Trabajo de Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Sistema de apoyo a la decisión para una fábrica multiproducto mediante simulación basada en agentes

Autor: Fernando Moreno Jiménez

Tutores: Antonio Lorenzo Espejo

Alejandro Escudero Santana

Dpto. de Organización Industrial y Gestión de Empresas II
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo de Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Sistema de apoyo a la decisión para una fábrica multiproducto mediante simulación basada en agentes

Autor:

Fernando Moreno Jiménez

Tutores:

Dr. Antonio Lorenzo Espejo – Profesor Sustituto Interino

Dr. Alejandro Escudero Santana - Catedrático de Universidad

Dpto. de Organización Industrial y Gestión de Empresas II

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2024

Proyecto Fin de Carrera: Sistema de apoyo a la decisión para una fábrica multiproducto mediante simulación basada en agentes

Autor: Fernando Moreno Jiménez

Tutor: Antonio Lorenzo Espejo –
Alejandro Escudero Santana

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

En primer lugar, quiero expresar mi profundo agradecimiento y dedicar este trabajo a mi familia. Han sido un pilar fundamental en toda mi trayectoria académica, siempre confiando en mí y ayudándome tanto a lo largo de estos años. En especial, agradecer a mis padres y abuelas, quienes día a día han estado a mi lado, sin ellos nada de esto hubiera sido posible. A mi abuela Maribel, a la que tengo presente cada día, sé que estaría muy orgullosa de mí y de todo mi esfuerzo.

Estos cuatro años en la ETSI han sido más que una etapa académica. Me han ayudado a entender la complejidad y a la importancia de la ingeniería. Han sido muchas horas de estudio, pero gracias a mis compañeros y mis amigos de la carrera, con los que he compartido tanto risas como desafíos y he crecido tanto académica como personalmente. Me no solo grandes conocimientos, sino también experiencias inolvidables y amigos que perduraran más allá de esta escuela.

Por último, mi agradecimiento a aquellos profesores que han compartido sus conocimientos y dedicación, esforzándose para que seamos grandes ingenieros y profesionales. En particular, a mi profesor y tutor Antonio Lorenzo Espejo, quien siempre ha sido muy cercano y agradable conmigo, su orientación y disposición constante para resolver mis dudas han sido claves para poder llevar a cabo este trabajo.

Gracias a todos aquellos que han estado conmigo estos años, por ser parte de esta etapa tan significativa para mí.

Fernando Moreno Jiménez

Sevilla, 2024

Resumen

La simulación basada en agentes es una técnica computacional que permite crear modelos de simulación compuestos por agentes. Estos pueden representar diferentes cosas como vehículos, personas con diferentes roles e ideas entre otros.

Para este trabajo se ha creado un modelo de simulación basada en agentes utilizando el software Anylogic. Su finalidad es servir como herramienta de ayuda a la decisión en una fábrica multiproducto. En concreto se trata de una fábrica de coches de Lego en la que se pueden producir hasta tres modelos diferentes con un total de diez estaciones de trabajo disponibles como máximo.

Este modelo virtual también podría ser utilizados en otras cadenas de producción multiproducto, obteniendo datos en tiempo real, anticipándonos al comportamiento del sistema y pudiendo realizar modificaciones en parámetros como la secuencia de montaje o la asignación de tiempos de producción de las distintas estaciones de trabajo, siempre en busca de la mejora de la productividad.

Una vez el usuario define el diseño de la planta y sus parámetros, los introduce en una Excel diseñada para facilitar la entrada de datos al modelo de Anylogic. Además, se realizan comprobaciones de las precedencias de montaje de los coches para verificar su viabilidad. Mediante un modelo cálculo, se agruparán todos los datos para importarlos automáticamente a la base de datos del modelo en Anylogic.

Al iniciar la simulación, se puede observar el funcionamiento de la fábrica gracias a una ventana de visualización 2D. Esto permite verificar si la disposición de las zonas de producción y el equilibrado es realmente posible. Los resultados obtenidos también permiten ver la eficiencia de dicha disposición e identificar posibles cuellos de botella.

Todos los datos relevantes obtenidos en la simulación se exportan a otra hoja Excel, para poder guardar una copia si se desea y así poder observar el comportamiento de la fábrica cambiando ciertos parámetros.

Abstract

Agent-based simulation is a computational technique that allows creating simulation models composed of agents, they can represent different things such as vehicles, people with dissimilar roles, ideas, among others.

For this work, an agent-based simulation model has been created using the Anylogic software, its purpose is to serve as a decision support tool in a multi-product factory. Specifically, it is a Lego car factory in which up to three different models can be produced and a total of ten workstations available at most.

This virtual model, could also be used in other multiproduct production chains, obtaining real time data, anticipating the behavior of the system and being able to make modifications to parameters such as the assembly sequence or the allocation of production times of the different stations always seeking to improve the productivity.

Once the user defines the plant design and its parameters, they enter them into an Excel designed to facilitate data entry into the Anylogic model. In addition, checks are conducted on the assembly precedence of the cars to verify their viability. Using a mathematical model, all the data will be grouped to be automatically imported into the model database in Anylogic.

When starting the simulation, you can observe the operation of the factory thanks to a 2D visualization window. This makes it possible to verify whether the layout of the production areas and the balancing is what is possible. The results obtained also allow us to see the efficiency of said arrangement and identify possible bottlenecks.

All the relevant data obtained in the simulation is exported to another Excel sheet, in order to save a copy if desired and thus be able to observe the behavior of the factory by changing certain parameters.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

La búsqueda constante de una alta productividad en el ámbito de la fabricación repercute considerablemente en todos los aspectos de los sistemas, desde el diseño de la planta hasta la gestión integral. En este contexto, la simulación se revela como una herramienta robusta que posibilita el modelado y la anticipación del comportamiento de sistemas complejos. Al crear modelos virtuales de la fábrica, se pueden explorar diversas disposiciones de planta con distintos escenarios y estrategias, permitiendo la observación en tiempo real de los resultados y la adaptación ágil de estrategias en función de los datos obtenidos.

Este estudio se centra específicamente en la implementación de un modelo de simulación basada en agentes mediante el uso del software AnyLogic. Diseñado con la finalidad de ayudar a la corrección de una práctica de la asignatura Sistemas Integrados de Producción del Grado en Ingeniería de Tecnologías Industriales impartido en la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Este modelo también podría ser utilizado en cadenas reales de producción que fabriquen simultáneamente hasta tres productos diferentes con hasta diez estaciones disponibles, ya que permite cambiar algunos parámetros como la secuencia de montaje, los tiempos de producción de cada una de las diez estaciones de trabajo, takt time e incluso un modo aleatorio que introduce cierta variabilidad controlada al proceso productivo. De igual forma, el modelo puede ser ampliado replicando los módulos correspondientes a las estaciones, para representar sistemas productivos más complejos.

Esta práctica trata de la construcción de un modelo LEGO 3 en 1, donde se configura un layout para la fabricación de tres modelos diferentes de automóviles. El principal objetivo es proporcionar a los estudiantes una experiencia práctica y aplicada en la planificación, organización de procesos de producción y configuración eficiente de una fábrica para la creación simultánea de varios modelos de coches.

Esta técnica computacional facilita el modelado de interacciones complejas entre agentes; cada agente

representa simbólicamente un coche que se ensambla a lo largo de diversas estaciones de trabajo disponibles en el layout diseñado por el alumno. Al simular este proceso de producción, el modelo generado puede ser una herramienta importante para la toma de decisiones, permitiendo optimizar el uso de recursos mediante ajustes en parámetros como la capacidad de cola en las estaciones, la asignación eficiente de piezas y el takt time, entre otros. Este enfoque no solo destaca la importancia de la simulación como un instrumento estratégico y de apoyo, sino que también subraya su papel esencial en la mejora continua de la eficiencia y la adaptabilidad en entornos de fabricación dinámicos.

Este trabajo consta de cinco capítulos. En primer lugar, el segundo capítulo presenta una revisión del estado del arte de temas como células de fabricación, simulación y sus aplicaciones, modelos de simulación, gamificación, así como una exploración de los softwares disponibles y sus funcionalidades, destacando especialmente AnyLogic. En el tercer capítulo, se explica todo lo relacionado con el modelo de simulación, incluyendo el funcionamiento del Excel con su cálculo matemático y la importación de datos desde Excel a AnyLogic. El cuarto capítulo comienza describiendo las herramientas del software utilizadas de manera general, detallando la ejecución de la simulación, la utilización de los bloques y todos los parámetros del modelo. También se presentan los resultados que se pueden obtener y cómo acceder a ellos tanto en AnyLogic como en un Excel creado para facilitar su visualización. En el quinto capítulo, se simula una prueba de validación del sistema con una solución diseñada manualmente. Se muestran los resultados obtenidos por el modelo de simulación y se explora cómo esta información puede ser de gran utilidad para identificar posibles mejoras.
Finalmente, se presenta una conclusión sobre el trabajo realizado.

2 ESTADO DEL ARTE

"Modeling for insights, not numbers"

- C. M. Macal -

Para crear un modelo de simulación basado en agentes y un layout de una fábrica multiproducto efectivo, es necesario adquirir conocimientos sobre su evolución, conceptos fundamentales y las diferentes herramientas de desarrollo disponibles. En este capítulo se explican dichos conceptos, tipos de simulación y se describen las características de diferentes softwares de simulación enfocándose principalmente en Anylogic.

2.1 Células de fabricación

La búsqueda continua por alcanzar la máxima productividad posible afecta en gran medida a todos los aspectos de los sistemas de fabricación como puede ser su diseño y su gestión. Para Chtourou (2008), el layout es uno de los principales aspectos a refinar para intentar alcanzar un alto nivel de rendimiento.

A mediados del siglo XX empezó a tomar fuerza el término Group Technology (GT), que fue introducido primeramente por Flanders (1925). Se basa en agrupar contenidos para situaciones que tengan las mismas decisiones, con criterios previamente seleccionados y compartidos en común.

La implementación en un entorno de fabricación del concepto GT es la fabricación en células (cellular manufacturing, CM), en la que los artículos que tengan un diseño o características de fabricación similares se agrupan (Süer. 1995) . Estas características pueden estar relacionadas con la forma, tamaño, propiedades de los materiales utilizados o incluso su proceso de fabricación.

La fabricación celular consiste en un taller que agrupa máquinas, trabajadores y herramientas para producir diferentes familias de piezas o productos. Esta agrupación física permite reducir tiempos de Setup, que son los tiempos necesarios para preparar y ajustar un equipo o sistema antes de comenzar a realizar una tarea especifica. Este tipo de distribución también facilita la transición entre distintos

productos de una misma familia y la producción en lotes, eliminando la rigidez de las líneas de producción tradicionales fomentando respuestas ágiles, mejora la eficiencia operativa y la optimización del flujo de trabajo entre otros beneficios. Askin (2013), comenta que son fáciles de aplicar en la práctica, la experiencia se adquiere rápidamente y las distancias de manipulación de los materiales se reducen.

Younes Sinaki (2022) agrupa los distintos parámetros necesarios para el diseño de sistemas de fabricación celular en tres etapas:

1. Formación de la célula: consiste en proporcionar la información necesaria para el proceso de fabricación, requisitos y la demanda para cada familia de piezas o productos para un conjunto dado de tipos de piezas/ productos. Las máquinas se agrupan para formar la fabricación de las células y las partes/ productos se asignan a las células (Papaioannou and Wilson 2010; Selim, Askin, and Vakharia 1998).
2. Planificación de la célula: basada en decisiones tácticas como por ejemplo la capacidad, inventario, asignación de la mano de obra y selección de procesos.
3. Programación de la célula: decisiones operativas y a corto plazo, incluyendo programación de la producción, decisión del tamaño de lote, secuenciación, elección de herramientas y su asignación. Además, recientemente se están incorporando y generalizando problemas de transporte, como podría ser el tamaño del lote y problemas de entrega (Süer, Mosadegh and Maihami 2018).

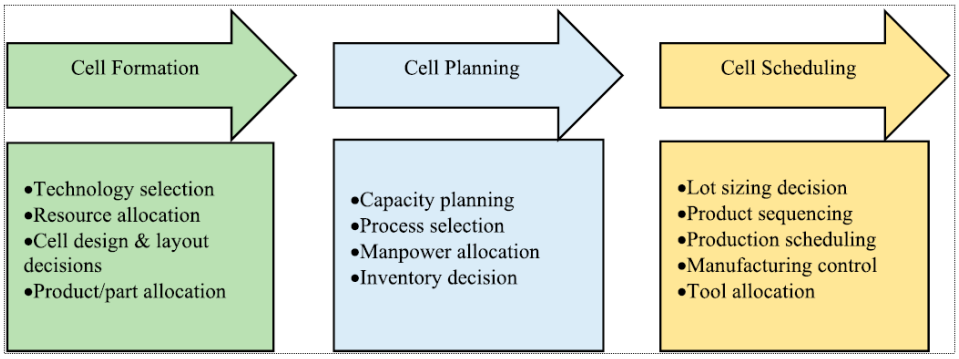

Figura 1. Problemas de fabricación celular en el horizonte de planificación (Younes Sinaki, 2022)

2.2 Simulación

La simulación puede ser considerada como una potente herramienta que permite diseñar y estudiar situaciones, sistemas y procesos complejos. Se posiciona como una metodología esencial para poder solventar numerosos problemas del mundo real, se emplea para describir y analizar el comportamiento de los sistemas, para plantear posibles interrogantes y casuísticas en relación con el sistema real, ayudando también a su diseño (Banks, 1999).

Para Shannon (1998), la simulación se define como "el proceso de diseñar un modelo de un sistema real y realizar experimentos con este modelo con el fin de comprender el comportamiento del sistema y/o evaluando diversas estrategias para el funcionamiento del sistema".

Estos experimentos consisten en aplicar un tratamiento específico a una parte aislada de la realidad, una vez concluido se observa que ha ocurrido y posteriormente se compara este objeto de análisis con otro que no ha recibido ningún tratamiento (generalmente llamado 'control').

Existen distintos paradigmas de simulación:

- Microsimulación: comienza con una base de datos representativa de agentes de una población sujeta a estudio. Este tipo de simulación crea modelos que no buscan explicar sino predecir, la ventaja que presenta respecto a otro tipo de simulaciones es que se comienza con una muestra de agentes reales (como podrían ser los resultados obtenidos de una encuesta), en vez de unos agentes hipotéticos o creados aleatoriamente (García-Valdecasas Medina, 2011).
- Dinámica de sistemas: desarrollada en los años 50 por Jay W. Forrester, destacado ingeniero del MIT, se caracteriza por un elevado nivel de abstracción con un enfoque determinista y un modelado continuo en sus términos temporales. Su funcionamiento está basado en la resolución numérica de ecuaciones diferenciales de primer orden.
- Simulaciones basadas en agentes: representación de sistemas que carecen de un comportamiento centralizado, formado por subsistemas que pueden ser tanto autónomos como colaborativos. Este enfoque imita la complejidad del mundo real, teniendo en cuenta características y comportamientos de estas entidades individuales llamadas 'agentes'(Mahdavi, 2020). Esta metodología permite estudiar como las interacciones de

estos agentes pueden dar lugar a patrones y resultados complejos en el sistema global.

- Simulación de eventos discretos: pueden ser tanto dinámicos como estocásticos, destacan por que los cambios ocurren solo en momentos discretos. En este tipo de simulación, la presencia de variables aleatorias como entradas implica también que las salidas serán aleatorias. Un ejemplo puede ser una simulación de un cajero de banco, en la que los tiempos entre llegada y de servicios son aleatorios (Sharma, 2015).

2.2.1 Agentes

Los agentes son módulos de los programas que representan a 'actores reales'. Entre los agentes virtuales y el mundo real, existe una relación para así facilitar tanto la interpretación como el diseño de los resultados en comparación con otros tipos de simulación (Gilbert, 2008: 14).

2.2.2 Eventos y actividades

Los eventos son sucesos instantáneos que cambian el estado de un modelo, según la clasificación de Carson (2005), quien los categoriza en primarios o secundarios.

Los primarios están impulsados por datos, como los tiempos entre llegadas y la finalización del proceso. Por otro lado, los secundarios son generados internamente por la lógica del modelo, como la cola de espera cuando una estación de producción está ocupada por otro agente.

Es importante diferenciar entre actividad y evento. Actividad se refiere a la duración de tiempo, como el intervalo entre llegadas de los agentes, iniciado por un evento en conjunto con el estado del modelo en un momento determinado.

2.2.3 Modelo de simulación

El término modelo implica la representación simbólica de conceptos para simplificar la comunicación humana. Estos símbolos, que pueden ser tanto palabras como objetos físicos o signos, constituyen representaciones simplificadas y parciales de algo más complejo.

Aunque los símbolos simplifican la comunicación, se limita la capacidad para definir por completo aquello que representan. Por otro lado, estos símbolos evitan el completo replanteamiento y se convierten en herramientas esenciales para tratar la información recibida. Los modelos más allá de la simple descripción representan el funcionamiento de algo (Mahdavi, 2020).

