furgoneta es el *Agente*, llamado *agent* (forma de referenciar a cada agente dentro del componente), y se *mueve a* destino indicado por el parámetro *aCliente* de la furgoneta.

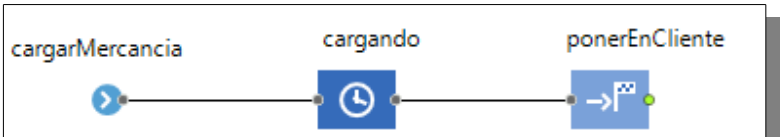


Figura 4.31: Proceso de carga de la mercancía en la furgoneta

Resumiendo: los tres componentes indicados para la carga de la furgoneta son los que se ven en la Figura 4.31:

- Cargar mercancía.
- Separación y Carga de mercancía en furgoneta con un tiempo estimado entre 10 y 30 minutos.
- Mover furgoneta a cliente especificado en *furgoneta.aCliente*.

usarFurgoneta (Figura 4.32)

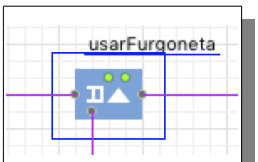


Figura 4.32: Usar furgoneta

Este componente es del tipo *Seize*. Según la ayuda de Anylogic *Coge un número dado de unidades de recurso desde un ResourcePool suministrado. Opcionalmente envía los recursos cogidos a una localización especificada.*

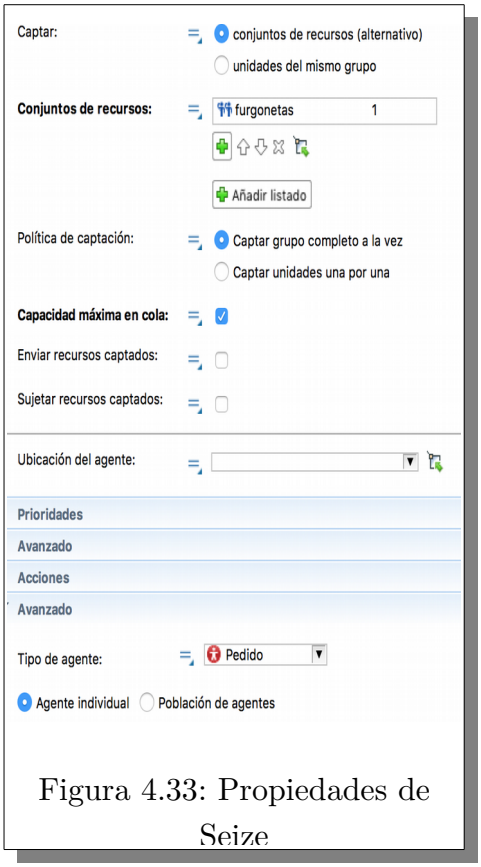


Figura 4.33: Propiedades de Seize

Sus propiedades son las observadas en la Figura 4.33.

Sus principales valores son: el Tipo de agente como Pedido e individual con capacidad máxima en cola y captación completa del grupo. La captación es de una unidad de un conjunto de recursos alternativo (furgonetas) y todas a la vez.

El evento *Al captar la unidad* contiene:

```
((Furgoneta) unit).aCliente = agent.deCliente;
```

que asigna el cliente del pedido al cliente del camión captado en los recursos.

Lo que hace este componente es recoger un Pedido por el puerto de entrada (*in)*, asignarle una furgoneta por el puerto de asir (*seize)* y dar salida a ambos por el puerto de salida (*out*).

descargando (Figura 4.34)

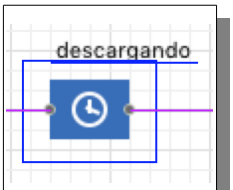


Figura 4.34: Retardo al descargar

Este componente es similar al anterior de *Separación y Carga* de mercancía en la furgoneta. El tiempo empleado, en este caso, oscila entre 15 y 30 minutos, según se especifica en las propiedades de la Figura 4.35.

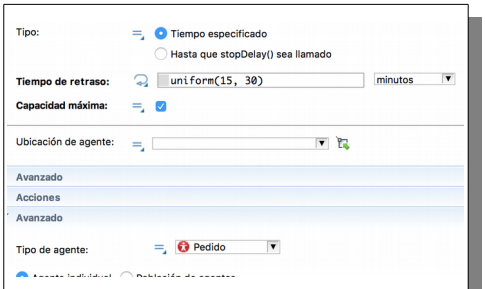


Figura 4.35: Propiedades de Descarga

Como se ve, es totalmente similar al componente definido con anterioridad.

terminar (Figura 4.36)

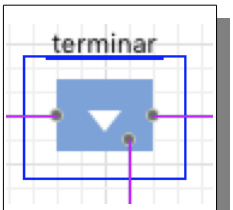


Figura 4.36: Componente Release

Este componente es del tipo *Release*. Según la ayuda de Anylogic *Libera un número determinado de unidades de recurso previamente cogidas por el objeto Seize. Los recursos movidos pueden Volver a localización inicial (si no son cogidos de forma inmediata por otro agente) o quedarse donde están tras ser liberados. La operación tarda cero segundos en terminar. Sink, que es el componente final, es obligado que se use en paralelo para consumir el componente que sale por el puerto out.* Sus propiedades son las indicadas en la Figura 4.37. Indicamos que la furgoneta (recurso asociado en *Seize*) vuelva a la posición inicial y que el agente es de tipo Pedido.

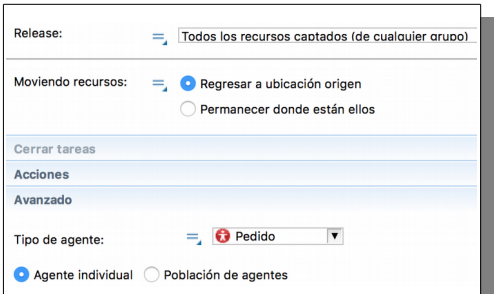


Figura 4.37: Propiedades de Release

quitaPedido (Figura 4.38)

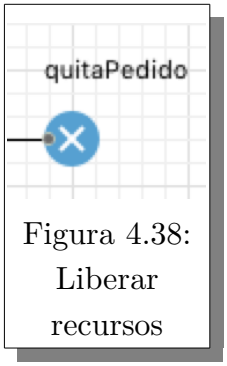


Figura 4.38: Liberar recursos

Este componente es del tipo *Sink*. Según la ayuda de Anylogic *Se deshace de los agentes. Es el punto final de un proceso de modelado. A menos que se use Sink los agentes no se quitan del modelo lo cual dará lugar a problemas de memoria, seguramente.*

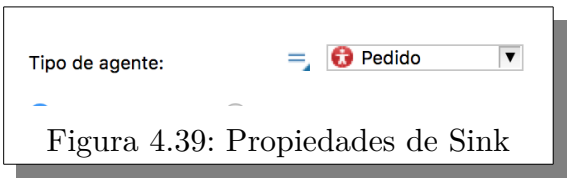


Figura 4.39: Propiedades de Sink

Sus propiedades son las indicadas en la Figura 4.39 donde sólo se establece Pedido como agente a terminar.

moverADistribuidor (Figura 4.41)

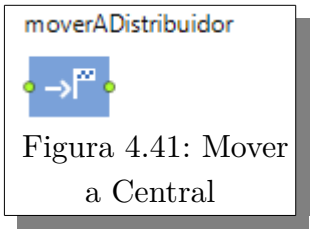


Figura 4.41: Mover a Central

Este componente es del tipo *MoveTo* y permite mover la furgoneta a la central.

Para ello, se indica en las propiedades de la Figura 4.40 que el agente *se mueva a main.central*.

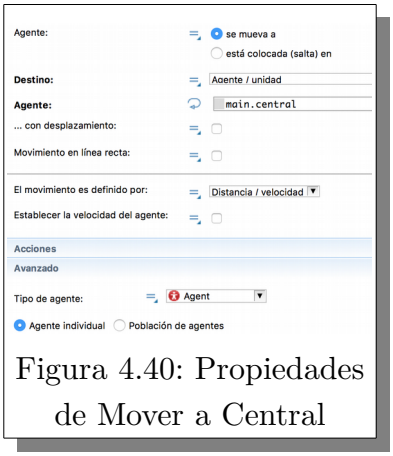


Figura 4.40: Propiedades de Mover a Central

esperandoSiguientePedido (Figura 4.42)

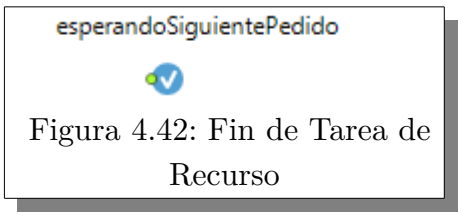


Figura 4.42: Fin de Tarea de Recurso

Componente del tipo ResourceTaskEnd. Según la ayuda de Anylogic *define el final de diagrama de flujo del modelado para las unidades de recurso.*

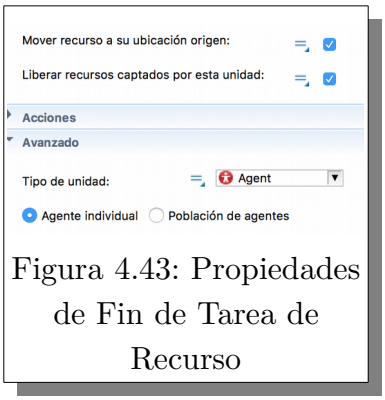


Figura 4.43: Propiedades de Fin de Tarea de Recurso

Sus propiedades son las indicadas en la Figura 4.43, indicando que el recurso vuelva a su posición inicial y que se liberen los recursos captados por la unidad.

Con todo lo explicado hasta ahora en este Modelado de Proceso y para que sirva de resumen:

- La central recibe un pedido urgente
- Separa y carga la mercancía en una furgoneta tardando entre 10 y 30

minutos

- El furgón sale hacia el cliente
- Cuando llega, descarga la mercancía tardando entre 15 y 30 minutos
- El furgón regresa a la central

4.2.10 Creando Diagrama de Modelado del Proceso de Pedidos no Urgentes

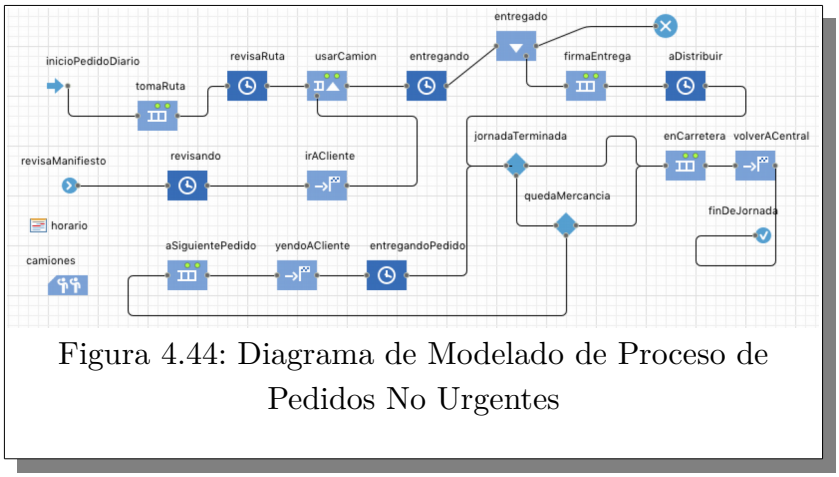


Figura 4.44: Diagrama de Modelado de Proceso de Pedidos No Urgentes

La Figura 4.44 muestra una visión global del proceso de Pedidos No Urgentes de clientes, desde su recepción en la central y separación de la mercancía hasta la carga de la Ruta a seguir en el camión, desplazamiento a cliente, entrega a todos los clientes de la Ruta y regreso a central.

El punto de partida es el mensaje recibido en la central enviado por un cliente. Este mensaje se envía desde el estado *esperandoPedidoSemanal* del diagrama de estados del agente *Cliente*

```
Pedido pedido = new Pedido(this, false, -1);
send(pedido,main.central);
```

donde this es el propio pedido, false indica que no es urgente y -1 es el camión que lo está gestionando, en este caso, ninguno, de momento.

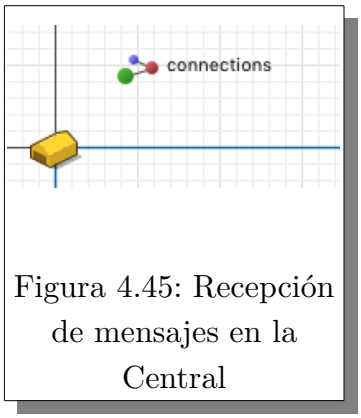


Figura 4.45: Recepción de mensajes en la Central

Y se recibe por la conexión de la central (Figura 4.45), componente *connections*.

Siendo la acción realizada la que se indica en *Sobre el mensaje recibido* de connections:

```
if (msg.isUrgente) {
        llegaPedidoUrgente.take(msg);
} else {
        addPedidoDiario(msg);
}
```

donde, como se ve, llama al método *addPedidoDiario* (explicado en el Apéndice 1) pasándole el Pedido en el campo *msg* y allí se queda almacenado hasta que los camiones recojan al día siguiente todos los pedidos de la ruta que han de entregar para iniciar la Ruta de Reparto.

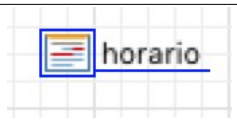


Figura 4.46: Controlar reparto

Esto se realiza mediante el componente *Schedule* (*Programa*) (Figura 4.46), cuyas propiedades (Figura 4.47) indican las acciones a realizar y cuándo, a base de intervalos de duración semanal.

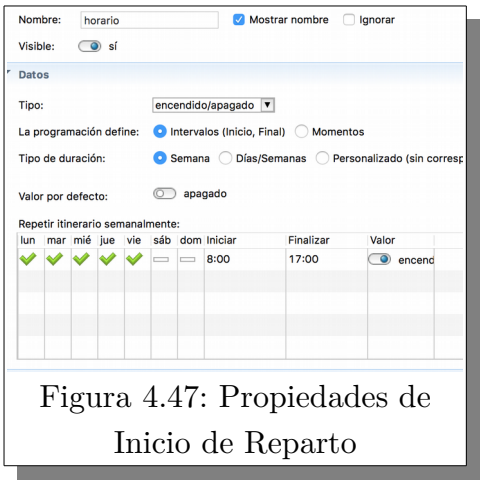


Figura 4.47: Propiedades de Inicio de Reparto

Como se puede ver, el reparto se inicia cada día a los 08:00 horas, de lunes a viernes, y carga la Ruta a seguir mediante las acciones, explicadas en el Apéndice 1:

```
cargarCamiones();
iniciarCamionesReparto();
```

quedando la acción de este componente:

```
if (self.getValue()) {
 cargarCamiones();
 if (isRepartir) {
 iniciarCamionesReparto();
 }
} else {
 isRepartir = false;
}
```

El reparto se termina por dos motivos:

- Fin de la jornada laboral, controlada mediante el mismo componente de la Figura 4.46 y cuya única acción consiste en poner el parámetro isRe-

partir a false cada día a las 17:00 horas. Este parámetro se controla en el ciclo de distribución y hace que el camión regrese a origen cuando se acabe la jornada laboral.

- Fin de mercancía a distribuir, regresando el camión en ese momento a origen.

4.2.11 Descripción de cada componente de la Figura 4.44

inicioPedidoDiario (Figura 4.49)

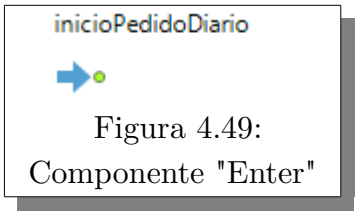


Figura 4.49: Componente "Enter"

La explicación de este componente es idéntica a la ya realizada en la Furgoneta. Sus propiedades son similares (Figura 4.48).

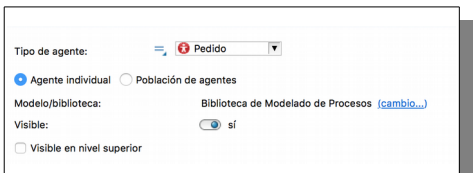


Figura 4.48: Propiedades del componente "inicioPedidoDiario"

tomaRuta, usarRuta (Figura 4.50)

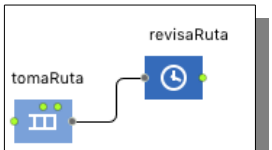


Figura 4.50: Iniciar Ruta

Estos dos componentes inician la recogida de la Ruta a Repartir, antes de recoger el camión.

revisaManifiesto, revisando e irACliente (Figura 4.51)

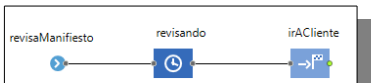


Figura 4.51: Carga del camión

Los componentes de la figura anterior son similares a los ya explicados con la furgoneta.

usarCamion (Figura 4.52)

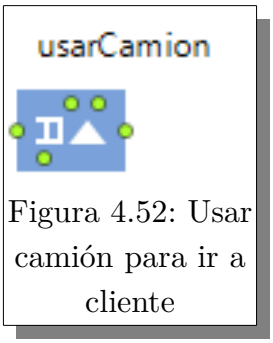


Figura 4.52: Usar camión para ir a cliente

Componente similar en todo al de la furgoneta salvo en la acción *Al captar unidad* que queda:

```
((Camion) unit).aCliente = agent.deCliente;
((Camion) unit).indx = agent.indx;
agent.deCliente.camion = ((Camion) unit);
```

con las dos últimas líneas añadidas con respecto a la furgoneta, indicando que se cargue en el camión la ruta del pedido y que se cargue en el camión del cliente del pedido el mismo camión. Estos valores se usarán en el desplazamiento al siguiente camión explicado en el Apéndice 1.