Según Carson (2005) un modelo no se limita a representar estáticamente un sistema o proceso, en el

contexto de la simulación abarca la estructura del sistema e incluye el tiempo y los cambios que se producen.

Dentro de los distintos tipos de modelo de simulación podemos encontrar:

- Determinísticos y estocásticos: los determinísticos no presentan ninguna aleatoriedad y trabajan bajo unas condiciones específicas. Los estocásticos incluyen elementos aleatorios, lo que puede generar diferentes resultados teniendo la misma configuración inicial debido a esta variabilidad

- Continuos y discretos: en los modelos continuos las variables cambian continuamente a lo largo del tiempo mientras que los discretos experimentan cambios en momentos específicos denominados tiempos de eventos. Estos últimos se fundamentan en conceptos de estados, actividades, eventos y procesos, destacando la relevancia crucial del tiempo en su funcionamiento.

2.2.4 Aplicación de la simulación

En la siguiente figura se muestran diferentes aplicaciones de simulación, todas ellas basadas en el nivel de abstracción de los correspondientes modelos:

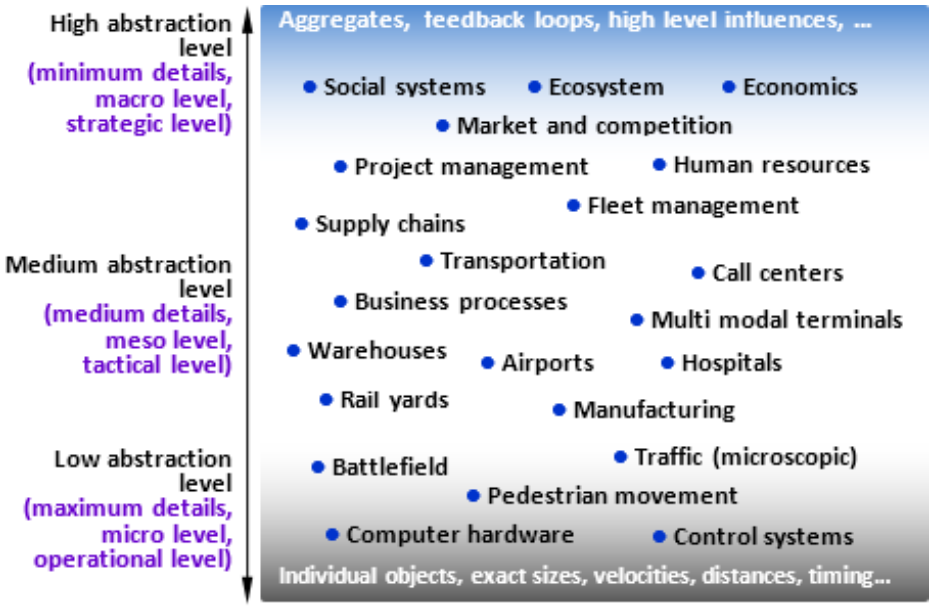

Figura 2: aplicaciones de simulación basada en la abstracción del modelo (Grigoryev, 2018)

En la zona inferior de la Figura 2 se pueden encontrar los modelos a nivel físicos, los cuales emplean representaciones con un alto nivel de detalles de objetos del mundo real. En este nivel nos enfocamos en la interacción física, como pueden ser las velocidades, las distancias y los intervalos de tiempos. Ejemplos de modelos de baja abstracción pueden ser los simuladores de vuelo, modelos de simulación de circuitos electrónicos y las acciones ejecutadas en un campo de batalla por los soldados.

Los modelos altamente abstractos son los superiores, normalmente se emplean agregados como pueden ser grupos de consumidores en lugar de objetos individuales. Debido a que estos objetos tienen un alto nivel de interacción, nos pueden ayudar a comprender relaciones, por ejemplo cómo afecta a las ventas de una empresa sus gastos en publicidad.

Otros modelos se sitúan en un nivel intermedio de abstracción, un ejemplo puede ser un diseño de la zona de urgencias de un hospital para calcular el desplazamiento del personal entre diferentes zonas de este.

Por último, los modelos con bajo nivel de abstracción suelen tener una gran cantidad de detalle como por ejemplo ocurre en los distintos elementos de hardware de un ordenador en el que tienen que ser muy precisos en las medidas, materiales, disposición, etc.

2.2.5 Ventajas y desventajas

Como se ha explicado anteriormente, la simulación permite la experimentación con un modelo de un sistema, identificar problemas y cuellos de botella, todo antes de construirlo y sin comprometer recursos e inversiones en un proyecto.

S. Carson (2005) recalca que no todo son ventajas, las simulaciones generalmente suelen necesitar una gran cantidad de tiempo y trabajo. Podemos encontrarnos que no se tienen todos los datos necesarios para poder realizar una simulación adecuada, que son muy costosos de conseguir o que el tiempo disponible antes de tomar una decisión no sea suficiente para sacar resultados concluyentes. Además, las animaciones y otro tipo de visualizaciones pueden llevar a tomar decisiones o conclusiones prematuras basadas en evidencias insuficientes.

Teniendo todo lo anterior en cuenta, la simulación sigue siendo una herramienta muy potente que puede proporcionar no solo resultados numéricos, sino también una profunda compresión del sistema. Por lo que, aunque pueda llegar a requerir mucho tiempo o dinero, su valor en nuevas ideas o

posibilidades suele ser superior al coste que supone.

2.3 Simulación basada en agentes

Se puede definir la simulación basada en agentes como una técnica computacional que permite, mediante la creación de modelos compuestos por agentes (entidades autónomas), la interacción entre ellos en un contexto determinado con el fin de llevar a cabo diversos experimentos virtuales.

También se puede definir como el proceso de diseñar un modelo de sistema real y llevar a cabo experimentos para comprender el comportamiento del sistema (Siebers , 2010)

La simulación basada en agentes puede representar diferentes cosas como pueden ser vehículos, personas con diferentes roles, ideas, proyectos, etc.

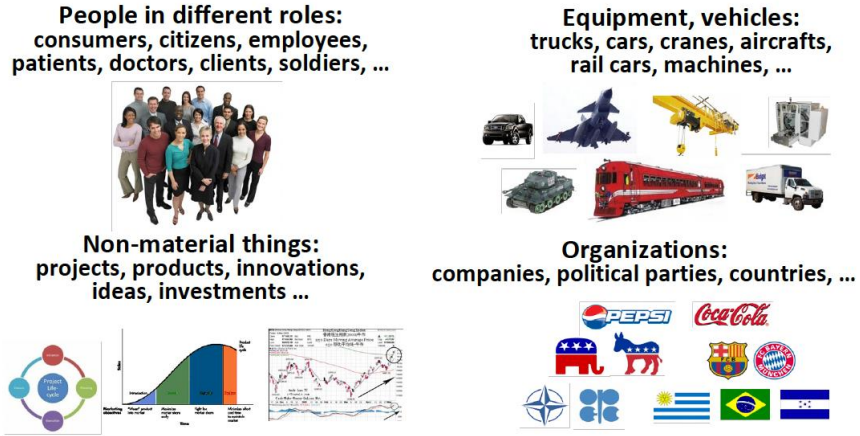

Figura 3: diferentes campos de aplicación de la simulación basada en agentes (Grigoryev, 2018)

2.3.1 Consideraciones de los agentes

Existen diferentes definiciones para el término agente dentro de la comunidad de ciencias de la computación.

Bandini y Manzoni (2012) los definen como entidades autónomas que tienen capacidad de decidir las acciones que va a realizar en el entorno y las interacciones que tienen con otros agentes.

En la investigación sobre modelos de interacción entre agentes, se exploran diferentes dimensiones y aspectos para establecer una taxonomía. Dentro de este contexto, se abordan dos enfoques distintos:

La interacción directa implica que los agentes intercambian mensajes sin considerar abstracciones del

canal de comunicación real. Esto significa que en el modelo no se incluyen detalles específicos sobre el medio o canal de comunicación utilizado en la realidad. Los modelos basados en Agent Communication Language (ACL) son ejemplos de este enfoque. Estos modelos presentan desafíos, ya que requieren que los agentes se conozcan mutuamente y tengan un nombre único en el sistema. Para superar esto, se emplean estrategias como el conocimiento previo entre los agentes o el uso de agentes intermedios.

Por otro lado, la interacción indirecta se caracteriza por la introducción de elementos intermediarios o la consideración del entorno como un factor influyente en la interacción entre agentes. Existen modelos que se centran en modelar el entorno del agente como el espacio donde ocurren las interacciones; este entorno influye en las interacciones y, por lo tanto, en el comportamiento de los agentes.

Comprender estas dimensiones en la interacción entre agentes es importante a la hora de desarrollar modelos que reflejen de manera precisa y efectiva las relaciones entre estos agentes.

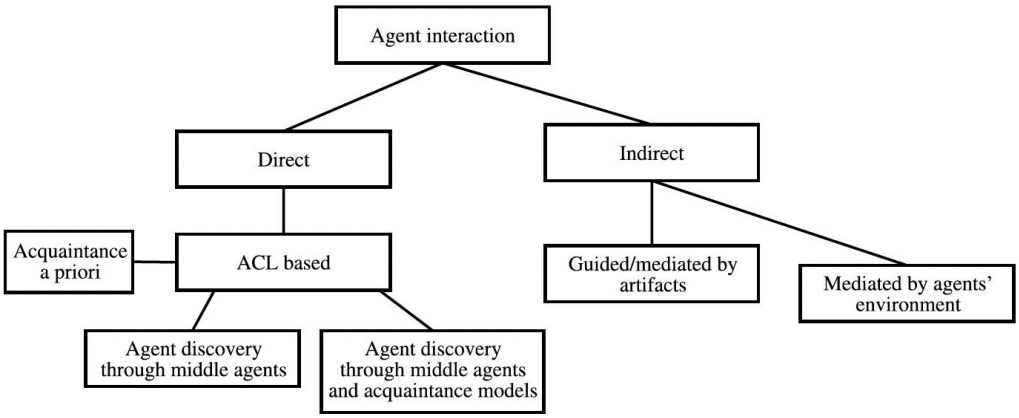

Figura 4: clasificación según interacción de los agentes (Bandini, 2009)

Por otro lado, Grigoryev (2018) recalca que, debido a la gran cantidad de modelos existentes, se pueden encontrar todo tipo de agentes, por lo que hay que tener siempre en cuenta las siguientes consideraciones respecto a estos:

- No son autómatas celulares, es decir, no necesitan vivir en un espacio discreto. Cuando se necesita representar este espacio suele ser continuo, como lo puede ser un mapa geográfico o un plano de instalaciones.

- No son necesariamente personas, pueden ser vehículos, equipos, proyectos, una organización...
- Un objeto que parece pasivo puede ser un agente. Como podría ser una pieza de un tren que se le asocia mantenimiento, coste y eventos relacionados con averías o reemplazos.
- Un modelo puede tener varios tipos de agentes.
- Existen modelos en los que los agentes no interactúan. Por ejemplo, algunos relacionados con la salud, uso de alcohol y enfermedades donde la dinámica individual solo depende de parámetros personales.

2.4 Software de simulación

Los programas informáticos conocidos como software de simulación permiten realizar simulaciones por computador en vez de actuar en un sistema físico real. Debido a los grandes desarrollos informáticos de los últimos años, se ha visto un notable desarrollo de distintos simuladores destinados a resolver una amplia variedad de problemas con diferentes alcances en un tiempo cada vez menor.

2.4.1 Vensim/Venapp

Software de alto nivel desarrollado por Ventana Systems Inc., destinado a mejorar el rendimiento de sistemas reales de numerosos ámbitos. Principalmente se utiliza para diseñar y analizar modelos de sistemas de realimentación dinámicos.

Originalmente fue desarrollado con el primitivo lenguaje de programación Pascal. En el 1988 fue traducido a lenguaje C y se implementó en el entorno gráfico de X-Windows. Con el paso del tiempo el software ha experimentado notables mejoras y actualizaciones, convirtiéndose en la actualidad en uno de los softwares más usados tanto en el ámbito industrial como el académico (Lorenzo-Espejo, 2018). Vensim permite elaborar fácilmente diagramas causales y diagramas de acumulación y flujo.

2.4.2 Arena

El software Arena es principalmente utilizado para modelar y simular sistemas de eventos discretos. Ofrece un entorno intuitivo que facilita la creación de los modelos. Estos modelos están compuestos por dos tipos de módulos: módulos de diagrama de flujo y módulos de datos.

Por otro lado, estos módulos de datos se presentan en forma de hoja de cálculo y se utilizan para complementar la información cuantitativa necesaria para la simulación como pueden, entre otros, los tiempos de proceso/Transporte, requisitos de recursos y cronogramas de procesos (Slota y Malopolski, 2007).

Los modelos se definen usando el lenguaje SIMAN y se lleva a cabo la simulación correspondiente. Posteriormente genera unos informes de simulación los cuales evalúan el desempeño del sistema en función de diferentes criterios. Adicionalmente el software ofrece otras aplicaciones adicionales que ayudan a definir datos de entradas (Input Analyzer), en la gestión de diferentes configuraciones del modelo, en el análisis de datos de salida (Output Analyzer) y en la búsqueda de la configuración óptima del modelo (OptQuest) (Pegden, 1990).

2.4.3 STELLA/iThink

iThink y Stella son dos marcas distintas de un mismo producto desarrollado por isee Systems.

Su distinción radica en el ámbito de aplicación: por un lado, STELLA se enfoca en el ámbito educativo y científico aportando incluso modelos de ejemplo y material como "An Introduction to Systems Thinking with STELLA" para ayudar a comprender el funcionamiento del software (isee systems inc, 2023).

Por otro lado, iThink está diseñado como una herramienta enfocada al mundo empresarial. Tiene otro tipo de modelos de ejemplo enfocados a planificación de recursos, finanzas e investigación operativa permitiendo la creación de aplicaciones visualmente llamativas y dinámicas basadas en modelos de simulación (J. Egne, 2011).

La característica más destacada de ambos softwares es la posibilidad de crear diagramas causales o de influencia, y diagramas de acumulación y flujo. Ofrecen una gran variedad de formatos para visualizar los resultados de la simulación, como gráficos, animaciones y tablas. Además, permiten el manejo de eventos discretos y proporcionan funcionalidades para el análisis de sensibilidad y métodos de optimización (isee systems inc, 2023).

Los programas de isee systems son recomendables para quienes deseen iniciarse al modelado y la simulación en dinámica de sistemas, debido a su relativa facilidad a la hora de aprender a utilizarlos y la gran cantidad de información y manuales disponibles incluso facilitados por la propia empresa.

2.4.4 Anylogic

AnyLogic es un software de simulación que permite la creación, simulación y análisis de sistemas complejos en diversas industrias. Desarrollada por The AnyLogic Company, esta herramienta es conocida por su habilidad para modelar sistemas a un nivel intermedio de abstracción (Medina , 2023).

Algo que distingue a Anylogic es que, además de las técnicas de simulación convencionales, también se pueden realizar simulaciones basadas en agentes. Esta técnica permite modelar sistemas compuestos

por entidades autónomas e interactivas, lo que es especialmente útil para representar sistemas complejos y dinámicos. Los agentes en un modelo de AnyLogic pueden representar una variedad de entidades, desde individuos hasta vehículos o incluso países, cada uno con su propio conjunto de propiedades y reglas de comportamiento.

Por otro lado, posee una interfaz de simulación 2D y 3D, permitiendo a los usuarios visualizar los resultados de la simulación en diversos formatos, incluyendo gráficos, animaciones y tablas.

Es utilizado en gran variedad de sectores como logística, fabricación, cadena de suministros o atención sanitaria, entre muchos otros, ya que posee paletas de trabajo diseñadas específicamente para estos campos.

En resumen, Anylogic es una herramienta de simulación potente y versátil, que gracias a su gran interfaz de simulación puede ser muy útil para poder exponer un proyecto de simulación.

Para tener un conocimiento más profundo sobre el software de simulación, en el capítulo 4.1 entorno de desarrollo de Anylogic, se explican algunas herramientas que incluye el software como las diferentes paletas, algunos bloques de la paleta utilizados en este trabajo, algunos gráficos disponibles, el concepto de variable, parámetros y como se pueden exportar datos de la simulación a una hoja Excel externa.

2.5 Gamificación

La gamificación es reconocida como una de las más importantes técnicas de enseñanza durante esta década (Johnson, 2014). Inicialmente se desarrolló para los ámbitos de marketing y negocios, comenzó en el 2008 pero no se acabó generalizando hasta el 2010.

Se puede definir como “la incorporación de elementos de diseño de juegos en contextos no relacionados con el juego” (Deterding, 2011) . Su finalidad es generar o transformar conocimientos de manera que sus emociones y compromisos sean los mismos que si de un juego se tratase.

Dentro del ámbito académico, se concibe que la gamificación posee un potencial que se sustenta en la premisa de proporcionar apoyo y motivación a los estudiantes, lo que puede impulsar a mejoras en los procesos y resultados de aprendizaje (Kapp, 2012). Cabe destacar que su implantación no es sencilla y que es necesario invertir una gran cantidad d tiempo y esfuerzo por parte de los educadores (Alhammad y Moreno, 2018).