entregando (Figura 4.53)

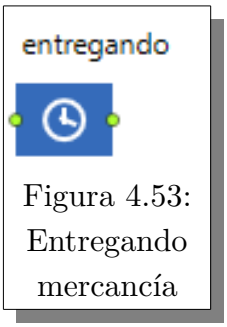


Figura 4.53: Entregando mercancía

Es similar al componente de la furgoneta, salvo la acción *onExit* que contiene:

```
send("recibidoDiario", agent.deCliente);
```

para que el cliente cambie de estado en el diagrama de estados.

entregado (Figura 4.54)

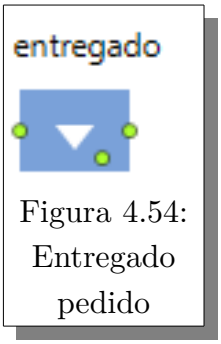


Figura 4.54: Entregado pedido

También es similar al de la furgoneta, salvo que el recurso *se queda donde está* en lugar *regresar a ubicación origen*.

marcaPedidoComoEntregado (Figura 4.55)

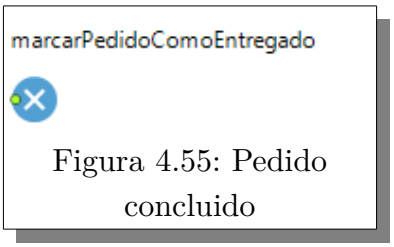


Figura 4.55: Pedido concluido

Similar a la furgoneta.

volverACentral, finDeJornada (Figura 4.56)

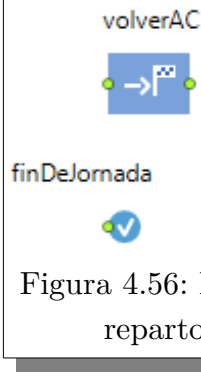


Figura 4.56: Fin de reparto

Similar a los componentes finales de la furgoneta, salvo la acción *onExit* de *volverACentral* que queda:

```
repedirPedidosPendientes(agent);
```

lanzando el método descrito en el Apéndice 1.

firmaEntrega, aDistribuir (Figura 4.57)

Figura 4.57: Firma Entrega

Controla la firma de la entrega por parte del cliente antes de realizar el siguiente paso de distribución.

jornadaTerminada (Figura 4.58)

Figura 4.58: ¿Fin de Jornada?

Este componente, según la Ayuda de Anylogic *Dirige el agente entrante a uno de los dos puertos de salida dependiendo de una condición expresada en condition.*

En este caso, la condición es *Jornada Terminada* (!isRepartir). Si es verdadero, sale por el puerto de la derecha. Si es falso, por el inferior.

quedaMercancia (Figura 4.59)

Figura 4.59: ¿Siguiente Pedido?

En este caso, la condición es *No queda mercancía que repartir*

```
!getNextPedidoReparto(agent.indx, agent);
```

Saliendo por verdadero en caso de no quedar y por falso si queda.

aSiguientePedido,yendoACliente,EntregandoPedido (Figura 4.60)

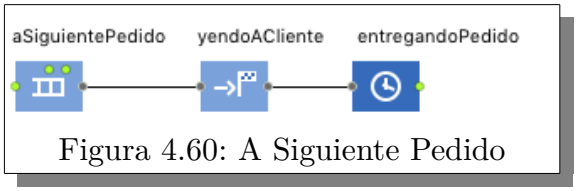


Figura 4.60: A Siguiente Pedido

Estos tres componentes realizan el proceso de reparto al siguiente cliente de la Ruta. No es necesario volver a la central, como hacen las furgonetas, ya que el camión dispone del destino del siguiente cliente.

El componente *yendoACliente* contiene el agente destino del camión, en este caso, *agent.aCliente*.

El componente *entregandoPedido* contiene en la acción de salida *send("recibidoDiario", agent.aCliente);* que es la que desactiva el "esperando Pedido" del cliente.

Por lo tanto, este proceso final queda como se puede observar en la Figura 4.61:

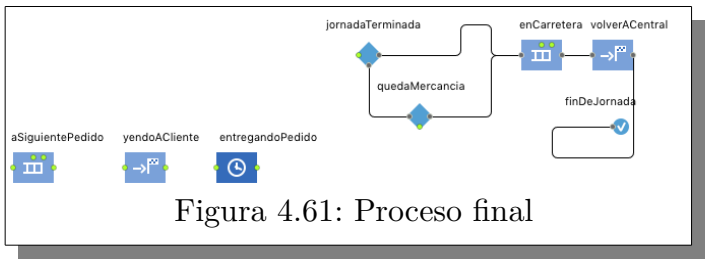


Figura 4.61: Proceso final

Estos componentes indican que, antes de continuar el camión con el siguiente pedido:

- comprobar si se ha acabado la jornada laboral
- en caso afirmativo, ir a Central
- en caso negativo, comprobar si queda mercancía
- si no queda, ir a Central
- si queda, ir a siguiente pedido

4.2.12 Creando la Sucursal de la Central

Este caso es completamente similar al de la Central.

- Creo el agente del tipo Sucursal
- Creo un punto GIS con su ubicación en el sur de la isla

- Se la asigno a la sucursal.

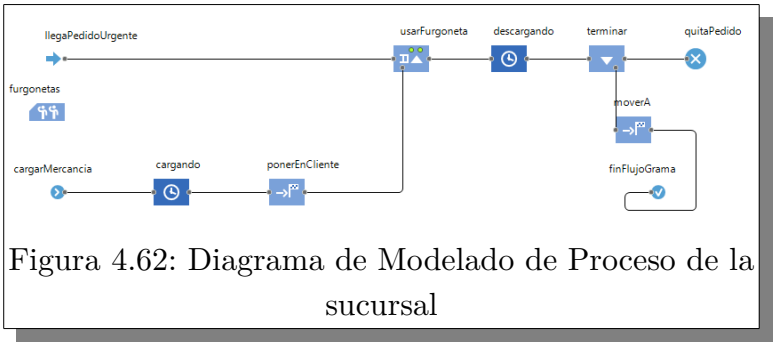


Figura 4.62: Diagrama de Modelado de Proceso de la sucursal

Este diagrama (Figura 4.62) es gemelo del usado para la central. De hecho, se puede gestionar todo lo que éste hace añadiendo un *switch* al de la central que controle si está resolviendo la Central o la Sucursal y haciendo que los componentes lo contemplen, pero no queda tan claro el diseño.

Las diferencias son:

- El número de furgonetas en el ResourcePool *furgonetas* se extrae de `main.cantidadFurgonetasSur.`
- *moverA* se desplaza a main.sucursal tras terminar la entrega.
- *Connections* contiene `llegaPedidoUrgente.take(msg);`

Los clientes siguen haciendo pedidos a la central que actúa de discriminador de éstos, transmitiendo a la Sucursal aquellos que van a ser tramitados por ella y no por la central.

Para ello, el componente *connections* se encarga de discriminar los pedidos con el siguiente código:

```
if (msg.isUrgente) {
 if ("N2".equalsIgnoreCase(msg.deCliente.zone))
 send(msg,main.sucursal);
 else
 llegaPedidoUrgente.take(msg);
} else {
 addPedidoDiario(msg);
}
```

Como la sucursal se encuentra ubicada en el Sur, la zona que es atendida por ésta es la N2.

Todo lo que llegue a la central perteneciente a clientes de la zona N2 es enviado a la Sucursal para que lo gestione.

Capítulo 5
Experimentando

En este capítulo se va explicar con detalle la simulación estándar de Anylogic, que va a servir de base a los experimentos que se pueden ver en los apartados 5.1.1, 5.1.2 y 5.1.3 con los que se va a da respuesta al número óptimo de furgonetas para la nueva sucursal a abrir en el Sur, las óptimas para la Central y el número óptimo de camiones de la empresa.

Se ha seleccionado una muestra de 196 clientes.

Para pedidos urgentes, 52 son atendidos por la sucursal del Sur y el resto, es decir, 144 son atendidos por la sede Central.

Para pedidos no urgentes, todos los clientes don atendidos por la Sede Central.

5.1 Simulación estándar

Anylogic viene de serie con un experimento estándar ya creado que permite visualizar la interacción entre los diversos agentes y componentes que contiene el proyecto.

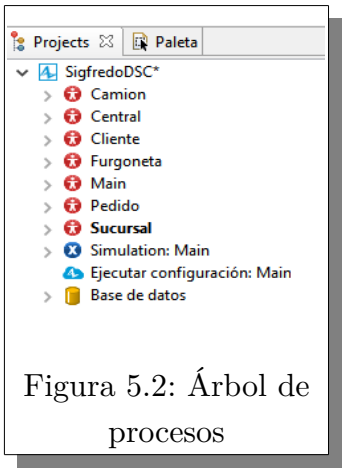


Figura 5.2: Árbol de procesos

Para lanzarlo, pulsar sobre *Simulation: Main* en el árbol de procesos (Figura 5.2).

Esto hará que aparezcan las propiedades del Simulador donde podremos cambiar, entre otras, las

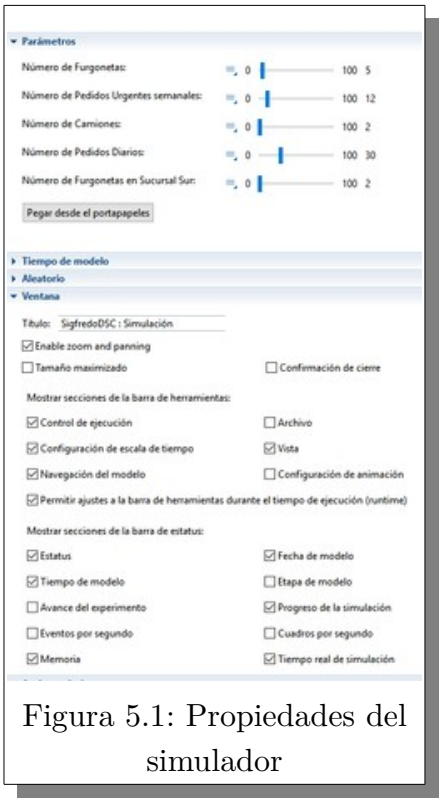


Figura 5.1: Propiedades del simulador

mostradas en la Figura 5.1 referentes a parámetros de nuestro agente *Main* y algunas de las opciones de la Ventana del experimento.

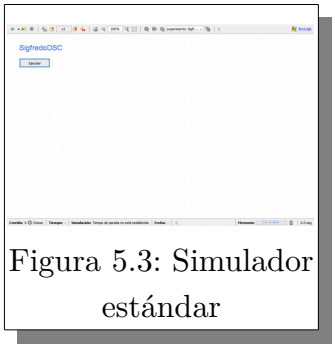


Figura 5.3: Simulador estándar

Haciendo doble click sobre la rama *Simulation: Main* o pulsando F5, compilamos el proyecto y lanzamos el Simulador, apareciendo la ventana del la Figura 5.3:

Pulsando sobre el botón *Ejecutar*, ponemos el Simulador en funcionamiento, apareciendo el mapa de Tenerife con los vehículos desplazándose por las diversas carreteras hasta llegar a los clientes que han realizado pedidos y entregando éstos, tanto urgentes como diarios.

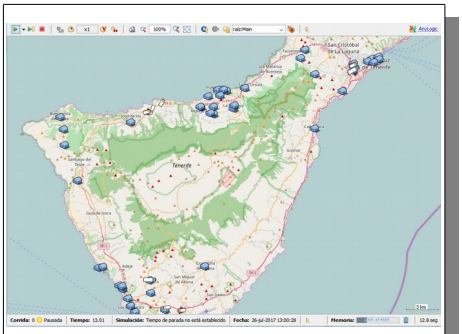


Figura 5.4: Simulador en funcionamiento

Con este simple simulador, indicado para visualizar si el proyecto se ejecuta adecuadamente, se es capaz de detectar si los clientes se encuentran ubicados en la zona correcta ya que si el reparto de camiones realiza zigzag entre las zonas de distribución, se está indicando que no están ubicados en la zona adecuada.

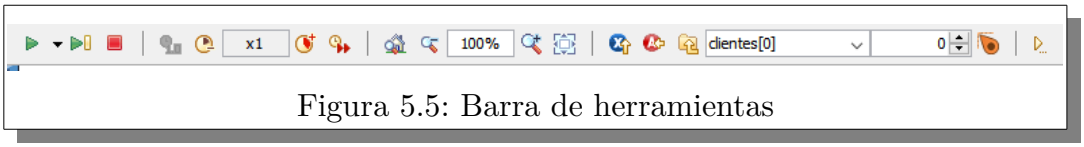


Figura 5.5: Barra de herramientas

En la barra de herramientas de la Figura 5.5, podemos usar el botón de la izquierda para pausar o reanudar el proyecto, el siguiente para ejecutarlo paso a paso y el tercero para detenerlo.

Los botones de la Figura 5.6 permiten acelerar la ejecución del proceso o ralentizarlo a voluntad.

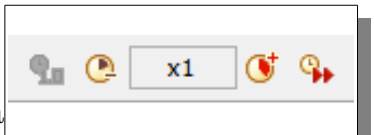


Figura 5.6: Botones de control de tiempo

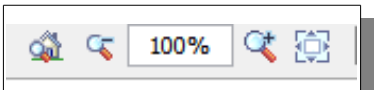


Figura 5.7: Botones de tamaño

Con los botones de la Figura 5.7, se puede controlar el tamaño del contenido de la ventana.

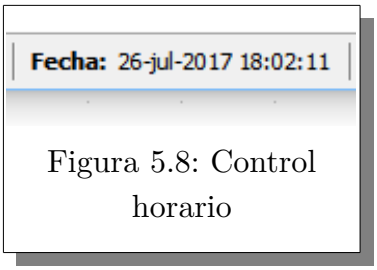


Figura 5.8: Control horario

La Figura 5.8 se puede visualizar en la barra de estado de la parte inferior del simulador. Con ella vemos como avanza el tiempo en este proceso.

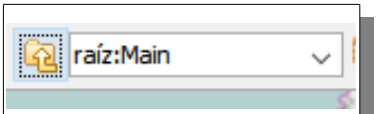


Figura 5.9: Selector de Agente

Pulsando en *Raíz:Main* de la Caja combinada de la barra de herramientas (Figura 5.9), podemos cambiar el contenido al agente que nos interese, por ejemplo a c*entral*, lo que hace que se muestre la Figura 5.10 con el *diagrama de Modelado del Proceso*.

En él, podemos apreciar los diversos componentes con la cantidad de entradas y/o salidas de cada uno y la relación entre ellos y observar como varía el tiempo de la barra de estado haciendo que a las 17:00 horas se acabe la jornada laboral y los camiones vuelvan a la central.

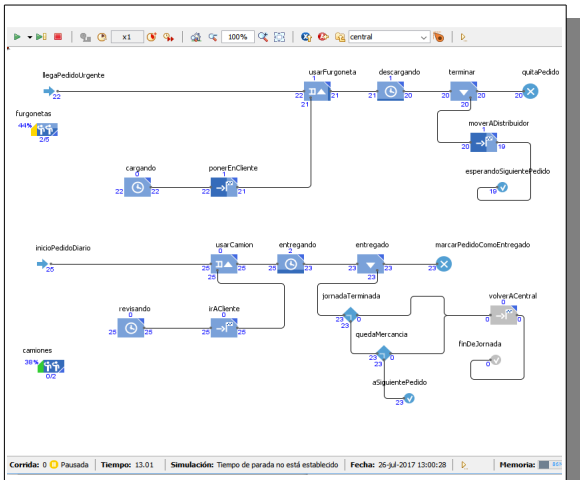


Figura 5.10: Modelado del proceso de la Central en funcionamiento

Si volvemos atrás, pulsando la flecha amarilla a la izquierda de *Central* (Figura 5.10) y seleccionamos clientes, al ser una lista, nos aparece el primer cliente de la misma, pudiendo movernos al siguiente o anterior.

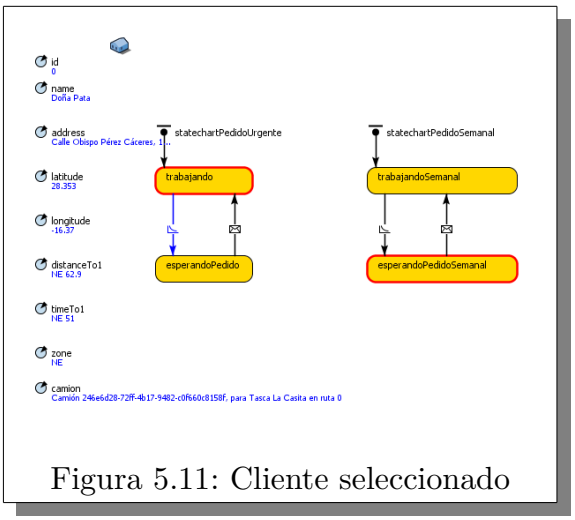


Figura 5.11: Cliente seleccionado

En esta ventana (Figura 5.11) podemos observar el *Diagrama de Estados* que tiene cada cliente del proyecto, así como las propiedades de cada uno, es decir, su nombre, dirección, coordenadas, etc.