Como se ha comentado previamente, la práctica que ha servido de base para la elaboración de este Trabajo Fin de Grado consta del diseño de una planta de fabricación multiproducto que debe acomodar

con el menor impacto posible la variabilidad presente en la demanda, tanto en término de tipo de producto como de su cantidad, y sin que ello suponga el sacrificio de la eficiencia del sistema. La práctica se encuadra dentro del temario de la asignatura Sistemas Integrados de Producción, de cuarto curso del Grado en Ingeniería de Tecnologías Industriales (Escudero-Santana et al., 2022). Esta asignatura sirve a los alumnos de introducción a la filosofía Lean Manufacturing, a través del estudio de sus principales pilares y técnicas de aplicación. Una parte fundamental de la asignatura es la realización de prácticas y juegos que faciliten la adquisición y procesamiento por parte del alumnado de los contenidos incluidos en el temario. En este sentido (Stadnicka & Deif, 2019) afirman, con el respaldo del análisis de un caso de estudio desarrollado con alumnos de universidades estadounidenses y polacas, que la gamificación permite mejorar la motivación de los estudiantes y la adquisición de conocimientos en el proceso de aprendizaje de la filosofía Lean Manufacturing.

3 METODOLOGÍA

A continuación, se desarrolla la metodología utilizada para conformar el modelo de simulación. Se explica tanto la nomenclatura de las piezas, los árboles de precedencias de los distintos tipos de coche con sus respectivas matrices de comprobación, así como los parámetros de entrada necesarios y cómo introducirlos en el Excel 'DatosEntradaCoche', el cual que utiliza el software para importar dichos parámetros.

3.1 Introducción

En el entorno empresarial actual, la eficiencia y la optimización de los procesos productivos son aspectos clave para lograr el éxito y la competitividad. En este contexto, la generación de flujo se convierte en un aspecto fundamental de una factoría, donde el material y la información deben fluir de manera constante. Cuando estos flujos se interrumpen, surgen desperdicios y se compromete la eficiencia global del sistema. En este contexto, se encomienda a los alumnos el diseño de una nueva factoría para la fabricación de tres coches LEGO que comparten ciertas piezas. Se busca un sistema productivo versátil, capaz de adaptarse a diferentes ritmos de producción según la demanda dada. Para lograr esto, contamos con la libertad de modificar el estándar de ensamblado de los distintos tipos de coches y crear también diferentes estaciones de ensamblado.

A continuación, se muestra el diseño final de los tres tipos de coches y un ejemplo de diseño de layout:

Figura 5: coche tipo1

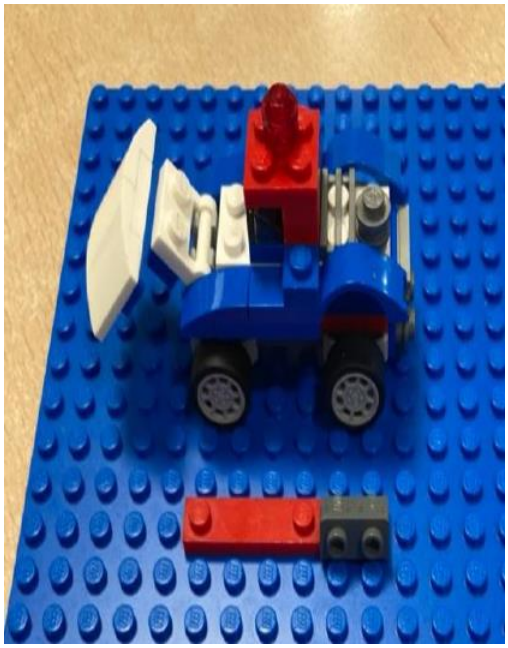

Figura 6: coche tipo 2

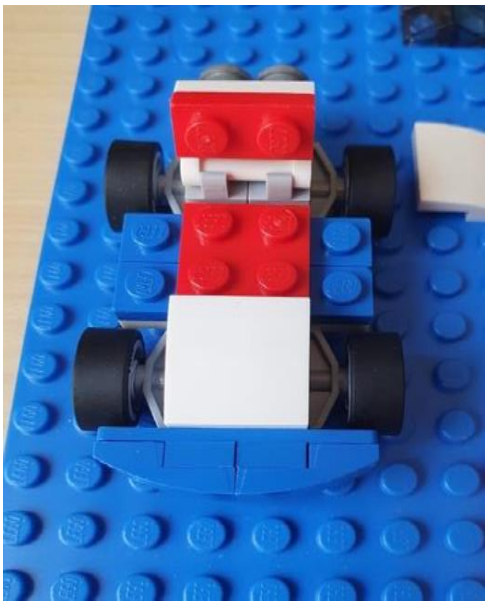

Figura 7: coche tipo 3

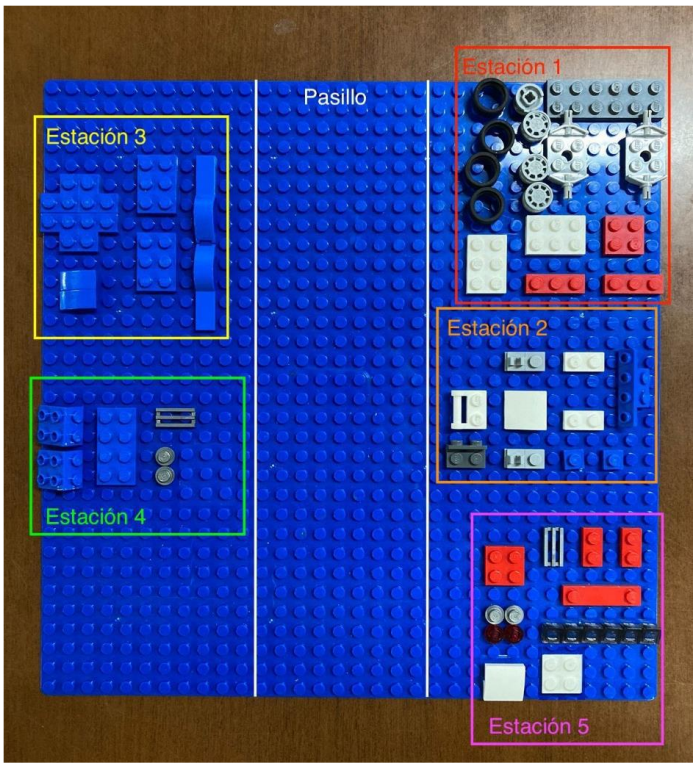

Figura 8: ejemplo layout planta

Para poder verificar el correcto diseño de esta factoría, en este trabajo se ha desarrollado un modelo de simulación basado en agentes. La principal diferencia entre cada tipo de coche radica en sus respectivos árboles de precedencia de montaje, composición de piezas y los tiempos asociados a sus procesos de ensamblaje.

Cada modelo de coche sigue un itinerario definido por el usuario, en base a las tareas necesarias para su montaje y su orden, avanzando secuencialmente a través de diferentes estaciones de trabajo. Estas tienen diferentes parámetros que han de ser previamente estudiados para crear una producción factible y eficiente, un ejemplo es la toma de tiempos, el alumno debe realizar varias mediciones de tiempo del montaje pieza a pieza y establecer el tiempo que tarda en ensamblar correctamente la pieza. El modelo de simulación basado en agentes creado proporciona una representación dinámica y realista de cómo cada coche avanza a través de las distintas etapas de montaje en la fábrica. La adaptabilidad intrínseca de este modelo permite considerar ciertas particularidades de cada fabrica diseñada, pudiendo cambiar diferentes parámetros como el takt time, capacidad de estaciones de trabajo, como son las secuencias de los tipos de coche e incluso realizar simulaciones con algunos parámetros aleatorios todo ello para entender el funcionamiento de la planta diseñada.

Para facilitar la entrada de datos a Anylogic se ha creado una hoja Excel que está conectada automáticamente con este software de simulación. El usuario introduce una serie de parámetros que serán explicados a lo largo de este capítulo. Esta hoja de cálculo también se encarga de verificar la viabilidad de construcción según los respectivos arboles de precedencia de cada tipo de coche. Mediante un cálculo matemático y algunas herramientas de Excel, todos los datos se agrupan en una sola hoja, que es la que se conecta con la base de datos de Anylogic.

Para poder incluir cada solución en este trabajo, se ha decido parametrizar todas las piezas.

3.2 Nomenclatura piezas

Para tener claro en todo momento de que pieza se trata, cada tipo de pieza tendrá asignado una variable con su nombre especifico. Se seguirá la siguiente nomenclatura explicada en la siguiente figura:

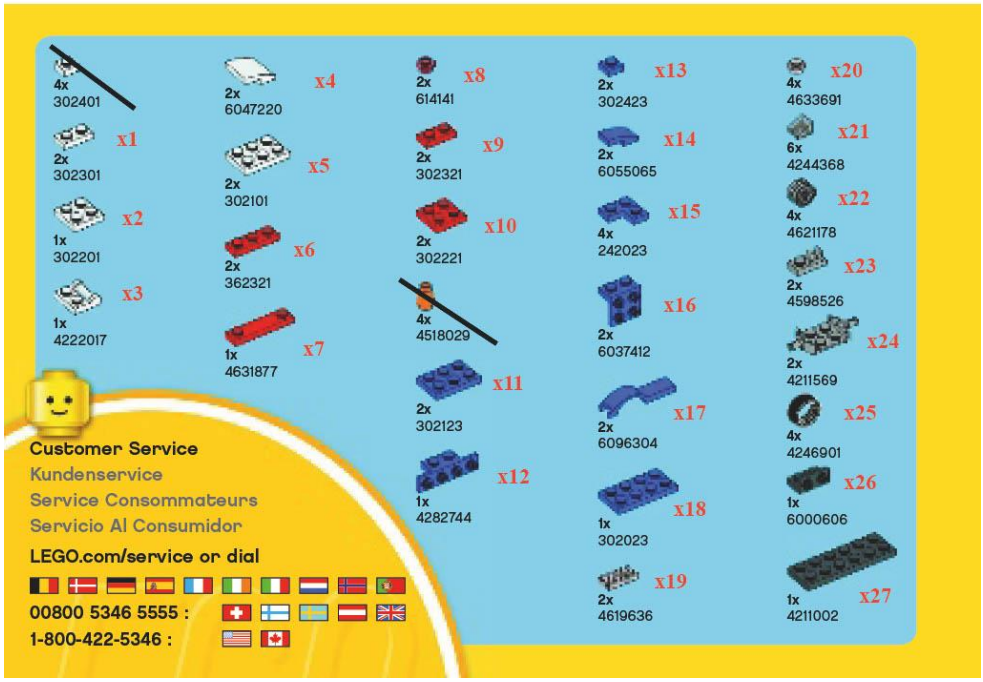

Figura 9: nomenclatura de piezas y variables (The LEGO Group, 2014)

Las piezas tachadas no serán utilizadas por lo que no se le asignan variables.

3.2.1 Nomenclatura para piezas idénticas en una misma toma de tiempo

En los casos en los que se utilice más de una pieza idéntica, como podría ser en el siguiente ejemplo con la pieza x21, a la hora de tomar los tiempos y referirnos a ese conjunto de piezas, se añadirá entre paréntesis tras el nombre de la variable las letras a,b,c,d,e,f,g en orden alfabético. Añadiremos una de las letras por cada pieza idéntica que haya, sin que se repita la letra. Aplicando esta nomenclatura al ejemplo anterior, la pieza x21, al colocarse 4 unidades de esta, se le denominaría x21(abcd) y se realizaría la toma del tiempo desde que se toma la primera pieza hasta que se coloca la cuarta.

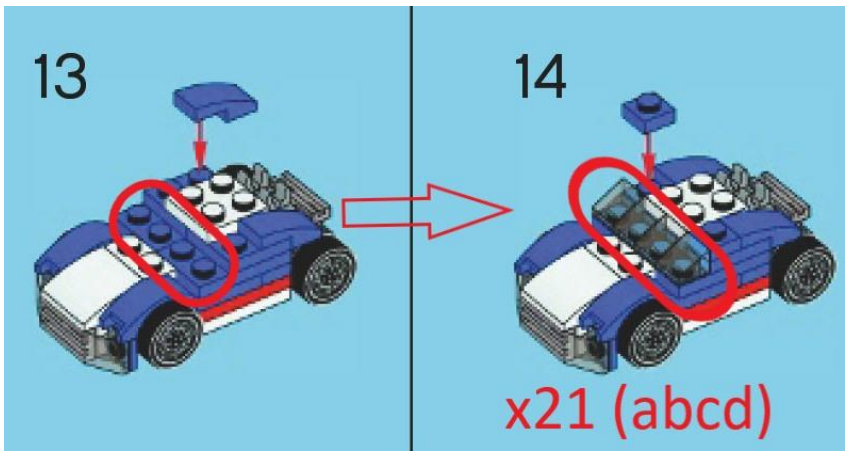

Figura 10: ejemplo nomenclatura piezas idénticas

3.2.2 Nomenclatura para piezas idénticas en etapa posterior de montaje

En el caso de que se haya utilizado ya en un paso previo (o en una estación diferente) una pieza idéntica, se le añadirán las letras entre paréntesis al igual que se hizo la primera vez, teniendo en cuenta las ya utilizadas.

En el siguiente ejemplo se puede ver como la primera vez se denomina x15(ab) (ya que hay dos piezas x15), y al continuar el proceso en una etapa posterior se nombra x15(cd)

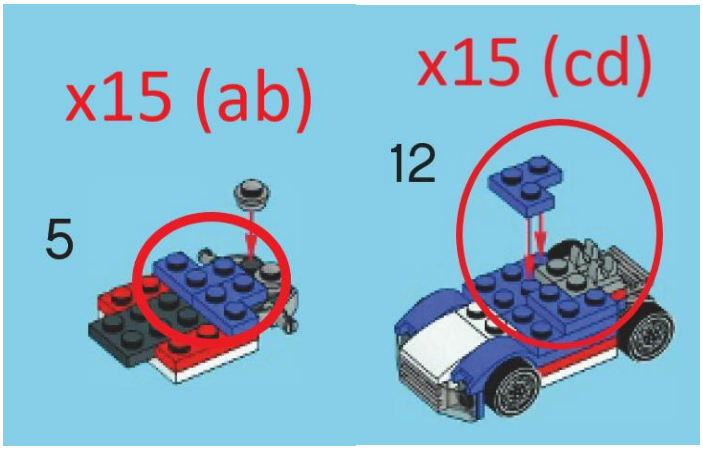

Figura 11: ejemplo nomenclatura piezas idénticas en etapa posterior de montaje

3.2.3 Piezas idénticas que se apilan

En el caso de que se apilen varias piezas idénticas, con la finalidad de dar la mayor libertad posible, se considerarán piezas con letra diferente. La pieza que se coloque primero será a la que se le asigne la primera letra disponible.

En el siguiente ejemplo, se puede ver como se asigna x9(a) a la pieza que se coloca primero. Si ya hubiese otra x9(a), la primera letra disponible sería la b, por lo tanto, se denominaría x9(b) a la primera pieza y x9(c) la que se coloca encima.

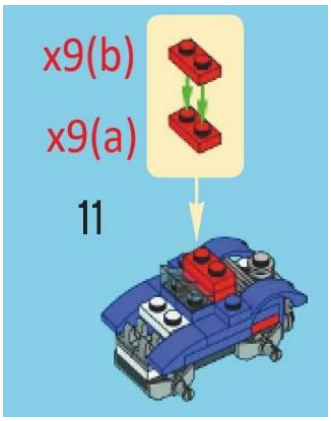

Figura 12: ejemplo nomenclatura piezas idénticas apiladas (1)

En los siguientes dos ejemplos ocurre la misma situación:

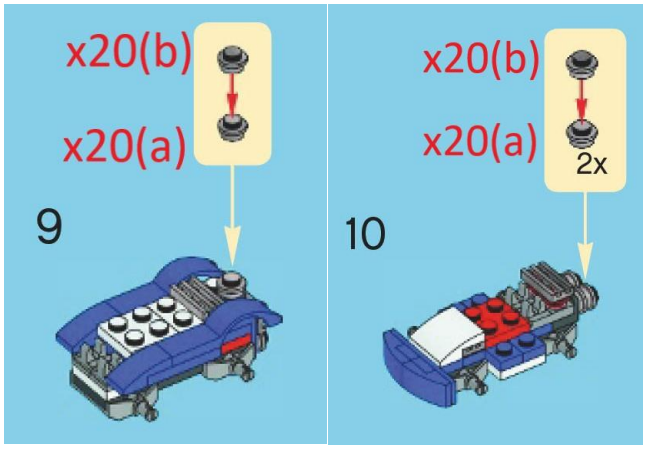

Figura 13: ejemplo nomenclatura piezas idénticas apiladas (2)

3.3 Árboles de precedencia

Para llevar a cabo el ensamblaje adecuado de cada tipo de coche, es estrictamente necesario tener claro las relaciones precedencia de cada uno de ellos. Se muestran imágenes de los árboles de precedencia para cada tipo de coche:

- Árbol de precedencia coche 1:

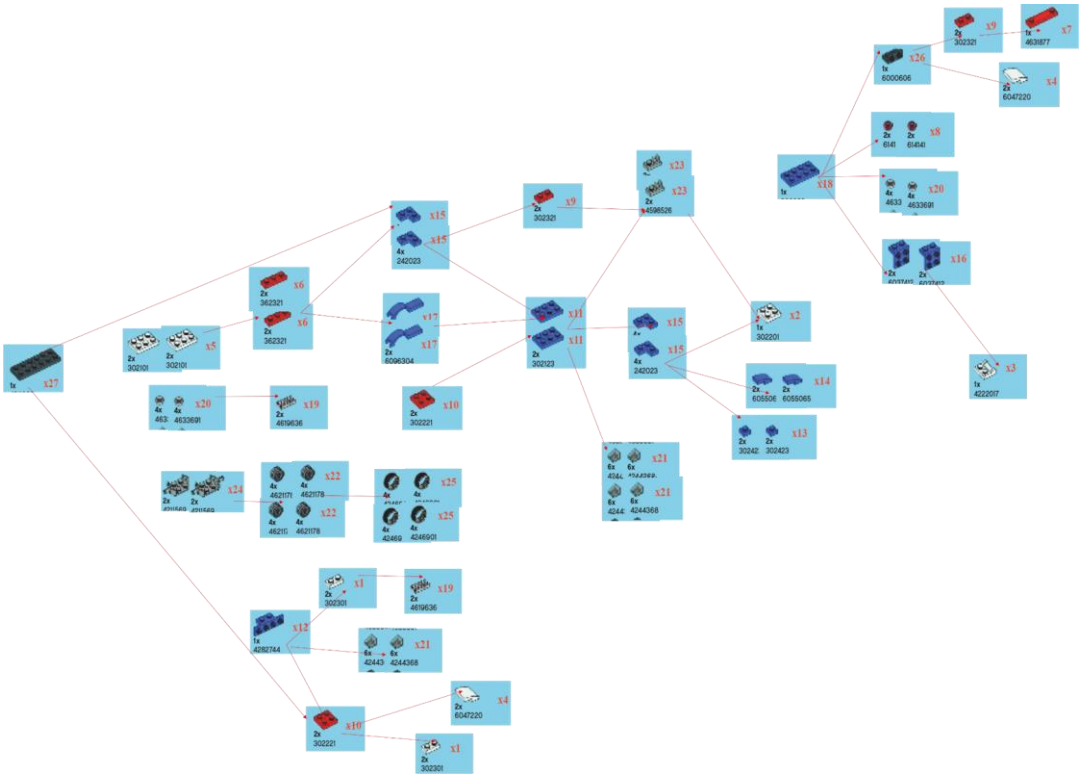

Figura 14: árbol precedencia coche 1

- Árbol de precedencia coche 2:

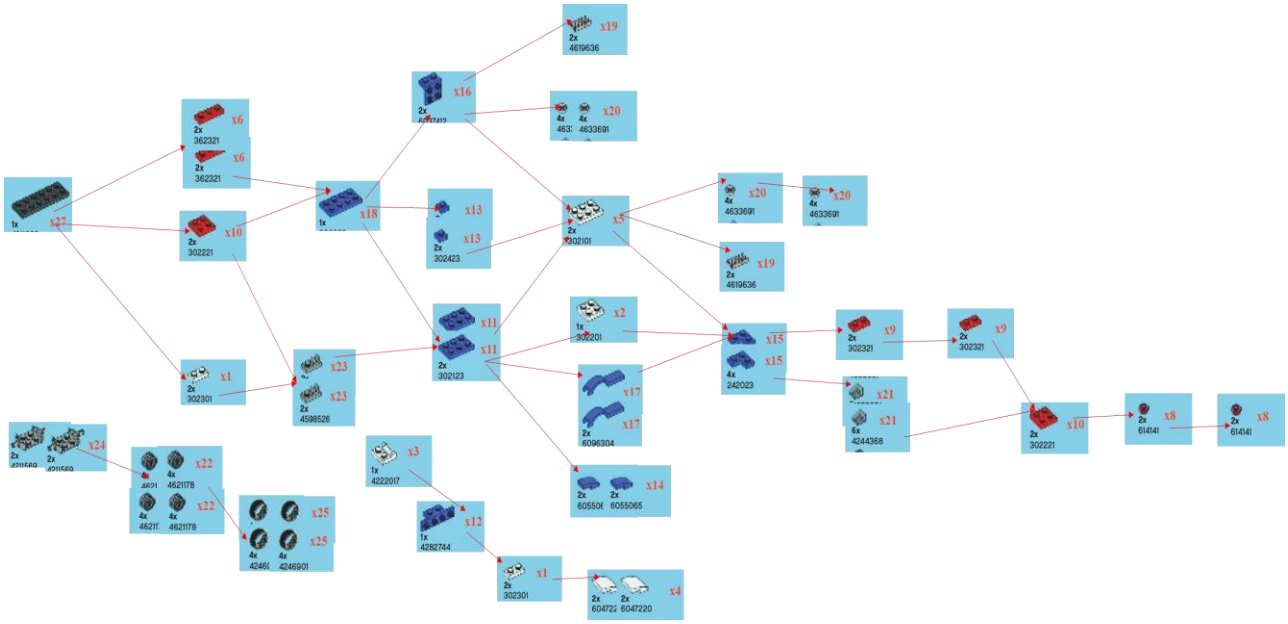

Figura 15: árbol precedencia coche 2

- Árbol de precedencia coche 3:

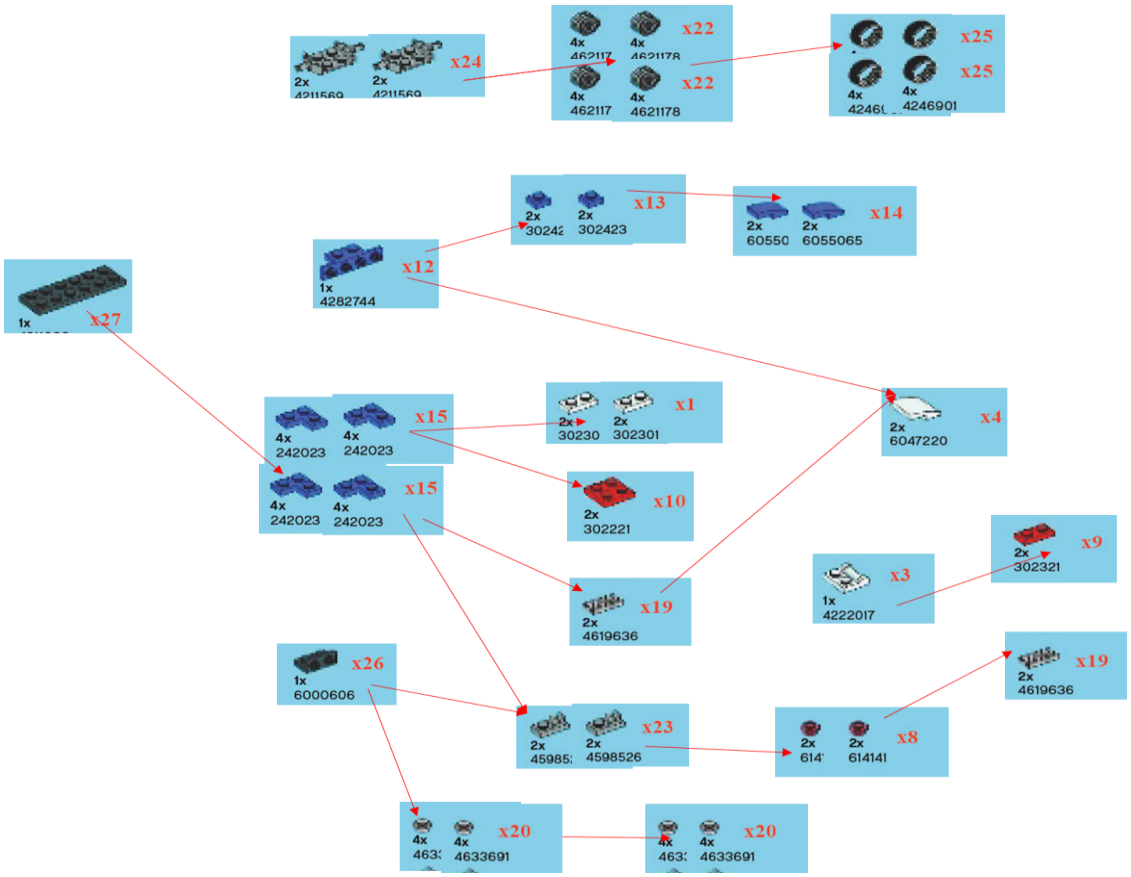

Figura 16: árbol precedencia coche 3

3.4 Matriz comprobación precedencias

Para verificar que el orden de tránsito por las estaciones sea compatible con las precedencias de ensamblaje, se ha diseñado una hoja en el Excel de datos dedicada a cada tipo de coche. En la hoja 'Datos entrada', el usuario ingresa la estación en la que se encuentra la pieza, y esta información se verifica en las hojas 'Prec coche 1', 'Prec coche 2' y 'Prec coche 3'.

Dado que también es crucial considerar el orden de la secuencia al evaluar el montaje, se ha asignado un peso a cada paso de la secuencia. Se ha creado una columna que, dependiendo de la estación en la que se encuentre la pieza, realiza un producto basado en el orden de secuencia específico de ese tipo de coche con respecto a la estación. Por ejemplo, si la estación está en el paso 1 de la secuencia, se multiplica por 1; si está en el paso 2, se multiplica por 3, y así sucesivamente hasta el paso 10, donde se multiplica por 27. De esta manera, se relaciona tanto la estación como el orden de la secuencia.

	Peso
Paso 1	1
Paso 2	3
Paso 3	6
Paso 4	9
Paso 5	12
Paso 6	15
Paso 7	18
Paso 8	21
Paso 9	24
Paso 10	27

Tabla 1: peso pasos matriz comprobación precedencias

Coche 3

Pieza	Valor
x1 (a)	12
x1 (b)	12
x3 (a)	12
x4 (a)	48
x8 (a)	45
x8 (b)	45
x9 (a)	45
x10 (a)	45
x12 (a)	12
x13 (a,b)	12
x14 (a,b)	12
x15 (a,b)	9
x15 (c,d)	9
x19 (a)	45
x19 (b)	48
x20 (a,b)	45
x20 (c)	48
x20 (d)	48
x22 (a,b,c,d)	1
x23 (a,b)	12
x24 (a,b)	1
x25 (a,b,c,d)	1
x26 (a)	12
x27 (a)	1

Tabla 2: valores matriz comprobación precedencias

Se han utilizado funciones condicionales de Excel como 'SI', 'Y'. 'O', 'SI. CONJUNTO'. Si la matriz tiene todas las celdas en verde con la palabra CORRECTO significa que el orden de montaje es correcto y se podrá producir el coche. En cambio, sí una de las celdas de la matriz está en rojo con la

palabra INCORRECTO, la fabricación de ese coche no será posible ya que no cumple con las precedencias necesarias para la construcción de este.

Se muestra como ejemplo la del tipo de coche 3:

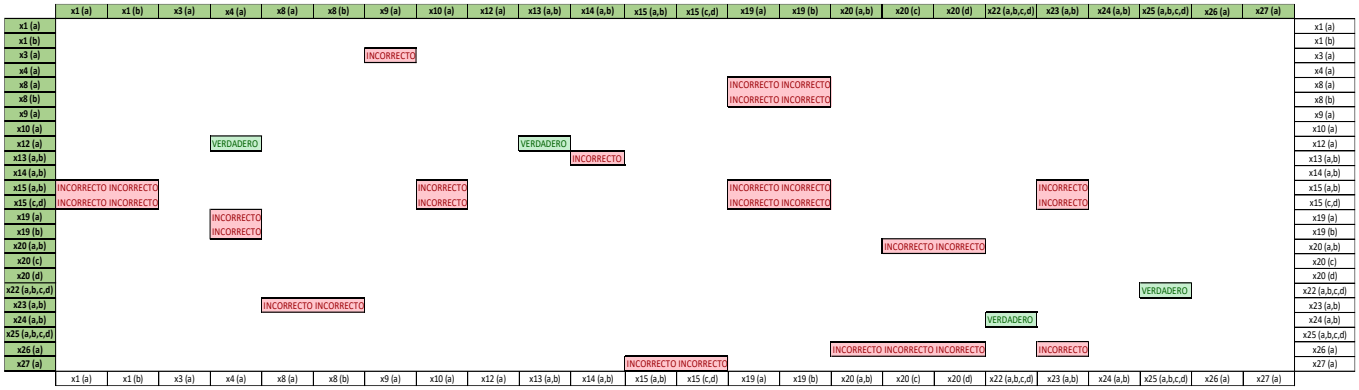

Tabla 3: Matriz comprobación precedencias Coche 3

Verificación Matriz
INCORRECTO

Tabla 4: Resumen matriz comprobación precedencias

Para simplificar el proceso y evitar tener que revisar todas las celdas de cada matriz de cada coche, hay una celda en la hoja de datos de entrada. Esta celda mostrará el mensaje ‘VERDADERO’ en color verde si es posible realizar el montaje de todos los tipos. En caso de que no sea posible alguno de ellos, se mostrará 'PRECEDENCIAS INCORRECTAS' en color rojo.

Coche 1	Coche 2	Coche 3
VERDADERO	INCORRECTO	INCORRECTO
COMPROBACIÓN PRECEDENCIAS		
INCORRECTO		

Tabla 5: Resumen global matrices comprobación precedencias

3.5 Cálculo de tiempo de proceso en estaciones

Para poder obtener los tiempos que tiene que estar cada tipo de coche en las diferentes estaciones, se utiliza el siguiente cálculo:

$$\sum_{i=0}^{40} \alpha_{i,j} * t_i \quad \forall\, j \in J$$

Donde:

$$\alpha_{i,j} =$$

$$\{1 \text{ si pieza i si se ensambla en la estación j; } 0 \text{ en caso contrario}\}$$

$$t_i = tiempo\ de\ ensamblaje\ pieza\ i$$

$$J = número\ total\ de\ estaciones$$

El sumatorio comienza en 0 y termina en 40 ya que este es el número de piezas diferentes que existen.

Para implementarlo, se ha creado una hoja Excel con nombre 'MODELO' en la que se encuentran para cada tipo de coche 4 matrices diferentes, 2 correspondientes al modelo, otra correspondiente a los tiempos de cada pieza y otra el resultado de los tiempos obtenidos del modelo.

- Matrices relaciones pieza con estación: la matriz de la izquierda de la Figura 2, columnas y tantas filas como piezas diferentes tenga el coche) sirve para tomar los datos dados por el usuario (en que estación va cada pieza) de la hoja datos de entrada.
  La matriz de la derecha de la figura está relacionada con el cálculo matemático previamente mencionado, las filas son las diferentes piezas y las columnas las distintas estaciones, si una pieza pasa en algún momento por una estación esa celda tendrá valor 1, en caso contrario valdrá 0.