5.1.1 Optimización de furgonetas de sucursal

Sigfredo está pensando en aumentar la flota de furgonetas actual que tiene en la sucursal del sur de la isla, previendo un posible incremento de la demanda y le gustaría saber si es realmente necesario este desembolso en la adquisición de nuevas furgonetas y bajo qué circustancias o, por el contrario, es suficiente el número de furgonetas actuales.

Mediante este experimento de Optimización se va a comprobar el número óptimo de furgonetas entre 1 y 6 necesarias para atender la demanda de los clientes de la zona sur.

El experimento va consistir en fijar una función objetivo, en este caso el número de furgonetas utilizadas y aplicar una Maximización del número necesario de furgonetas para que su uso no sea inferior al 90%. No se pone 100% para así poder contemplar averías, mantenimiento, etc. de las mismas.

Para ello, se accede a la creación de nuevo experimento (Figura 5.13), lo llamamos OptimizaciónSucursalSur y seleccionamos *Optimización* en el tipo de experimento, pulsando luego la tecla *Finalizar*.

Y accedemos a las propiedades del experimento (Figura 5.12) con la rama del árbol ya marcada en la pestaña de la izquierda.

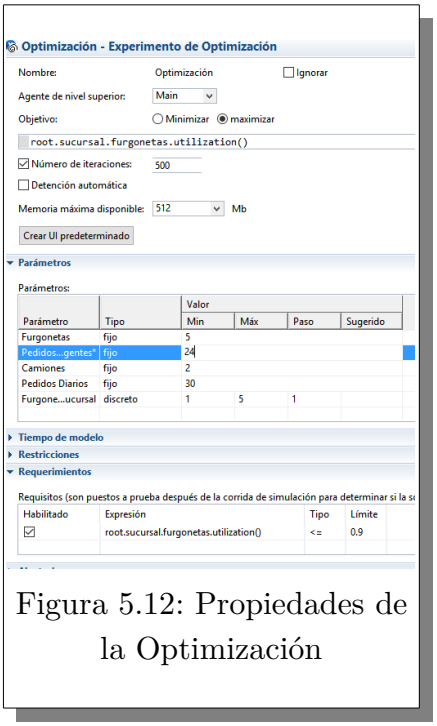


Figura 5.12: Propiedades de la Optimización

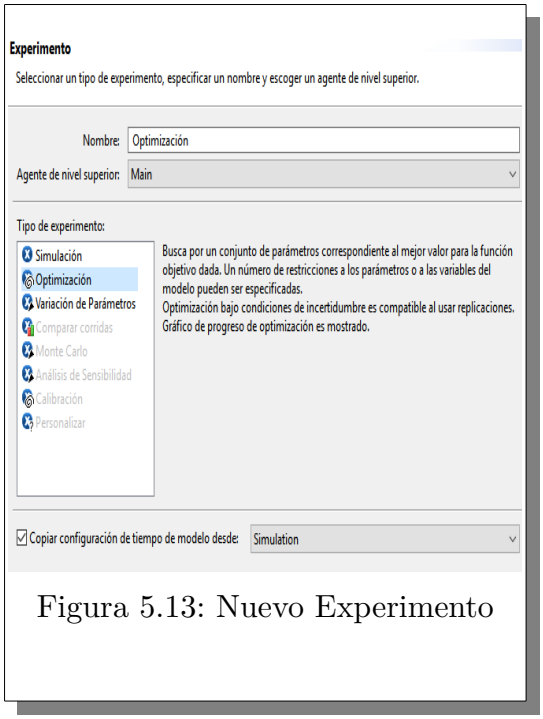


Figura 5.13: Nuevo Experimento

En este panel (Figura 5.12), indicamos el objetivo de la optimización con-

sistente en:

- Maximizar el número de furgonetas necesario
- Añadimos la función objetivo que se va a usar para evaluar:

  ```
  root.sucursal.furgonetas.utilization()
  ```

- En parámetros cambiamos los valores de *Furgonetas Sucursal* a
  - Tipo discreto
  - Valor Mín a 1 furgoneta
  - Valor Máx a 6 furgonetas
  - En Paso lo dejamos a 1 furgoneta cada vez.
- En parámetros, también, cambiamos el número de Pedidos Urgentes a un valor de 2, que es el máximo del intervalo semanal por cliente, partiendo de cero, que se va a usar.
- En requerimientos, indicamos que la utilización de furgonetas no puede ser inferior al 90%. Para ello:
  - añadimos la función anterior

    ```
    root.sucursal.furgonetas.utilization()
    ```

  - siendo su Tipo <=
  - y el límite 0.9
- y pulsamos el botón de *Crear UI predeterminado,* apareciendo el prediseño del experimento en la rejilla del panel central (Figura 5.14).

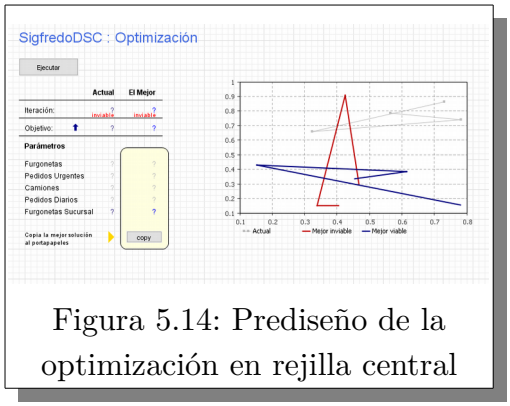


Figura 5.14: Prediseño de la optimización en rejilla central

Si seleccionamos la optimización en el botón *Play de la barra de herramientas* y en la ventana que aparece, tras compilar el proyecto, pulsamos el botón *Ejecutar*, el experimento se pone en marcha, dando como resultado la optimización planteada.

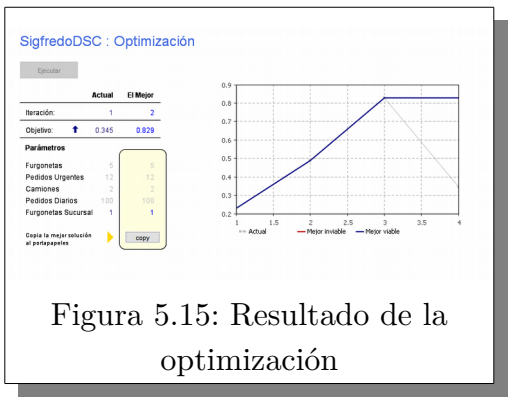


Figura 5.15: Resultado de la optimización

En este resultado (Figura 5.15), observamos que:

- con el 70,4% de utilización
- con una sola furgoneta es suficiente
- siempre que los pedidos urgentes estén entre 0 y 2 semanales (Parámetro establecido antes de lanzar el experimento).

Luego, observado lo anterior, no sería necesaria la ampliación de furgonetas en la sede del sur. No obstante, dependiendo de la cantidad de pedidos realizados por los clientes, la cosa puede cambiar... Para ello se adjunta la tabla 5.1.

Número de Pedidos semanales por Cliente	Furgonetas necesarias	Porcentaje de uso <= 90%
[0,2]	1	70,4
[0,3]	1	87,3
[0,4]	2	71,7
[0,5]	2	81,4

Tabla 5.1: Furgonetas necesarias en sucursal

En ella, se puede observar la media de pedidos semanales por cliente, el número de furgonetas necesarias y en que porcentaje se utilizan.

Baste, como explicación, la segunda fila, en la cual vemos que para una media de entre cero y tres pedidos semanales por cliente, con una furgoneta sería suficiente, usando ésta a un 87,3%.

Realmente, viendo esta tabla, una furgoneta es suficiente ya que hasta 3 pedidos urgentes semanales por cliente se acerca bastante a lo que Sigfredo nos ha indicado (off the record), pero tiene que ser él quien decida.

5.1.2 Optimización de furgonetas de central

El proceso es similar al del apartado anterior. La única diferencia estriba en el número de clientes asignados. La tabla 5.2 indica los valores para este caso, asignando un cantidad de furgonetas entre 1 y 15 para realizar el experimento. En ella, se puede apreciar que 4 furgonetas podrían bastarle para atender a los clientes urgentes.

Número de Pedidos semanales por Cliente	Furgonetas necesarias	Porcentaje de uso <= 90%
[0,2]	3	67,3
[0,3]	4	77,9
[0,4]	5	82,8
[0,5]	6	85,8

Tabla 5.2: Furgonetas necesarias en central

5.1.3 Optimizar camiones de la empresa

Sigfredo desea saber si con los camiones actuales sería capaz de abastecer a los clientes teniendo en cuenta determinada cantidad de pedidos diarios.

Al igual que en el apartado anterior, crearemos un experimento de optimización para el cual vamos a fijar una función objetivo, en este caso el número de camiones utilizados y aplicaremos una Maximización del número necesario de camiones en un intervalo entre 2 y 30 para ver cual es el número óptimo de camiones que nos propone el experimento dependiendo del número de pedidos diarios.

Para ello, se accede a la creación de nuevo experimento (Figura 5.16), lo llamamos OptimizarCamiones y seleccionamos *Optimización* en el tipo de experimento, pulsando luego la tecla *Finalizar*.

Y accedemos a las propiedades del experimento (Figura 5.17) con la rama del árbol ya marcada en la pestaña del panel izquierdo.

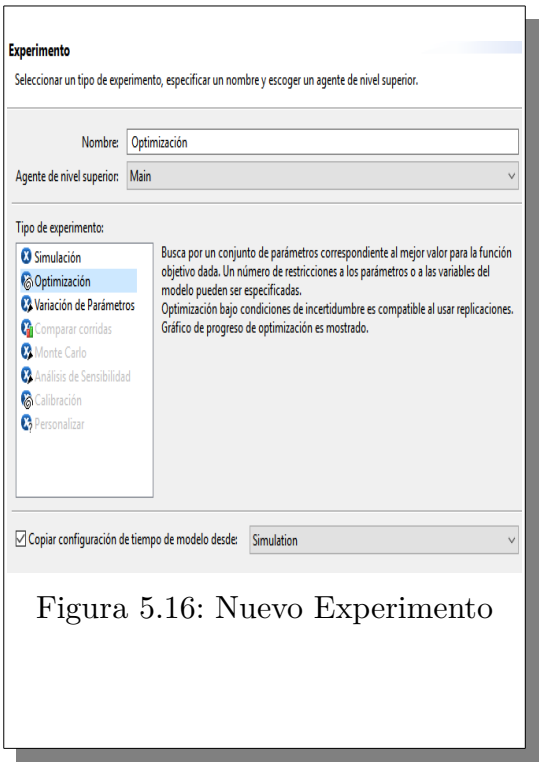


Figura 5.16: Nuevo Experimento

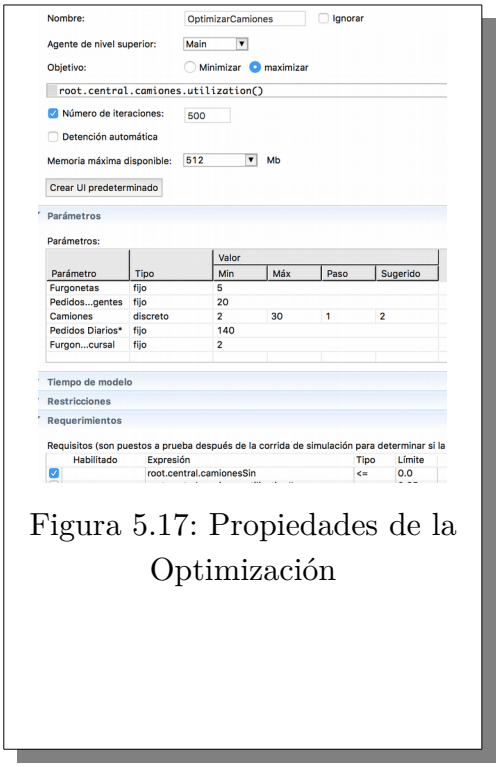


Figura 5.17: Propiedades de la Optimización

En este panel (Figura 5.17), indicamos el objetivo de la optimización consistente en:

- Maximizar el número de camiones necesario que, en este caso, será cuanto menor, mejor.
- Añadimos la función objetivo que se va a usar para evaluar:

  `root.central.camiones.utilization()`

- En parámetros cambiamos los valores de *Camiones* a
  - Tipo discreto
  - Valor Mín a 2 camiones
  - Valor Máx a 30 camiones
  - En Paso lo dejamos a 1 camión de incremento en cada iteración.
- En requerimientos, indicamos que el número de camiones con Pedidos sin entregar, una vez concluida la jornada, es cero. Para ello:
  - añadimos el parámetro

    `root.central.camionesSin`

    este parámetro se inicializa en el método *cargarCamiones* (Ver

Apéndice 1) y se incrementa en método *repedirPedidosPendientes*.

- siendo su Tipo <=
- y el límite 0.0

y, al igual que en experimento anterior, pulsamos el botón de *Crear UI predeterminado,* apareciendo el prediseño del experimento en la rejilla del panel central (Figura 5.18).

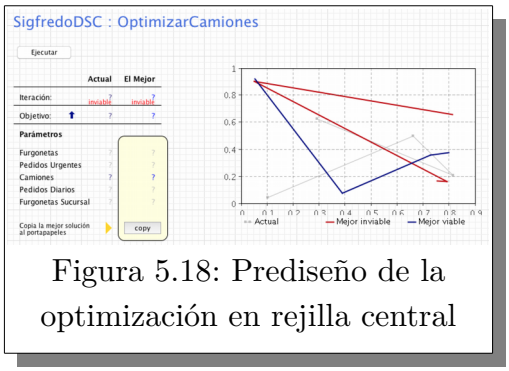


Figura 5.18: Prediseño de la optimización en rejilla central

Si seleccionamos la optimización en el botón *Play de la barra de herramientas* y en la ventana que aparece, tras compilar el proyecto, pulsamos el botón *Ejecutar*, el experimento se pone en marcha, dando como resultado la optimización planteada.

En este resultado (Figura 5.19), observamos que:

- con (1 – 0,312) * 100% = 68,8%
- con 140 pedidos diarios
- son necesarios 14 camiones para realizar el reparto y que no quede mercancía sin entregar.

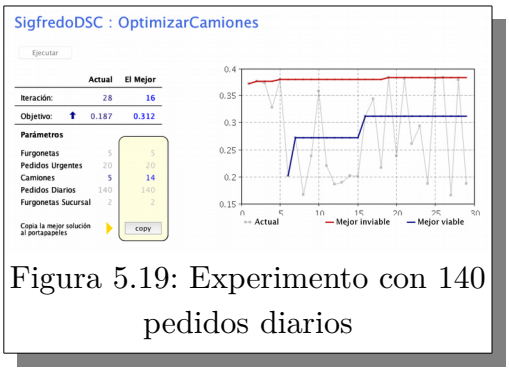


Figura 5.19: Experimento con 140 pedidos diarios

Para ver los distintos camiones necesarios en base a una suposición de pedidos diarios, se adjunta la Tabla 5.3 siguiente en la cual se han realizado las optimizaciones necesarias cambiando el número de pedidos diarios a un intervalo supuesto de 2 a 30 camiones.

Número de camiones	Número de Pedidos	Camiones con Pedidos sin entregar	Camiones necesarios
[2,30]	20	0	13
	40		14
	60		15
	80		14
	100		14
	120		13
	140		14
	160		14
	180		15
	200		15

Tabla 5.3: Camiones necesarios

Capítulo 6
Conclusiones y líneas futuras

6.1 Conclusiones

El objetivo de este trabajo ha consistido en desarrollar, mediante una herramienta de simulación, un entorno similar al disponible en la empresa Sigfredo Melchor, SL, para así poder llevar a cabo una serie de experimentos en su cadena de distribución y extraer conclusiones que sirvan de guía para futuras inversiones.

Se ha usado AnyLogic como herramienta para el diseño, desarrollo y ejecución de las simulaciones comprobando a lo largo de todo el proyecto su robustez y lo difícil que es conseguir que se produzca algún tipo de fallo.

Con este proyecto se ha conseguido utilizar en el simulador conceptos realmente caros de implementar en la vida real, tales como:

- Adquirir más vehículos.
- Abrir una sucursal nueva.
- Idoneidad de la ubicación de los clientes, sobre todo si la zona donde figuran es la más adecuada.
- Idoneidad de disponer de sólo una Central o si es necesario abrir alguna sucursal.

El "*qué pasa si*" queda fácilmente implementado en cada experimento que se realiza:

- Poner más coches es un proceso tan simple como hacer clic sobre una barra deslizadora: inversión cero.
- Para abrir una sucursal nueva sólo hay que crear un nuevo agente con

un tiempo relativamente rápido de implementación: inversión mínima comparado con la vida real. Y digo mínima suponiendo que el cliente no sabe desarrollar con esta herramienta y tiene que contratar a personal que lo haga.

- Cambiar a un cliente de ubicación se realiza simplemente con un desplazamiento en el plano. Incluso éste se puede ampliar hasta conseguir la posición exacta. Esto ahorra tiempo, dinero y, sobre todo, desplazamientos erróneos que tienen como consecuencia no poder entregar la mercancía, enfado del cliente e incluso su pérdida.
- Al poder jugar con sucursales nuevas, podemos ver su flujo en situaciones cercanas a la realidad con inversión mínima.
- Analizar si es mejor tener más o menos furgonetas que vehículos consiste en comparar unos con otros dando resultados de forma gráfica y comprensible sin necesidad de largos textos y tratados que, por otra parte, nadie se lee.

La facilidad con que se visualiza el uso de Rutas de Distribución hacen que sea totalmente recomendable para las empresas de hoy en día, teniendo en cuenta que es más barato invertir en este tipo de software que invertir en situaciones de las que desconocemos su resultado y que son infinitamente más caras que este producto.