COCHE 1	
Pieza	Estación
x1 a	1
x1 b	2
x2 (a)	3
x3 (a)	4
x4 (a)	5
x4 (b)	2
x5 (a)	3
x5 (b)	3
x6 (a,b)	5
x7 (a)	4
x8 (a)	5
x8 (b)	1
x9 (a)	2
x9 (b)	3
x10 (a)	5
x10 (b)	10
x11 (a,b)	8
x12 (a)	5
x13 (a,b)	4
x14 (a,b)	5
x15 (a,b)	4
x15 (c,d)	5
x16 (a)	1
x16 (b)	2
x17 (a,b)	10
x18 (a)	8
x19 (a)	8
x19 (b)	1
x20 (a,b)	5
x20 (c)	9
x20 (d)	9
x21 (a,b)	7
x21(c,d)	7
x21 (e,f)	10
x22 (a,b,c,d)	10
x23 (a,b)	10
x24 (a,b)	6
x25 (a,b,c,d)	3
x26 (a)	2
x27 (a)	4

	1	2	3	4	5	6	7	8	9	10
x1 a	1	0	0	0	0	0	0	0	0	0
x1 b	0	1	0	0	0	0	0	0	0	0
x2 (a)	0	0	1	0	0	0	0	0	0	0
x3 (a)	0	0	0	1	0	0	0	0	0	0
x4 (a)	0	0	0	0	1	0	0	0	0	0
x4 (b)	0	1	0	0	0	0	0	0	0	0
x5 (a)	0	0	1	0	0	0	0	0	0	0
x5 (b)	0	0	1	0	0	0	0	0	0	0
x6 (a,b)	0	0	0	0	1	0	0	0	0	0
x7 (a)	0	0	0	1	0	0	0	0	0	0
x8 (a)	0	0	0	0	1	0	0	0	0	0
x8 (b)	1	0	0	0	0	0	0	0	0	0
x9 (a)	0	1	0	0	0	0	0	0	0	0
x9 (b)	0	0	1	0	0	0	0	0	0	0
x10 (a)	0	0	0	0	1	0	0	0	0	0
x10 (b)	0	0	0	0	0	0	0	0	0	1
x11 (a,b)	0	0	0	0	0	0	0	1	0	0
x12 (a)	0	0	0	0	1	0	0	0	0	0
x13 (a,b)	0	0	0	1	0	0	0	0	0	0
x14 (a,b)	0	0	0	0	1	0	0	0	0	0
x15 (a,b)	0	0	0	1	0	0	0	0	0	0
x15 (c,d)	0	0	0	0	1	0	0	0	0	0
x16 (a)	1	0	0	0	0	0	0	0	0	0
x16 (b)	0	1	0	0	0	0	0	0	0	0
x17 (a,b)	0	0	0	0	0	0	0	0	0	1
x18 (a)	0	0	0	0	0	0	0	1	0	0
x19 (a)	0	0	0	0	0	0	0	1	0	0
x19 (b)	1	0	0	0	0	0	0	0	0	0
x20 (a,b)	0	0	0	0	1	0	0	0	0	0
x20 (c)	0	0	0	0	0	0	0	0	1	0
x20 (d)	0	0	0	0	0	0	0	0	1	0
x21 (a,b)	0	0	0	0	0	0	1	0	0	0
x21(c,d)	0	0	0	0	0	0	1	0	0	0
x21 (e,f)	0	0	0	0	0	0	0	0	0	1
x22 (a,b,c,d)	0	0	0	0	0	0	0	0	0	1
x23 (a,b)	0	0	0	0	0	0	0	0	0	1
x24 (a,b)	0	0	0	0	0	1	0	0	0	0
x25 (a,b,c,d)	0	0	1	0	0	0	0	0	0	0
x26 (a)	0	1	0	0	0	0	0	0	0	0
x27 (a)	0	0	0	1	0	0	0	0	0	0

Tabla 6: matrices pieza-estación

- Matrices relacionadas con el tiempo: la matriz de la izquierda de la Tabla 7 (2 columnas y tantas filas como piezas diferentes tenga el coche) toma los datos de la hoja ‘datos de entrada’. Por otro lado, la matriz de la derecha es el resultado obtenido del cálculo (de la matriz binaria que relaciona las piezas con las estaciones y la matriz tiempo que se acaba de nombrar), en esta se pueden ver los tiempos totales que está el coche en la estación.

TIEMPO	
COCHE 1	
Pieza	Tiempo
x1 a	17
x1 b	6
x2 (a)	5
x3 (a)	4
x4 (a)	3
x4 (b)	3
x5 (a)	6
x5 (b)	7
x6 (a,b)	4
x7 (a)	3
x8 (a)	6
x8 (b)	3
x9 (a)	3
x9 (b)	5
x10 (a)	6
x10 (b)	8
x11 (a,b)	8
x12 (a)	3
x13 (a,b)	2
x14 (a,b)	1
x15 (a,b)	3
x15 (c,d)	5
x16 (a)	7
x16 (b)	8
x17 (a,b)	9
x18 (a)	6
x19 (a)	6
x19 (b)	5
x20 (a,b)	4
x20 (c)	0
x20 (d)	8
x21 (a,b)	9
x21(c,d)	6
x21 (e,f)	4
x22 (a,b,c,d)	3
x23 (a,b)	2
x24 (a,b)	6
x25 (a,b,c,d)	7
x26 (a)	8
x27 (a)	9

TIEMPO ESTACIÓN	
COCHE 1	
1	32
2	28
3	30
4	21
5	32
6	6
7	15
8	20
9	8
10	26

Tabla 7: matrices tiempo-estación

3.6 Datos entrada

Con el objetivo de simplificar la entrada de datos en Excel para el usuario, se ha determinado que todos los datos previamente adquiridos por el usuario durante sus pruebas y cálculos se registrarán exclusivamente en una hoja de cálculo designada. Esta hoja se utiliza para transferir los datos a las hojas correspondientes del cálculo matemático, a aquellas destinadas a la validación de los árboles de precedencia y a la hoja de datos de salida. Esta última será la fuente desde la cual el software AnyLogic

tomará los datos necesarios.

Los tipos de datos de entrada se explicarán en los siguientes subapartados:

3.6.1 Tiempos de proceso

Se introducen los tiempos requeridos para el ensamblaje de cada pieza de acuerdo con el tipo de coche. Es importante destacar que, incluso si se trata de la misma pieza, los tiempos de ensamblaje pueden variar entre los diferentes modelos de coche debido a las diferencias en sus conjuntos de piezas. Esto puede deberse, por ejemplo, a la dificultad de acceso al lugar donde se debe instalar la pieza, y refleja situaciones reales acaecidas en la industria automovilística.

Para poder contabilizar correctamente los tiempos e introducirlos en la hoja Excel, es muy importante seguir la nomenclatura anteriormente utilizada. La columna en la que se introducen los tiempos es la que se encuentra resaltada en rojo en la siguiente Tabla 8:

COCHE 3		
Pieza	Tiempos	Estación
x1 (a)	22	1
x1 (b)	11	2
x3 (a)	31	3
x4 (a)	16	4
x8 (a)	10	5
x8 (b)	0	0
x9 (a)	0	0
x10 (a)	0	0
x12 (a)	0	0
x13 (a,b)	0	0
x14 (a,b)	0	0
x15 (a,b)	0	0
x15 (c,d)	0	0
x19 (a)	0	0
x19 (b)	0	0
x20 (a,b)	0	0
x20 (c)	0	0
x20 (d)	0	0
x22 (a,b,c,d)	0	0
x23 (a,b)	0	0
x24 (a,b)	0	0
x25 (a,b,c,d)	0	0
x26 (a)	0	0
x27 (a)	0	0

Tabla 8: matriz de datos de entrada por tipo de coche (tiempos)

3.6.2 Estación de la pieza

Para poder conocer los tiempos que cada coche pasa en las diferentes estaciones, se introduce para

cada pieza la estación en la que será ensamblada. La columna en la que se introducen es la que se encuentra resaltada en rojo en la siguiente matriz

COCHE 3		
Pieza	Tiempos	Estación
x1 (a)	22	1
x1 (b)	11	2
x3 (a)	31	3
x4 (a)	16	4
x8 (a)	10	5
x8 (b)	0	0
x9 (a)	0	0
x10 (a)	0	0
x12 (a)	0	0
x13 (a,b)	0	0
x14 (a,b)	0	0
x15 (a,b)	0	0
x15 (c,d)	0	0
x19 (a)	0	0
x19 (b)	0	0
x20 (a,b)	0	0
x20 (c)	0	0
x20 (d)	0	0
x22 (a,b,c,d)	0	0
x23 (a,b)	0	0
x24 (a,b)	0	0
x25 (a,b,c,d)	0	0
x26 (a)	0	0
x27 (a)	0	0

Tabla 9: matriz de datos de entrada por tipo de coche (estación)

3.6.3 Número estaciones

El parámetro inicial proporcionado por el usuario tiene como finalidad prevenir la asignación de una estación de trabajo a una pieza que exceda el número total de estaciones previamente definidas. En caso de que el usuario introduzca un número de estación superior al límite establecido, la hoja de cálculo lo identifica y se resalta en color rojo el valor correspondiente en la columna de estaciones, facilitando así la detección visual del error.

Número de estaciones	5

Figura 17: dato de entrada número de estaciones

3.6.4 Orden estaciones para cada coche

Matriz que relaciona el tipo de coche con los pasos que realiza y las estaciones por las que pasa.

	Orden estaciones		
	Coche 1	Coche 2	Coche 3
Paso 1	1	1	1
Paso 2	3	2	3
Paso 3	2	4	2
Paso 4	4	3	5
Paso 5	5	5	4
Paso 6	-1	-1	-1
Paso 7			
Paso 8			
Paso 9			
Paso 10			
FINAL			

Tabla 10: matriz de datos de entrada orden estaciones para cada coche

En la Tabla. 10, se aprecia que la columna representa el tipo de coche, y las filas corresponden a los pasos seguidos. Los números introducidos en cada celda indican las estaciones por las cuales pasa el coche en cuestión. Por ejemplo, en el paso 2 (fila 2) para el coche 1 (columna 1), si se introduce el número 3, significa que la segunda estación visitada por el coche 1 será la estación 3.

Es de vital importancia considerar que para indicar que el coche se desplaza directamente al final de la producción, sin regresar a ninguna otra estación, se debe introducir el valor -1. Esto se aplica, por ejemplo, si el coche 2 ha completado todas las 5 estaciones asignadas (el coche no tiene por qué pasar por todas las estaciones disponibles), en el paso 6 y en la columna correspondiente al coche 2, se debe escribir un -1 para indicar que pasa directamente al final del proceso de producción.

3.6.5 Selección tipo demanda

Con el fin realizar un estudio aún más completo se ha integrado un modo de demanda aleatoria Este modelo está disponible para su selección en la hoja de cálculo de Excel a través de un desplegable que permitirá elegir entre dos modos: Aleatorio y Específico.

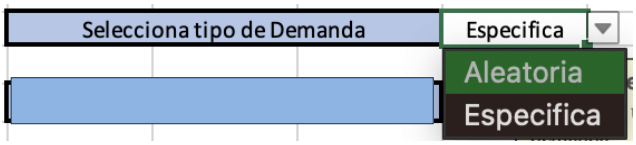

Figura 18: dato de entrada número de estaciones

Se generará de manera aleatoria el tipo de coche a fabricar. Además, el intervalo de tiempo entre la llegada de un coche y otro se ajusta según una distribución normal de media 5 y desviación típica 0, con el propósito de incrementar la aleatoriedad en el modelo. Es importante destacar que más adelante se proporciona una explicación detallada de este proceso, incluyendo el código en Java utilizado.

3.6.6 Vector tipo de coche demandado (Exclusivo modo específico)

Este vector se empleará exclusivamente cuando sea seleccionado el caso de demanda específica. El usuario introduce los tipos de coche que desea producir en el orden deseado. A modo de ejemplo, si el usuario introduce la secuencia 1, 1, 3, 2, 2, esto indica que se fabricarán dos coches de tipo 1 consecutivamente, luego un coche de tipo 3 y, finalmente, dos coches de tipo 2 en sucesión.

Vector Tipo coche demanda
1
1
3
2

Tabla 11: vector secuencia tipo coche demandado

3.6.7 Número de secuencias repetidas (Exclusivo modo específico)

Se utilizará exclusivamente para el caso de demanda específica para determinar el número de veces que se repetirá el vector tipo de coche demandado.

Figura 19: dato de entrada número de secuencias repetidas

El número total de coches producidos se obtiene multiplicando el número de componentes en el vector del tipo de coche demandado por el número de secuencias repetidas introducido. Para simplificar se

muestra el número total de coches producidos en un cuadro dentro de la hoja de cálculo, evitando así que el usuario tenga que realizar este cálculo manualmente.

Número Coches por Secuencia		Número totales de coches Producidos	
4		40	

Figura 20: número de coches por secuencia y total producidos

3.6.8 Capacidad de las estaciones

El usuario tiene que introducir la capacidad máxima de cada estación, es decir, la capacidad que tiene la estación para tener esperando en cola otro coche mientras se está produciendo uno. Es importante destacar que, aunque sean coches de tipos diferentes, en la estación solo se puede estar produciendo un coche simultáneamente.

Capacidad de las estaciones	2

Figura 21: dato de entrada capacidad estaciones

En la figura anterior se aprecia una configuración en la que el bloque 'Wait' (utilizado para que un agente espere antes de entrar en una estación) tendría una capacidad de dos, es decir, además del coche que se esté produciendo, la estación podría tener otros dos coches en cola.

3.6.9 Total coches (Exclusivo modo aleatorio)

Al igual que para la demanda específica se obtiene el número total de coches, para la demanda aleatoria al no existir un vector tipo de coche y por tanto tampoco tener influencia el número de secuencias repetidas, se le pide al usuario introducir el número total de coches que quiere que se produzcan en este caso de demanda aleatoria.

Total Coches Aleatorios	130

Figura 22: total coches aleatorios

3.7 Conexión Base de datos Anylogic con hoja de datos de entrada

Como se mencionó anteriormente, para que el software Anylogic tome los datos introducidos por el usuario y los resultados obtenidos por el Excel, se ha decidido unificarlo todo en una sola hoja de cálculo. Esta hoja tiene el nombre 'DatosSalidaAny' y se puede ver en la siguiente Tabla 12:

TiemposC1	TiemposC2	TiemposC3	Numero Esta	Takt Time	Tipo Coche D	Tipo de dema	Numero Secu	Orden Est C1	Orden Est C2	Orden Est C3
32	25	29	10	36	1	1	10	1	1	1
28	21	24			2			3	2	3
30	28	11			3			2	4	2
21	15	5			3			4	3	5
32	23	12			2			5	5	4
6	8	11			1			6	10	8
15	6	3						7	9	7
20	15	3						8	8	9
8	19	12						9	7	10
26	5	13						10	6	6
								-1	-1	-1

Tabla 12: Excel datos salida para Anylogic (1)

Capacidad de las estacion	Total Coches	Total Coches Aleatori	Max Total Coches
3	60	130	130

Tabla 13: Excel datos salida para Anylogic (2)

Todas las columnas han sido explicadas previamente en los subapartados del 3.6 a excepción de MaxTotalCoches que sirve para dar una capacidad a los bloques TimeMeasureEnd, esta variable toma el valor máximo entre los coches que se producirían por demanda aleatoria y los que se producirían por demanda específica.

4 DESCRIPCIÓN DEL MODELO DE SIMULACIÓN

En este capítulo se detallan todos los aspectos relacionados con el modelo de simulación desarrollado en Anylogic. Se explican variables, parámetros del agente creado, bloques y sus acciones, gráficos implementados y la exportación de datos realizada. Para tener una visión global, se muestra a continuación una Figura del modelo de simulación completo:

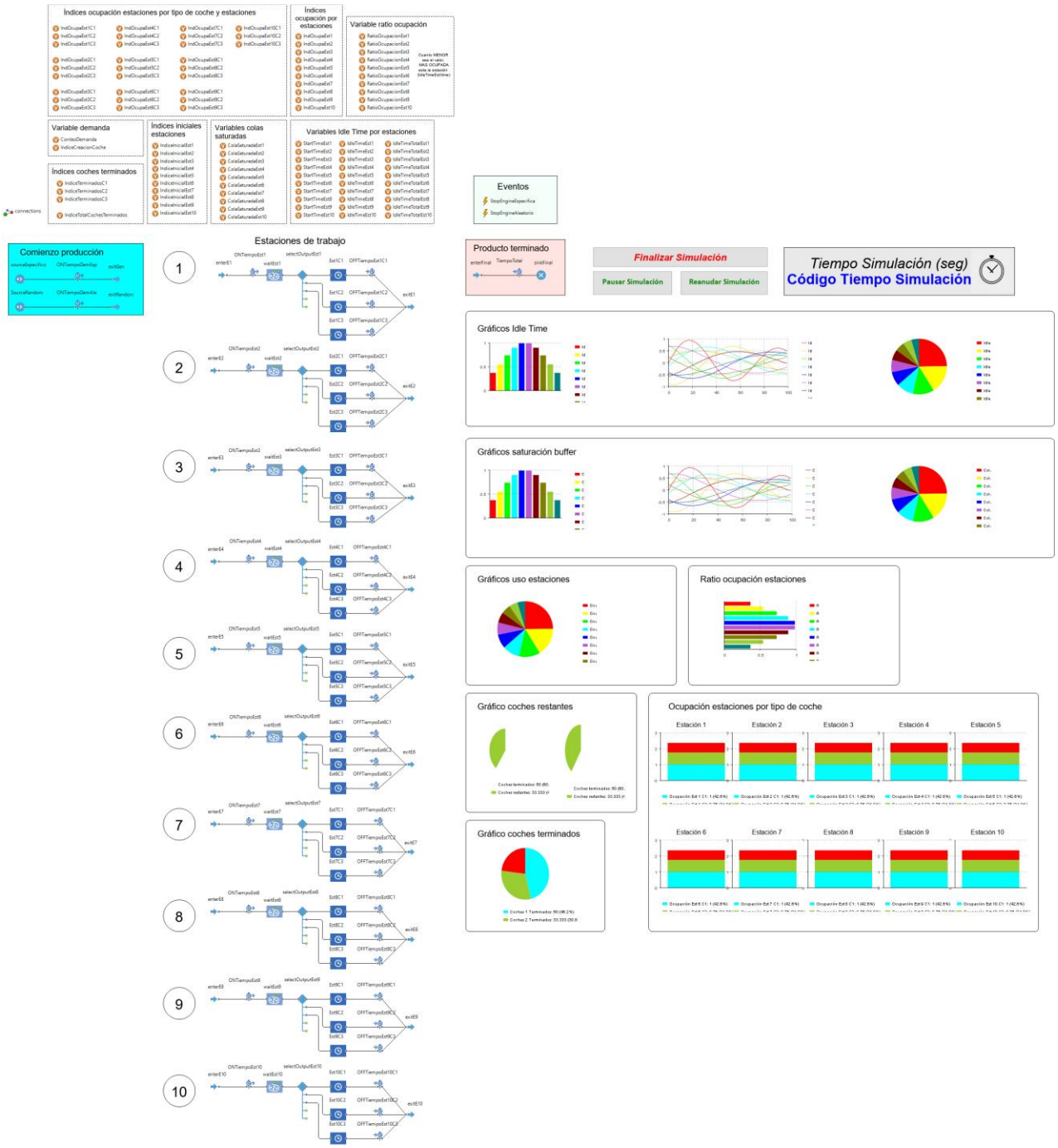

Figura 23: modelo de simulación completo

4.1 Entorno de desarrollo de Anylogic, visión general

Se explican algunas herramientas que incluye el software como las diferentes paletas, algunos bloques de la paleta utilizados en este trabajo, algunos gráficos disponibles, el concepto de variable, parámetros y como se pueden exportar datos de la simulación a una hoja Excel externa.

Paletas disponibles (The Anylogic Company, 2023):

Conjunto de herramientas predefinidas que ayudan al usuario a la creación de los modelos de simulación. arrastrando y soltando los bloques disponibles dentro cada paleta acelera el desarrollo de estos.

- Process Modeling Library: permite el modelado de procesos y actividades secuenciales (secuencias de operaciones, colas, retrasos…)
- Material Handling Library: simplifica la simulación de sistemas de fabricación complejos. Operaciones, almacenamiento y gestión de flujo de materiales.
- Pedestrian Library: utilizada para la simulación de flujo de peatones en entornos físicos como aeropuertos, edificios, centros comerciales, etc.
- Rail Library: diseñada para modelar, simular y visualizar sistemas de flujos de tráfico ferroviario de cualquier complejidad y tamaño. Útil para operaciones ferroviarias y planificación de rutas.
- Road traffic Library: permite diseñar, simular y planificar tráfico vehicular en carreteras y calles. Ayuda al análisis de congestiones, tiempos de viaje y planificación de carreteras.
- Fluid Library: modelado de flujo de fluidos, materia a granel o grandes cantidades. Útil para simulación de redes de tuberías y sistemas hidráulicos.
- System dynamics: permite el modelado de sistemas dinámicos complejos
- Agent: permite el modelado de sistemas basados en agentes, son los principales bloques de Anylogic.
- Presentation: permite diseñar presentaciones 2D y 3D complejas e interfaces de usuario, compuesta por varias formas (círculos, rectángulos, etc.)

- Analysis: herramientas para el análisis y visualización de datos. Objetos de análisis de datos, gráficos simples (barras, circulares, barras apiladas), gráficos temporales, histogramas.
- Controls: implementación de controles para crear modelos interactivos (botones, casillas de verificación, deslizadores, etc.).
- Statechart: construcción visual para describir comportamientos basados en eventos y tiempos, bloques puntos de entrada, estados, transiciones, ramificación, etc.
- Connectivity: facilita la integración y las conexiones con bases de datos externas y otros sistemas
- Pictures: contiene imágenes frecuentemente usadas en Anylogic en formato vectorial.
- 3D objects: contiene diferentes imágenes 3d que son frecuentemente utilizados en Anylogic como tipos de personas, edificios, elementos de supermercados, oficinas, carreteras, etc.

En la siguiente figura, recuadrado en rojo, se pueden ver todas las paletas anteriormente explicadas:

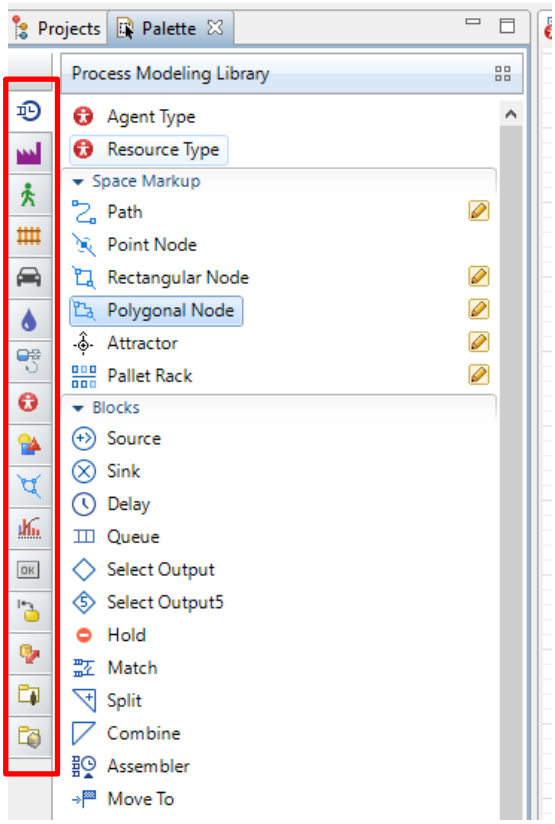

Figura 24: paletas disponibles en Anylogic

Bloques de la Process Modeling Library (The Anylogic Company, 2023) :

Esta librería está formada por más de 40 bloques, algunos de ellos serán explicados a continuación. Permite modelar operaciones logísticas, de fabricación y otro tipo de procesos de forma dinámica. También ayuda a los usuarios a comprender la dinámica de los procesos y obtener información útil para la toma de decisiones.

Los puntos verdes que se pueden ver en las siguientes imágenes, indican las posibles conexiones con otros bloques.

- Source: genera los agentes. Puedes personalizar los agentes generados y las acciones que realizan antes de que salga. Suele ser el comienzo de del modelo.

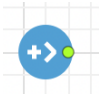

Figura 25: bloque Source Anylogic

- Sink: elimina los agentes del modelo. Suele estar situado al final del modelo.

Figura 26: bloque Sink Anylogic

- Queue: almacenas entidades antes de ser procesadas por el siguiente bloque.

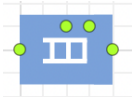

Figura 27: bloque Queue Anylogic

- SelectOutput5: dirige a las entidades, según unas condiciones (probabilísticas o determinísticas) establecidas por el usuario, a una de las cinco salidas que tiene.

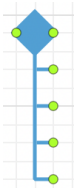

Figura 28: bloque Queue Anylogic

- Enter: permite la entrada de las entidades a los bloques conectados.

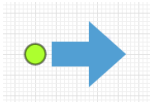

Figura 29: bloque Enter Anylogic

- Exit: toma los agentes del bloque que tiene conectado y permite al usuario decidir a que otro bloque se dirigen.

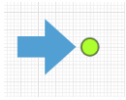

Figura 30: bloque Exit Anylogic

- Delay: retrasa el avance al siguiente bloque de los agentes durante una cantidad de tiempo determinada por el usuario.

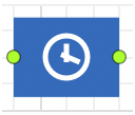

Figura 31: bloque Delay Anylogic

- MoveTo: mueve una entidad a una ubicación específica en el modelo, si se trata de un agente que tiene adjunto un recurso, se moverá junto a él.

Figura 32: bloque MoveTo Anylogic

- TimeMeasureStart y TimeMeasureEnd : estos bloques miden el tiempo que tardan los agentes en pasar entre un bloque y otro. TimeMeasureStart marca el inicio de esta medición y TimeMeasureEnd la detiene.

Figura 33: bloques TimeMeasureStart y TimeMeasureEnd Anylogic

- Wait: hace que un agente espere hasta que se cumpla una condición específica. Se suele utilizar para modelar situaciones en las que las entidades han de esperar antes de continuar con su procesamiento.

Figura 34: bloque Wait Anylogic

- RackStore: coloca un agente en una celad de una estantería específica o una zona de almacenamiento.

Figura 35: bloque RackStore Anylogic

- SelectOutput: en función de una condición dada por el usuario, reenvía al agente a una salida u otra.

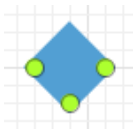

Figura 36: bloque SelectOutput Anylogic

- Hold: bloquea y desbloquea el flujo del agente.

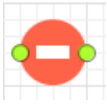

Figura 37: bloque Hold Anylogic

Algunos gráficos disponibles (The Anylogic Company, 2023) :

- Diagrama de barras: muestra una serie de datos como barras, su altura es proporcional al tamaño del dato.

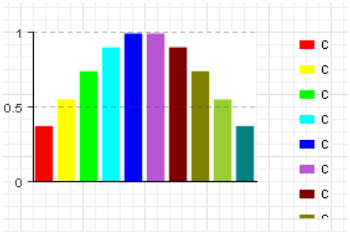

Figura 38: gráfico diagrama de barras Anylogic

- Gráfico temporal (Time Plot): herramienta de visualización que muestra cómo cambia una variable a lo largo del tiempo de la simulación, facilitando la comprensión de posibles tendencias y patrones.

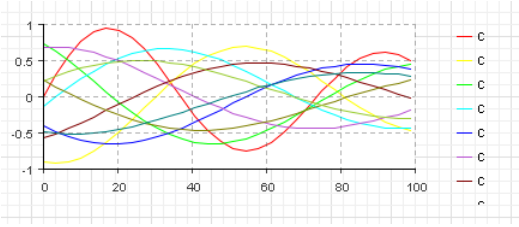

Figura 39: gráfico temporal Anylogic

- Gráfico circular (Pie Chart): Muestra distintos datos en forma de sectores circulares, el tamaño del sector es proporcional al valor del dato. Si el valor es negativo o cero no se mostrará en ningún sector.

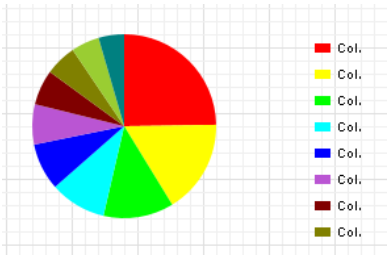

Figura 40: gráfico circular Anylogic

- Histograma (Histogram): permite la visualización de las estadísticas recopiladas por varios objetos de datos de histogramas. Ambos ejes se escalan automáticamente a lo largo de la simulación, si se desea es posible mostrar las barras de función de distribución de probabilidad, las líneas de función de distribución acumulada y la ubicación media

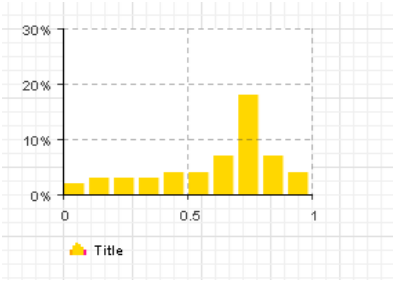

Figura 41: histograma Anylogic

- Diagrama de barras acumulados (Stack Chart): muestra varios elementos de datos mediante barras apiladas una encima de otra, el primer elemento se sitúa en la parte inferior. Los tamaños son proporcionales a los valores de los correspondientes datos. Si el valor es 0 no se mostrará y si es negativo se producirá un error.

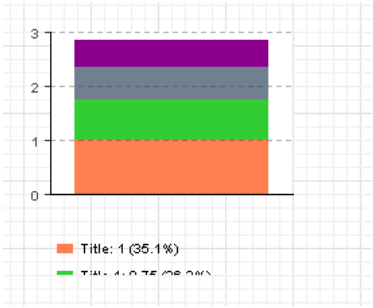

Figura 42: gráfico diagrama de barras acumulado Anylogic

- Plot: conjunto de pares de valores <x.y> pudiendo mostrar puntos separados o conectando puntos con líneas, lo que permite representar varios conjuntos de datos simultáneamente.

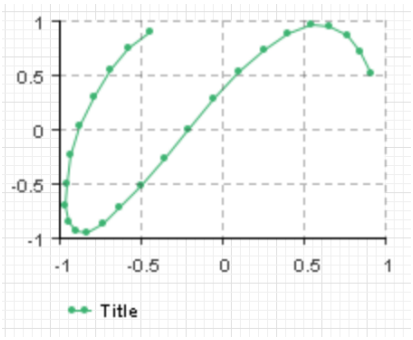

Figura 43: gráfico plot Anylogic

Variables, parámetros y variables creadas por código:

En Anylogic se pueden utilizar tanto variables como parámetros con la librería 'Agent' y que se incluyen dentro de la interfaz de simulación como se puede apreciar en la Figura 44:

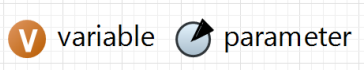

Figura 44: variables y parámetros Anylogic

Dentro de estos bloques se puede seleccionar el valor inicial y el tipo de variable o parámetro que se desea declarar disponiendo las siguientes opciones:

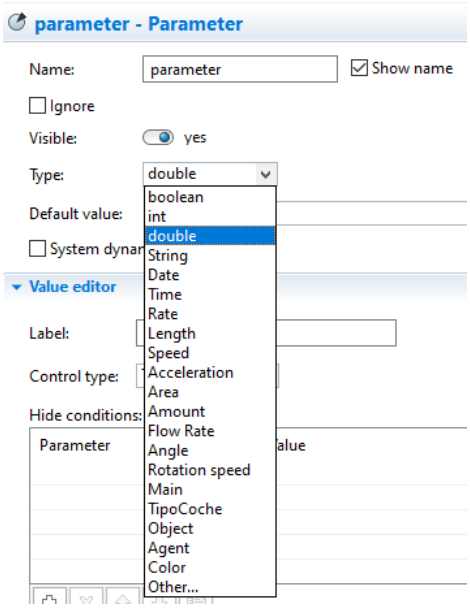

Figura 45: opciones tipos parámetros Anylogic

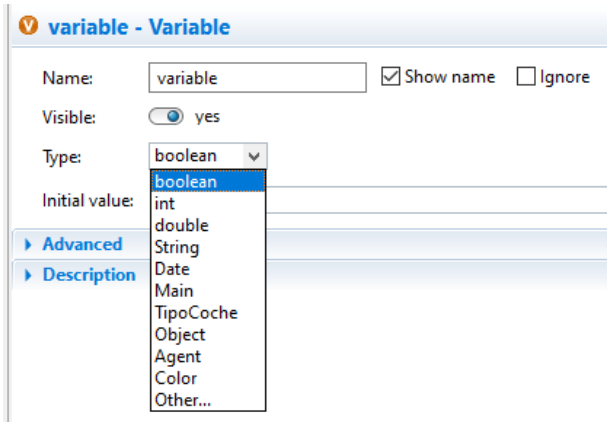

Figura 46: opciones tipos variables Anylogic

Las variables se suelen utilizar para guardar resultados del modelo de simulación o para modelar algunos datos o características de un objeto que van cambiando a lo largo del tiempo. Los parámetros son frecuentemente usados para representar algunas características del objeto modelado y son útiles cuando los objetos tienen el mismo comportamiento descrito en el tipo de agente, pero tiene algunos diferentes valores.

La principal diferencia entre parámetros y variables en Anylogic es que estas últimas representan un estado del modelo, que puede cambiar a lo largo de la simulación, mientras que los parámetros son usados para describir objetos estáticos, normalmente suelen ser una constante y se suele cambiar exclusivamente cuando se necesita ajustar el comportamiento del modelo (The AnyLogic Company, 2023).

Por otro lado, también se pueden declarar variables y listas de distintos tipos por código dentro de los distintos bloques. Se muestra un ejemplo de declaración de una variable tipo *double* con un valor tomado de una hoja de cálculo para determinar el Delay time de un bloque Delay:

```
(double) selectFrom(datos_salida_any)
.firstResult(datos_salida_any.tiempos_c1)
```

También se pueden declarar listas como en el siguiente ejemplo (lista de valores enteros) cuyos valores también han sido tomados de una hoja de cálculo:

```
List<Integer> tipoCocheDemandas = selectFrom(datos_salida_any)
    .list(datos_salida_any.tipo_coche_demanda);
```

Ajuste de la velocidad de tiempo de simulación:

Una vez ejecutado el modelo, dentro de la interfaz de simulación, el software permite ajustar la velocidad de la simulación. El número que se encuentra recuadrado, indica la relación entre la unidad de tiempo en el modelo de simulación y la unidad de tiempo real (‘Model time unit per real second’). Para incrementar la velocidad, se selecciona el círculo con el símbolo "+", mientras que para reducirla se utiliza el círculo con el símbolo "-". Si se desea alcanzar la máxima velocidad posible permitida por el rendimiento de tu ordenador (conocida como ‘virtual time mode’), se hace clic en el círculo que incorpora el símbolo de avance en su interior. Este proceso facilita que el usuario adapte la simulación a sus diferentes necesidades, permitiendo una observación detallada o una ejecución más rápida según los requerimientos específicos del análisis.

Figura 47: opciones velocidad de simulación Anylogic

Registro de los resultados de simulación:

El software de forma nativa permite crear un registro de los resultados obtenidos en la simulación. Para poder obtener este registro se debe activar la opción dentro de la ventana de proyectos en el apartado base de datos, en la ventana de propiedades activar la opción 'Log model execution'.