Los experimentos realizados han permitido establecer las necesidades de la flota actual de Sigfredo e indicar si es necesario su ampliación, tanto en número de camiones como de furgonetas, objetivo de este trabajo.

6.2 Líneas futuras

A continuación, se relacionan las diversas mejoras que se pueden contemplar en este proyecto cara al futuro, de forma agrupada por conceptos.

6.2.1 Zonas de Reparto

- Si el camión en la Ruta de Zona Sur pasa a la Zona Noreste, distribuir en sentido inverso ya que el camión llegará antes a los clientes más alejados de la central que a los más cercanos y, tal y como está ahora, tras terminar en la Zona Sur, sigue por los clientes más cercanos a la central de la Zona Noreste.

- Dividir Zona Noreste en dos o tres.
- Asignar Zonas fijas de reparto a cada camión, si hay camiones suficientes para ello.

6.2.2 Experimentos de Simulación

- Cambiar central de sitio y ver si mejoran los tiempos de distribución y la cantidad de clientes atendidos.
- Usar más combinaciones de optimizaciones.
- Usar con otra línea de experimentos, siempre que la versión de Anylogic lo permita, tales como
  - Variación de parámetros
  - Comparar ejecuciones
  - Monte Carlo

6.2.3 Proyecto en general

- Añadir camiones a la sucursal.
- Añadir Fin de Jornada Laboral a las furgonetas
- Contemplar que al regresar, el camión se quede cargado con los pedidos no entregados y que sean su punto de partida del día siguiente, añadiendo los nuevos a continuación de éstos al iniciar la ruta.
- Contemplar que el camión siga cargado con los pedidos no entregados y que al día siguiente se reclasifique con los nuevos. Es decir que no sea necesario descargar el camión al llegar a la central con los pedidos no entregados.
- Para observar bajo rendimiento en la entrega en distancias similares a la central, dentro de la misma zona, contemplar decisión de siguiente cliente en base a distancia óptima desde donde se encuentre el camión una vez acabado de entregar un cliente.

Capítulo 7
Summary and Conclusions

7.1 Conclusions

The objective of this work has been to develop, through a simulation tool, an environment similar to the one available at Sigfredo Melchor, SL, in order to carry out a series of experiments in its distribution chain and draw conclusions that serve as guide for future investments.

AnyLogic has been used as a tool for the design, development and execution of the simulations checking throughout the project its robustness and how difficult it is to get some kind of failure.

With this project has been used in the simulator concepts really expensive to implement in real life, such as:

- Acquiring more vehicles.
- Open a new branch.
- Suitability of the clients' location, especially if the area where they are listed, is the most appropriate.
- Suitability to have only one central office or if it is necessary to open a branch.

The "what if" is easily implemented in each experiment that is performed:

- Putting more cars is as simple a process as clicking on a slide bar: zero inversion.

- To open a new branch you only have to create a new agent with a relatively fast implementation time: minimum investment compared to real life. And I say minimal assuming that the client does not know how to

develop using this tool and has to hire staff to do it.

- Switching to a new location client is simply done with a shift in the plane. Even this can be extended to get the exact position. This saves time, money and, above all, erroneous displacements that have the consequence of not being able to deliver the merchandise, anger of the customer and even its loss.

- By being able to play with new branches, we can see their flow in situations close to reality with minimal investment.

- Analyze whether it is better to have more or less vans than vehicles is to compare with each other giving results in a graphic and understandable way without the need for long texts and treatises that, on the other hand, nobody reads.

The ease for visualization of Distribution Routes makes it totally recommendable for the companies of today, considering that it is cheaper to invest in this type of software than to invest in situations of which we do not know its result and that are infinitely more expensive than this product.

The experiments carried out have made it possible to establish the needs of Sigfredo's current fleet and indicate if it is necessary to expand it, both in number of trucks and vans, the objective of this work.

7.2 Future lines

The following are the various improvements that can be contemplated in this project facing the future, in a grouped by concepts.

7.2.1 Distribution Zones

- If the truck on the South Zone Route passes to the Northeast Zone, distribute in the opposite direction, since the truck will arrive earlier to the clients that are farther from the center than to the closest ones and, as it is now, after finishing in the Southern Zone, is followed by the clients closest to the Central of the Northeast Zone.

- Divide Northeastern Zone into two or three.

- Assign fixed delivery zones to each truck, if there are enough trucks for it.

7.2.2 Simulation Experiments

- Change site center and see if they improve distribution times and the number of clients served.
- Use more combinations of optimizations.
- Use with another line of experiments, provided that the version of Anylogic allows, such as
    - Variation of parameters
    - Compare executions
    - Monte Carlo

7.2.3 Project in general

- Add trucks to the branch.
- Add End of Labor Day to the vans
- Contemplate that when returning, the truck will be loaded with the orders not delivered and that are their starting point of the following day, adding the new ones following these when starting the route.
- Contemplate that the truck is still loaded with orders not delivered and that the next day is reclassified with new ones. That is to say, it is not necessary to unload the truck when arriving at the plant with the orders not delivered.
- To observe low delivery performance at distances similar to the central one, within the same zone, contemplate next customer decision based on the optimal distance from where the truck is located once finished delivering a customer.

Capítulo 8
Presupuesto

Presupuesto de este Trabajo en Horas Empleadas (HE). Cada HE la valoro en 50€.

8.1 Anylogic

Estudio y aprendizaje de Anylogic:

- Realización de diversos cursos enfocados a control de Agentes y Características generales de la herramienta.
- Estudio de manuales con el fin de conseguir una formación adecuada antes del desarrollo del proyecto.
- Investigación de Casos de Uso para conseguir el dominio de la herramienta.
- Puesta en marcha de ejemplos estudiados.

Tipo	Descripción	HEs	Suma
Estudio Anylogic	Aprendizaje de AnyLogic mediante el estudio de varios manuales y la realización de Webinars	15	750,00 €
Investigación	Implementación de Casos Prácticos	35	1.750,00 €
Pruebas	Comprobación del correcto funcionamiento de los ejemplos realizados	6	300,00 €
	Total	**56**	**2.800,00 €**

Tabla 8.1: Anylogic

8.2 Desarrollo Base

Diseño e Implementación de módulo Base y las furgonetas

- Desarrollar Mapa GIS en la isla de Santa Cruz de Tenerife.
- Ubicar la Central distribuidora en el mapa.
- Ubicar los Clientes en el mapa.
- Crear Pedidos.
- Crear interacción entre los agentes.
- Puesta en marcha de todos los agentes.

Tipo	Descripción	HEs	Suma
Módulo Base	Desarrollo de Módulo Base para su posterior uso por los diversos agentes del proyecto	1	50,00 €

Tipo	Descripción	HEs	Suma
Investigación	Estudio de la forma correcta de implementarlo	15	750,00 €
Pruebas	Comprobación del correcto funcionamiento de los diversos agentes	7	350,00 €
	Total	**23**	**1.150,00 €**

Tabla 8.2: Desarrollo Base

8.3 Desarrollo de Furgonetas

Diseño e Implementación de las furgonetas.

- Creación de Furgonetas.
- Sincronizar con Módulo Base.
- Diagrama de Flujo de Modelado de Proceso.
- Puesta en marcha.

Tipo	Descripción	HEs	Suma
Furgonetas	Desarrollo de Furgonetas	10	500,00 €
Investigación	Estudio de la forma correcta de implementar diagrama de Flujo de Modelado del Proceso	5	250,00 €
Pruebas	Comprobación del correcto funcionamiento del agente	6	300,00 €
	Total	**21**	**1.050,00 €**

Tabla 8.3: Desarrollo Furgonetas

8.4 Desarrollo de Camiones

Diseño e Implementación de los camiones.

- Creación de Camiones.
- Sincronizar con Módulo Base.
- Diagrama de Flujo de Modelado de Proceso.
- Puesta en marcha.

Tipo	Descripción	HEs	Suma
Camiones	Desarrollo de Camiones	25	1.250,00 €
Investigación	Estudio de la forma correcta de implementar diagrama de Flujo de Modelado del Proceso	15	750,00 €
Pruebas	Comprobación del correcto funcionamiento del agente	12	600,00 €
	Total	**52**	**2.600,00 €**

Tabla 8.4: Desarrollo Camiones

8.5 Desarrollo de Experimento

Diseño e Implementación del Experimento.

- Creación del Experimento.
- Puesta en marcha.

Tipo	Descripción	HEs	Suma
Experimento	Desarrollo del Experimento	12	600,00 €
Investigación	Estudio de la forma correcta de implementarlo	3	150,00 €
Pruebas	Comprobación del correcto funcionamiento del experimento	2	100,00 €
	Total	**17**	**850,00 €**

Tabla 8.5: Desarrollo Furgonetas

8.6 Resumen

Suma de los diversos módulos del presupuesto.