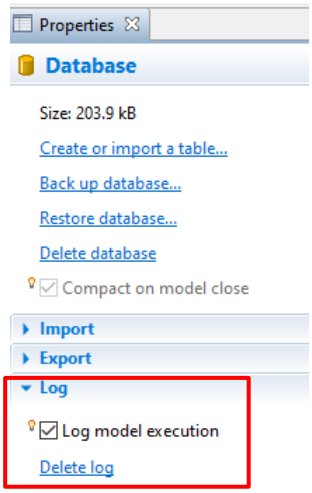

Figura 48: propiedades base de datos Anylogic

Una vez que se ha simulado el modelo, se obtiene una lista de registros. Si no se desea registrar alguno de ellos, se hace clic en el registro que no se quiere incluir y en su ventana de propiedades, se activa la opción 'I do not need this type of log'. De esta manera, estos datos no se registran y en el desplegable de registro aparecerá en la lista con un color más claro.

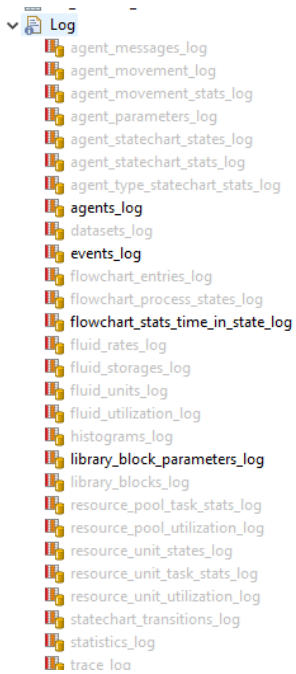

Figura 49: registros de la simulación Anylogic

Algunos de estos registros son:

- agent_parameters_log: almacena los parámetros iniciales de los agentes del modelo.
- agents_log: registra el tiempo de creación y el de destrucción de todos los agentes (para usuarios avanzados).
- agent_movement_stats_log: recopila la estadística de movimiento acumuladas para cada gente que se ha desplazado. Este movimiento se registra solo si se ha utilizado el bloque MoveTo o la función MoveTo().
- event_log: contiene información sobre los eventos que se producen en la simulación.
- flowchart_process_states_log: estadísticas que muestran el tiempo que pasan los agentes en diferentes estados, como pueden ser los tiempos de espera o de procesamiento.
- histograms_log: almacena datos de histograma que se recopilan a lo largo de la simulación del modelo.
- Statistics_log: recopila todos los datos estadísticos del modelo durante su ejecución (es necesario que tengan la propiedad registrar en la base de datos).
- library_block_parameters_log: registra los parámetros iniciales de los bloques utilizados en el modelo.

Por otro lado, con la versión estudiante, se puede generar tablas dentro de la base de datos e

introducir mediante código datos creados en tu propio modelo.

Toda esta información facilita a la comprensión del modelo creado y sus resultados.

Exportación de datos:

Tanto los datos del registro generado por Anylogic como los que el usuario puede generar, pueden ser exportados a un excel. Para que esto sea posible, es necesario acceder al apartado base de datos dentro del proyecto. Dentro su ventana de propiedades activar la opción 'Export tables at the end of model execution' y, a su vez, seleccionar dentro de las opciones de exportación las tablas y registros que se desean exportar a un archivo Excel previamente creado.

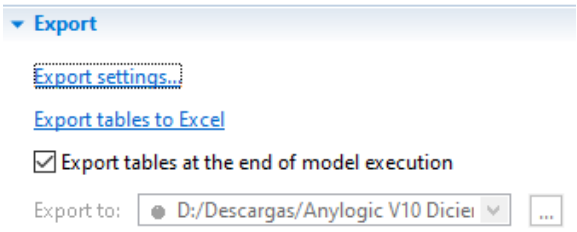

Figura 50: exportación resultados Anylogic

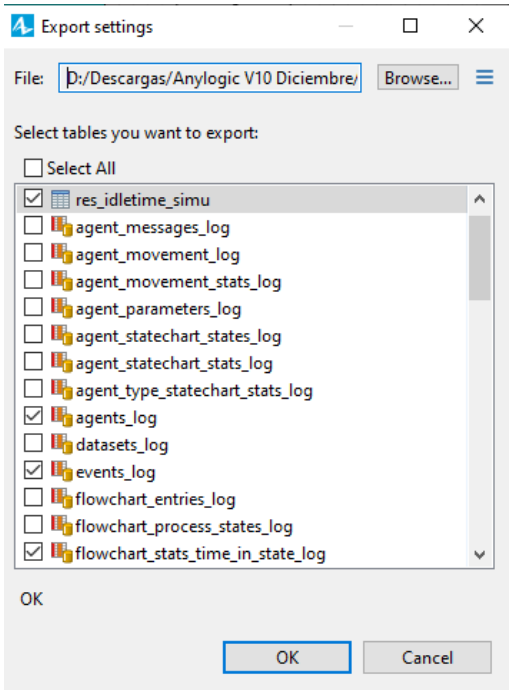

Figura 51: configuración exportación datos resultados Anylogic

4.2 Consideraciones previas

- Todo el modelo ha sido desarrollado en la versión estudiante de Anylogic, esto implica ciertas limitaciones en las paletas propias del programa (algunas características como escribir mensajes por pantalla, conexiones con bases de datos, opciones de simulación, etc).
- Todos los datos de entrada para Anylogic son tomados del Excel llamado 'DatosEntradaCoche', de una hoja llamada 'DatosSalidaAny' que recoge todos los datos previamente introducidos por el usuario y que han sido procesados por el Excel.
- La gran mayoría de códigos que se explicarán a continuación serán los correspondientes a la estación 1, siendo análogos en el resto de las estaciones.
- A veces las tablas que se importan del Excel al Anylogic no se visualizan correctamente (aparecen algunas celdas en blanco), pero sí están correctamente importadas.
- La simulación comienza a las 00:00 del 01/01/2024
- Al escribir código dentro de los bloques, no funcionan correctamente los condicionales 'else if' dentro de los bloques 'Exit' y 'Delay' no funcionan correctamente, por lo que se han utilizado 'if' para realizar esa función. Tambien se producen errores en los registros que genera Anylogic al finalizar la simulación (log), ninguno de estos afecta a la simulación ni a el resto de los resultados exportados. Es posible que esto se deba a que el modelo se ha ido desarrollando a la vez que se han producido varias actualizaciones del software.

4.3 Previo a la simulación

Antes de iniciar la simulación, es necesario vincular la base de datos con el excel 'DatosEntradaCoche' y también configurar para que se exporten los datos de salida al excel 'ResultadoSimulacionAnyLogic'. Para registrar los datos de salida de Anylogic en Excel, se requiere hacer una simulación completa si no existe ningún registro previo. Por lo tanto, primero se tiene que realizar la simulación y, posteriormente, completar la configuración para la exportación de los datos de salida. A continuación, se explica paso a paso como hacer ambas vinculaciones:

- Vinculación base datos Anylogic con Excel 'DatosEntradaCoche': es la primera acción a realizar al abrir modelo de simulación por primera vez. Acceder al apartado de la Database en

la ventana proyectos, clic derecho → New → Database table → Import database table(s) → seleccionar el archivo Excel en el apartado File → Show list of tables → seleccionar solo la tabla DatosSalidaAny → clic en 'Update data on the model startup' para que se actualice sin necesidad de repetir esos pasos.

En el caso de que salga una señal de aviso al seleccionar la tabla DatosSalidaAny elegir la opción Replace tables.

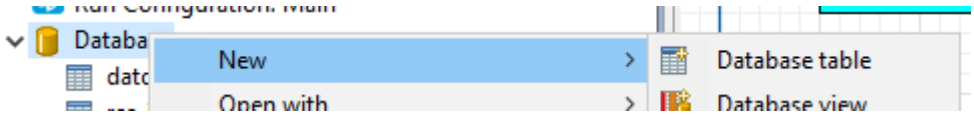

Figura 52: importar Excel a base datos Anylogic (1)

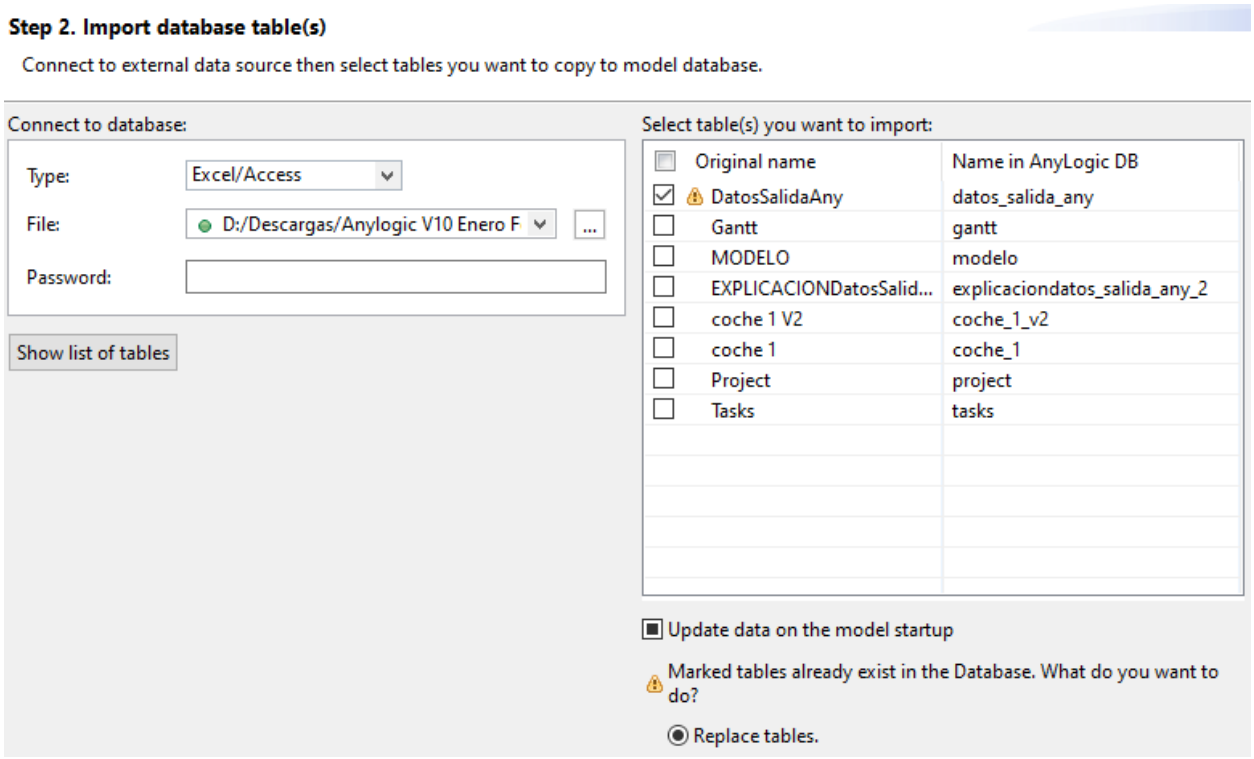

Figura 53: importar Excel a base datos Anylogic (2)

- Exportación base de datos y registro de Anylogic a Excel ResultadoSimulacionAnyLogic': acceder al apartado de la Database en la ventana proyectos, abrir su ventada de propiedades y

acceder a la sección Export → Export settings → buscar con el botón Browse el archivo Excel que se va a sobreescribir → seleccionar las tablas res_idletime_simu / agents_log / events_log/ flowchart_process_states_log/flowchart_stats_time_in_state_log/ library_block_parameters_log /statistics_log

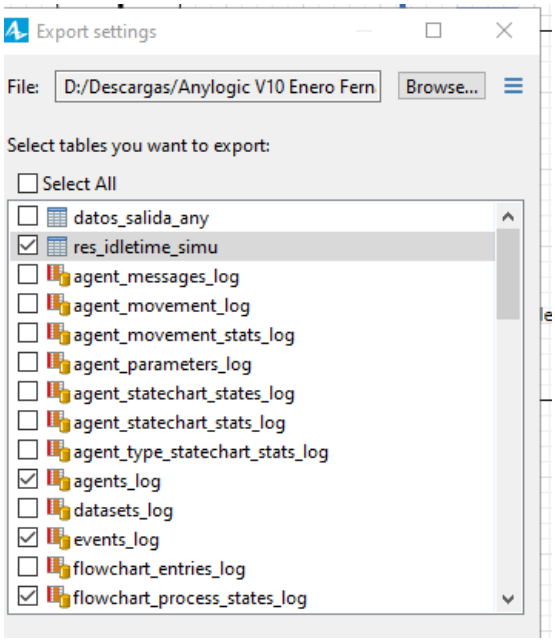

Figura 54: registros y tablas exportadas Anylogic

4.4 Agente tipo coche

Para poder llevar a cabo la simulación, se ha creado un tipo de agente llamado 'TipoCoche'. Este agente tiene las siguientes características:

-IdCoche: código único que dispone cada coche que sirve para identificarlo a lo largo de los diferentes bloques en la simulación (como un Vehicle Identification Number). Se trata de una variable tipo 'int' y es utilizada en los dos bloques Source de las demandas.

-OrdenEstC1/ OrdenEstC2/ OrdenEstC3: tres parámetros diferentes que indican el orden por el que pasa el agente (cada tipo de coche tiene un orden de estaciones diferentes). Estos parámetros son de tipo (`List<Integer>`), una lista de valores enteros.

-IndiceEstC1/ IndiceEstC2/ IndiceEstC3: Se emplean tres parámetros de tipo 'int' para gestionar el paso de cada tipo de coche por una estación. En los bloques de salida ('Exit') de cada estación, se incrementa en una unidad el valor del parámetro correspondiente cada vez

que un coche sale de la estación.

-TipoDeCoche: parámetro de tipo 'String' que permite identificar qué tipo de coche es. La cadena de caracteres se escribe en los bloques Source dependiendo de la variable 'valor', si valor = 1 se escribe la cadena 'Coche1' (al igual que para los otros dos tipos).

4.5 Generación de agentes

La generación de agentes se realiza mediante el bloque 'source'. En el modelo se encuentran dos de estos bloques, uno para la demanda aleatoria y otro para la demanda específica.

Figura 55: zona producción agentes Anylogic

Para que este bloque funcione correctamente datos del Excel de datos del coche y se llevaran a cabo las siguientes acciones:

-Tiempo entre llegadas: lo que tarda en generarse un agente. Para el caso de demanda específica ese valor lo da el usuario, para el de aleatoria viene dada por el valor del usuario y una pequeña desviación dada por una distribución.

- Específica:

```
(int) selectFrom(datos_salida_any)
.firstResult(datos_salida_any.takt_time
```

El valor de la variable 'takt_time' determina cuando se produce un agente.

- Aleatoria:

```
normal(5, 4)+ ((int) selectFrom(datos_salida_any)
   .firstResult(datos_salida_any.takt_time))
//añadimos esa variabilidad con normal(5,4)
```

-Agentes por llegada: número de agentes que se generan por cada simulación. Para este parámetro se utilizan códigos diferentes para la demanda aleatoria y la específica.

- Específica:

```
(int) selectFrom(datos_salida_any)
.firstResult(datos_salida_any.tipo_de_demanda)== 1 ? 0 : 1
```

La última línea del anterior código sirve para diferenciar la producción de agentes aleatoria o específica. Si el resultado obtenido de la variable 'tipo_de_demanda' es igual a 1, entonces el valor de toda la expresión será 0 (el usuario indicó que la demanda tiene que ser aleatoria, por tanto el bloque SourceEspecifico no producirá agente). De lo contrario, si el resultado no es igual a 1, el valor será 1 (el bloque SourceEspecifico sí producirá agentes).

- Aleatoria:

```
(int) selectFrom(datos_salida_any)
.firstResult(datos_salida_any.tipo_de_demanda)== 1 ? 1 : 0
```

Código similar al de la específica pero cuando 'tipo_de_demanda' es igual a 1, entonces el valor de toda la expresión será 1.

-Número máximos de llegadas: para limitar el número de agentes que se producen en cada simulación, se ha de activar la opción 'Limited number of arrivals'. Para indicar el valor de los coches que se van a producir, se toma de la hoja Excel de datos de entrada la variable 'total_coches' para la demanda específica y 'total_coches_aleatorio' para la aleatoria, cada una tiene respectivamente el valor total de coches que se desean producir.

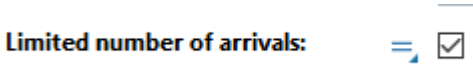

Figura 56: opción limitar número agentes bloque Source Anylogic

- Específica:

```
(int) selectFrom(datos_salida_any)
.firstResult(datos_salida_any.total_coches)
```

- Aleatoria:

```
(int) selectFrom(datos_salida_any)
.firstResult(datos_salida_any.total_coches_aleatorio)
```

- Acción en la salida caso demanda específica:

Se comienza seleccionando una lista de valores de tipo de coche demandado (‘tipoCocheDemandas’) desde una hoja del Excel de datos de entrada llamada ‘datos_salida_any’. Esta lista puede contener valores nulos, por lo que se crea una nueva lista llamada ‘tipoCocheDemandasNoNulos’ que contiene solo los valores no nulos de la lista original.