Tipo	HEs	Suma
Anylogic	56	2.800,00 €
Módulo Base	23	1.150,00 €
Furgonetas	21	1.050,00 €
Camiones	52	2.600,00 €
Experimento	17	850,00 €
Total	**169**	**8.450,00 €**

Tabla 8.6: Resumen

Capítulo 9
Apéndice 1: Programación

9.1 Método para Añadir Pedido Diario de clientes

```
/**************************************************************************
*
* UBICACIÓN Agente Central, Java avanzado, Código de clase adicional
*
*      MÉTODO addPedidoDiario
* PARÁMETROS Pedido con pedido a añadir
* DEVUELVE nada
*
**************************************************************************
*
* AUTOR: Jesús Marín
*
* FECHA: 20/04/2017
*
* DESCRIPCIÓN
*
* Añade Pedido a ArrayList de pedidos diarios.
* Lo inserta de tal forma que los pedidos quedan clasificados
* por el campo distanceTo1 del cliente que hace el pedido.
*
**************************************************************************/

public synchronized void addPedidoDiario(Pedido pedido) {
  if (pedidosDiarios.isEmpty())
      pedidosDiarios.add(pedido);
```

```
      else {
        boolean isDone = false;
        for (int i=0; i<pedidosDiarios.size(); i++) {
        if (pedidosDiarios.get(i).deCliente.distanceTo1
              .compareTo(pedido.deCliente.distanceTo1) == 0) {
        if (pedidosDiarios.get(i).deCliente.name.equals(pedido.deCliente.name)) {
          isDone = true;
          break;
        }
      }
      if (pedidosDiarios.get(i).deCliente.distanceTo1
              .compareTo(pedido.deCliente.distanceTo1) >= 0) {
        pedidosDiarios.add(i, pedido);
        isDone = true;
        break;
      }
    }
    if (!isDone)
      pedidosDiarios.add(pedido);
    }
}
```

9.2 Método para Cargar Camiones con Ruta

```
/*****************************************************************
*
* UBICACIÓN Agente Central, Java avanzado, Código de clase adicional
*
*      MÉTODO cargarCamiones
* PARÁMETROS ninguno
* DEVUELVE nada
*
*****************************************************************
*
```

```
* AUTOR: Jesús Marín
*
* FECHA: 22/04/2017
*
* DESCRIPCIÓN
*
* Carga Arraylist pedidosXCamion con la
* "n mercancía pedida a la central" / número de camiones
*
*****************************************************************************/
public void cargarCamiones() {
 pedidosNoEntregados.clear();
 camionesSin = 0;
 if (!pedidosDiarios.isEmpty()) {
   int amount = pedidosDiarios.size() / main.cantidadCamiones;
   for (int i=0; i<main.cantidadCamiones; i++) {
     ArrayList newArray = new ArrayList();
     for (int j=0; j < amount; j++) {
       Pedido pedido = pedidosDiarios.remove(0);
       pedido.indx = i;
       newArray.add(pedido);
     }
     if (pedidosXCamion.size() > i) {
       ArrayList oldArray = pedidosXCamion.get(i);
       oldArray.removeAll(newArray);
       oldArray.addAll(newArray);
       pedidosXCamion.remove(i);
       pedidosXCamion.add(i, oldArray);
     } else {
       pedidosXCamion.add(i, newArray);
     }
   }
   isRepartir = true;
 }
}
```

9.3 Método para Inicializar camiones con reparto

```
/*************************************************************************
*
* UBICACIÓN Agente Central, Java avanzado, Código de clase adicional
*
*      MÉTODO iniciarCamionesReparto
* PARÁMETROS ninguno
* DEVUELVE nada
*
*************************************************************************
*
* AUTOR: Jesús Marín
*
* FECHA: 15/05/2017
*
* DESCRIPCIÓN
*
* Iniciar reparto usando camiones, cada uno con su ruta ya asignada.
*
*************************************************************************/

public void iniciarCamionesReparto() {
 for (int i=0; i< main.cantidadCamiones; i++) {
   inicioPedidoDiario.take((Pedido) pedidosXCamion.get(i).remove(0));
 }
}
```

9.4 Método que Obtiene Siguiente Pedido a repartir

```
/*************************************************************************
*
* UBICACIÓN Agente Central, Java avanzado, Código de clase adicional
*
*      MÉTODO getNextPedidoReparto
* PARÁMETROS from con camión a procesar = índice del array
* DEVUELVE nada
*
```

```
*************************************************************************
*
* AUTOR: Jesús Marín
*
* FECHA: 15/06/2017
*
* DESCRIPCIÓN
*
* Iniciar reparto usando camiones, cada uno con su ruta ya asignada.
*
*************************************************************************/

public void getNextPedidoReparto(int from, Camion camion) {
 boolean isQuedaReparto = (pedidosXCamion.get(from).size() > 0);
 if (isQuedaReparto) {
   camion.aCliente = ((Pedido) pedidosXCamion.get(from).remove(0)).deCliente;
 }
 return isQuedaReparto;
}
```

9.5 Método que Repone Pedidos en Camión

```
/*************************************************************************
*
* UBICACIÓN Agente Central, Java avanzado, Código de clase adicional
*
*      MÉTODO repedirPedidosPendientes
* PARÁMETROS Camion con camión a procesar=índice del array=camion.indx
* DEVUELVE nada
*
*************************************************************************
*
* AUTOR: Jesús Marín
*
* FECHA: 10/07/2017
*
* DESCRIPCIÓN
*
```

```
* Vuelve a pedir pedidos que quedan en el camión.
* Es decir, todos los pedidos que quedan sin entregar en camión, se
* vuelven a colocar convenientemente en la cola de Pedidos pendientes
* para que vuelva a reprocesarse al día siguiente
*
**************************************************************************/

public synchronized void repedirPedidosPendientes(Camion camion) {
 int amount = pedidosXCamion.get(camion.indx).size();
 if (amount > 0)
   camionesSin++;
 for (int j=0; j < amount; j++) {
   pedidosNoEntregados.add(((Pedido) pedidosXCamion.get(camion.indx).get(0)));
   addPedidoDiario((Pedido) pedidosXCamion.get(camion.indx).remove(0));
 }
}
```

Bibliografía

Anylogic dispone, afortunadamente, de varios webinars, how-to y libros:

Los usados por mí principalmente son:

- Road Traffic Library Webinar en https://classic.anylogic.com/road-traffic-library-webinar
- Ilya Grigoriev. (2014). Anylogic 7 in three days, a quick course in simulation modeling. E-textbook. Anylogic.
- Ivanov D. (2016). Operations and Supply Chain Simulation with AnyLogic 7.2. Decision-oriented introductory notes for management students in master programs. E-textbook. Berlin: School of Economics and Law.
- Cálculo de distancias entre puntos GIS en https://www.google.es/maps
- AnyLogic GIS Example Part 1 en https://www.youtube.com/watch?v=57sYE4h4-40
- AnyLogic GIS Example Part 2 en https://www.youtube.com/watch?v=mw4rx0kb_Tw
- AnyLogic GIS Example Part 3 en https://www.youtube.com/watch?v=dVwkvb0oWcg&t=2s
- AnyLogic GIS Example Part 4 en https://www.youtube.com/watch?v=mIoBo2th1I8&t=23s
- AnyLogic GIS Example Part 5 en https://www.youtube.com/watch?v=O9oYk8ebFGY
- Statechart Interaction in AnyLogic [Agent-Based Modeling for Health Policy with AnyLogic] en https://www.youtube.com/watch?v=m30XLOAv_kk
- Building Interacting Statecharts in AnyLogic 7 en https://www.youtube.com/watch?v=uMWIZVDfgiQ
- AnyLogic Input from Excel Parameter File en https://www.youtube.com/watch?v=fO_AQV1fiQM

- AnyLogic 7.1 Demo Featuring GIS en https://www.youtube.com/watch?v=RmhW_wTGjqA
- Webinar sobre Delivery Fleet Optimization with GIS en https://www.youtube.com/watch?v=xSXUMX2HOL4&t=425s
- Manual pdf de Delivery Fleet Optimization with GIS en https://classic.anylogic.com/upload/vids/Webinar_Manual.pdf
- Ayuda en general de Anylogic en https://help.anylogic.com/index.jsp?topic=/com.xj.anylogic.help/html/gis/tutorial/Supply%20Chain%20GIS%20-%20Phase%203.html

Textos sobre Cadenas de Suministro:

- Sunil Chopra and Peter Meindl. (2006). Supply Chain Management. 3° Edition. Capítulo 1. Entender qué es la cadena de suministro. Pearson/Prentice Hall.
- David Blanchard. (2010), Supply Chain Management Best Practices, 2nd. Edition, John Wiley & Sons, ISBN 9780470531884
- James B. Ayers (2000) Handbook of Supply Chain Management, St. Lucie Press. ISBN 1574442732