Después, se verifica si la lista de demandas de tipo de coche no nulos (‘tipoCocheDemandasNoNulos’) no es nula y no está vacía. Si esto se cumple, se procede a realizar acciones dependiendo del valor de ‘IndiceCreacionCoche’.

Si ‘IndiceCreacionCoche’ excede el tamaño de la lista ‘tipoCocheDemandasNoNulos’, se reinicia a 0 para asegurar un ciclo continuo a través de los valores de la lista.

Luego, se obtiene el valor en la posición indicada por ‘IndiceCreacionCoche’ de la lista ‘tipoCocheDemandasNoNulos’. Dependiendo de este valor, se asigna al agente un tipo de coche específico ("Coche1", "Coche2" o "Coche3"). Además, se extrae la secuencia de estaciones correspondiente al tipo de coche asignado desde el conjunto de datos y se asigna al agente. También se establece el índice de estación para el tipo de coche en 0. Posteriormente, se incrementa el contador de demanda (‘ConteoDemanda’) y se asigna al agente un identificador único de coche (‘IdCoche’).

Finalmente, se incrementa el índice de creación de coche (‘IndiceCreacionCoche’) para continuar con el siguiente ciclo.

En resumen, este bloque de código produce la demanda de tipo de coche, asigna un tipo de coche

específico al agente, y realiza otras acciones relacionadas al salir del bloque de simulación.

```
List<Integer> tipoCocheDemandas = selectFrom(datos_salida_any)
    .list(datos_salida_any.tipo_coche_demanda);

// Crear un vector auxiliar que contenga solo los elementos no nulos del vector
tipoCocheDemandas
List<Integer> tipoCocheDemandasNoNulos = new ArrayList<>();
for (Integer valor : tipoCocheDemandas) {
    if (valor != null) {
        tipoCocheDemandasNoNulos.add(valor);}
}
if (tipoCocheDemandasNoNulos!= null && !tipoCocheDemandasNoNulos.isEmpty()) {
    if (IndiceCreacionCoche >= tipoCocheDemandasNoNulos.size()) {
            IndiceCreacionCoche = 0;

    }
    Integer valor = tipoCocheDemandasNoNulos.get(IndiceCreacionCoche);

    if (valor != null) {
        if (valor == 1) {

          agent.TipoDeCoche = "Coche1";

        List<Integer> OrdenEstCoche1 = selectFrom(datos_salida_any)
          .list(datos_salida_any.orden_est_c1); // vector OrdenEstCoche1
          List<Integer> OrdenEstCoche1NoNulos = new ArrayList<>();
          for (Integer val : OrdenEstCoche1) {
              if (val != null) {
                  OrdenEstCoche1NoNulos.add(val);
              }
          }
        agent.OrdenEstC1 = OrdenEstCoche1NoNulos;
        agent.IndiceEstC1 = 0;
        //Para saber el número exacto del coche producido se usa ConteoDemanda
       ConteoDemanda++;
       agent.IdCoche=ConteoDemanda;

        } else if (valor == 2) {
            agent.TipoDeCoche = "Coche2";
```

```
                    //
        List<Integer> OrdenEstCoche2 = selectFrom(datos_salida_any)
          .list(datos_salida_any.orden_est_c2); // vector OrdenEstCoche1
          List<Integer> OrdenEstCoche2NoNulos = new ArrayList<>();
          for (Integer val : OrdenEstCoche2) {
              if (val != null) {
                  OrdenEstCoche2NoNulos.add(val);
              }
          }
        agent.OrdenEstC2 = OrdenEstCoche2NoNulos;
        agent.IndiceEstC2 = 0;
          //Para saber el número exacto del coche producido se usa ConteoDemanda
       ConteoDemanda++;
       agent.IdCoche=ConteoDemanda;
        //
        } else if (valor == 3) {
            agent.TipoDeCoche = "Coche3";
                  //
        List<Integer> OrdenEstCoche3 = selectFrom(datos_salida_any)
          .list(datos_salida_any.orden_est_c3); // vector OrdenEstCoche1
          List<Integer> OrdenEstCoche3NoNulos = new ArrayList<>();
          for (Integer val : OrdenEstCoche3) {
              if (val != null) {
                  OrdenEstCoche3NoNulos.add(val);
              }
          }
        agent.OrdenEstC3 = OrdenEstCoche3NoNulos;
        agent.IndiceEstC3 = 0;
        //
          //Para saber el número exacto del coche producido se usa ConteoDemanda
       ConteoDemanda++;
       agent.IdCoche=ConteoDemanda;
            }
    }
}

IndiceCreacionCoche++;
```

- Acción en la salida caso demanda aleatoria:

el siguiente código realiza la asignación aleatoria de un tipo de coche a un agente en la salida de un proceso, según un número aleatorio generado en el rango de 1 a 3 utilizando la función 'uniform_discr(1, 3)'. Se utilizan condicionales 'if-else' para determinar qué tipo de coche se asignará al agente, dependiendo del número aleatorio generado:

- Si el número aleatorio es 1, se asigna al agente el tipo de coche "Coche1". Se extrae del conjunto de datos la secuencia de estaciones para el Coche1, se eliminan los valores nulos de la secuencia y se asigna al agente la secuencia de estaciones y un índice inicial de 0 para esa secuencia. Además, se incrementa el contador de demanda ('ConteoDemanda') y se asigna al agente un identificador único de coche ('IdCoche').
- Si el número aleatorio es 2, se realiza un proceso similar para el tipo de coche "Coche2".
- Si el número aleatorio es 3, se realiza un proceso similar para el tipo de coche "Coche3".

En pocas palabras, este bloque de código asigna de forma aleatoria un tipo de coche al agente en la salida del proceso, junto con la secuencia de estaciones correspondiente y un identificador único de coche.

```
int numeroAleatorio = uniform_discr(1, 3);

if (numeroAleatorio == 1) {
    agent.TipoDeCoche = "Coche1";
    //
  List<Integer> OrdenEstCoche1 = selectFrom(datos_salida_any)
.list(datos_salida_any.orden_est_c1); // vector OrdenEstCoche1
      List<Integer> OrdenEstCoche1NoNulos = new ArrayList<>();
      for (Integer val : OrdenEstCoche1) {
          if (val != null) {
              OrdenEstCoche1NoNulos.add(val);
          }
      }
  agent.OrdenEstC1 = OrdenEstCoche1NoNulos;
  agent.IndiceEstC1 = 0;
    //Para saber el número exacto del coche producido se usa ConteoDemanda
    ConteoDemanda++;
    agent.IdCoche=ConteoDemanda;
```

```
        } else if (numeroAleatorio == 2) {
            agent.TipoDeCoche = "Coche2";

            //
          List<Integer> OrdenEstCoche2 = selectFrom(datos_salida_any)
        .list(datos_salida_any.orden_est_c2); // vector OrdenEstCoche1
              List<Integer> OrdenEstCoche2NoNulos = new ArrayList<>();


              for (Integer val : OrdenEstCoche2) {
                  if (val != null) {
                      OrdenEstCoche2NoNulos.add(val);
                  }
              }
          agent.OrdenEstC2 = OrdenEstCoche2NoNulos;
          agent.IndiceEstC2 = 0;
            //Para saber el número exacto del coche producido se usa ConteoDemanda
            ConteoDemanda++;
            agent.IdCoche=ConteoDemanda;

        } else if (numeroAleatorio == 3) {
            agent.TipoDeCoche = "Coche3";

            //
          List<Integer> OrdenEstCoche3 = selectFrom(datos_salida_any)
        .list(datos_salida_any.orden_est_c3); // vector OrdenEstCoche1
              List<Integer> OrdenEstCoche3NoNulos = new ArrayList<>();


              for (Integer val : OrdenEstCoche3) {
                  if (val != null) {
                      OrdenEstCoche3NoNulos.add(val);
                  }
              }
          agent.OrdenEstC3 = OrdenEstCoche3NoNulos;
          agent.IndiceEstC3 = 0;
```

```
        //Para saber el número exacto del coche producido se usa ConteoDemanda
        ConteoDemanda++;
        agent.IdCoche=ConteoDemanda;}
```

4.6 Estaciones de trabajo

Existen un total de 10 estaciones de trabajo por las que los coches podrían pasar.

Cada estación de trabajo está compuesta por los siguientes elementos:

- Bloque Enter y Exit
- Bloque TimeMeasure (inicial y final)
- Bloque wait
- Bloque selectOutput5
- Bloque Delay

Todos estos elementos en su conjunto se pueden ver en la siguiente Figura 57 (el resto de las estaciones tienen la misma estructura que las dos que se muestran) y serán explicadas sus funciones en los subapartados.

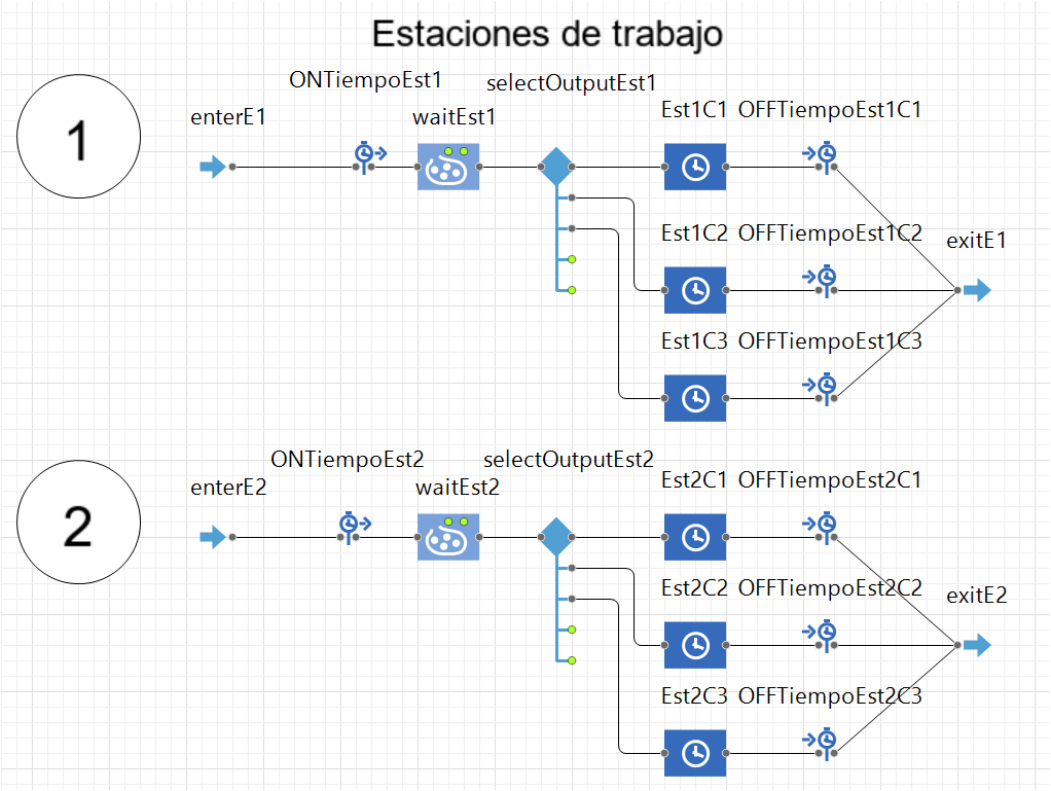

Figura 57: estaciones de trabajo Anylogic

4.6.1 Entrada y salida de la estación

- **Entrada:** permite la entrada de los agentes en la estación con el bloque Enter, dentro del bloque se selecciona el agente TipoCoche.

- **Salida:**

  El objetivo principal de este fragmento de código es dirigir el flujo de agentes a través de diferentes estaciones de producción, dependiendo del tipo de coche que estén manejando.

  Primero, se verifica el tipo de coche (Coche1, Coche2, Coche3). Dependiendo del tipo de coche y el valor asignado para la siguiente estación de destino, el agente es dirigido a la estación correspondiente utilizando else if. Si el valor de la próxima estación es -1, indica que el agente debe dirigirse a la estación de Producto Terminado.

  Después de gestionar la distribución de los agentes, se libera el primer agente situado en la cola de la estación si realmente hay agentes esperando.

  Finalmente, se controla el tiempo de inactividad de la estación. Si no hay coches en espera en ninguna de las tres colas de la estación, se registra el tiempo de inicio del tiempo de inactividad. Esto se hace para contabilizar el tiempo en que la estación está inactiva mientras no hay agentes que estén siendo procesados.

```
if (agent.TipoDeCoche.equals("Coche1")) {
      int valorC1= agent.OrdenEstC1.get(agent.IndiceEstC1);
      //Valor que indica la siguiente estación a la que va el coche
      //Se le suma 1 unidad al índice para saber que ha pasado por una estación
        if(valorC1==1){ enterE1.take(agent); agent.IndiceEstC1++;}
        else if(valorC1==2){enterE2.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==3){enterE3.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==4){enterE4.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==5){enterE5.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==6){enterE6.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==7){enterE7.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==8){enterE8.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==9){enterE9.take(agent);agent.IndiceEstC1++;}
        else if(valorC1==10){enterE10.take(agent);agent.IndiceEstC1++;}
```

```
        //Si toma valor -1 u otro diferente va a la estación Producto Terminado

        else if(valorC1==-1){enterFinal.take(agent);agent.IndiceEstC1=0;}
        else{enterFinal.take(agent);agent.IndiceEstC1=0;}
}
if (agent.TipoDeCoche.equals("Coche2")) {
      int valorC2= agent.OrdenEstC2.get(agent.IndiceEstC2);
        if(valorC2==1){enterE1.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==2){enterE2.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==3){enterE3.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==4){enterE4.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==5){enterE5.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==6){enterE6.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==7){enterE7.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==8){enterE8.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==9){enterE9.take(agent);agent.IndiceEstC2++;}
        else if(valorC2==10){enterE10.take(agent);agent.IndiceEstC2++;}

        else if(valorC2==-1){enterFinal.take(agent);agent.IndiceEstC2=0;}
         else{enterFinal.take(agent);agent.IndiceEstC2=0;}
}
if (agent.TipoDeCoche.equals("Coche3")) {
      int valorC3= agent.OrdenEstC3.get(agent.IndiceEstC3);
        if(valorC3==1){enterE1.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==2){enterE2.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==3){enterE3.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==4){enterE4.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==5){enterE5.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==6){enterE6.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==7){enterE7.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==8){enterE8.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==9){enterE9.take(agent);agent.IndiceEstC3++;}
        else if(valorC3==10){enterE10.take(agent);agent.IndiceEstC3++;}

        else if(valorC3==-1){enterFinal.take(agent);agent.IndiceEstC3=0;}
             else{enterFinal.take(agent);agent.IndiceEstC3=0;}
}
//////////////////// LIBERACION DE ESTACION 1 ////////////////////////////////////
```

```
if (waitEst1.size()>0){waitEst1.free(waitEst1.get(0));} //Permite la salida del
primer agente del bloque wait indicado

////////////////////////// IDLE TIME DE ESTACION 1
//////////////////////////////////////

if(Est1C1.size() == 0 && Est1C2.size() ==  0 && Est1C3.size() ==0){StartTimeEst1 =
time();} //Inicio para contabilizar el Idle Time
```

4.6.2 Contabilización tiempos estación y coches

Para registrar el tiempo que tarda cada coche en pasar por cada estación, considerando los tiempos de espera en los bloques Wait, y el tiempo total que tarda cada coche, se utilizan los bloques TimeMeasureStart y TimeMeasureEnd para iniciar y finalizar la medición respectivamente. A continuación, se explican donde han sido utilizados en este modelo:

- TimeMeasureStart: ONTiempoEst$_i$ para contabilizar el tiempo total que está cada coche en cada estación que pasa, localizados justo antes de los bloques waitEst$_i$.

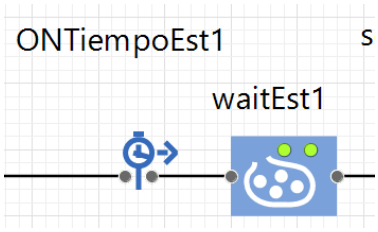

Figura 58: bloque ONTiempoEst$_i$ Anylogic

Para registrar el tiempo desde que los coches salen de los bloques Source, para demanda específica y aleatoria se utilizan ONTiempoEsp y ONTiempoDemAle respectivamente. Localizados justo después de ambos bloques Source

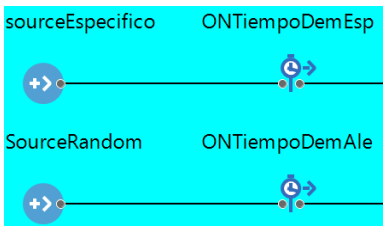

Figura 59: bloque ONTiempoDemEsp y ONTiempoDemAle Anylogic

- TimeMeasureEnd: OFFTiempoEst$_i$C$_j$ para señalar el final de la contabilización en cada estación relacionados con los bloques ONTiempoEst$_i$. Se localizan justo después de los bloques Delay, es decir, antes de salir de las estaciones de trabajo.

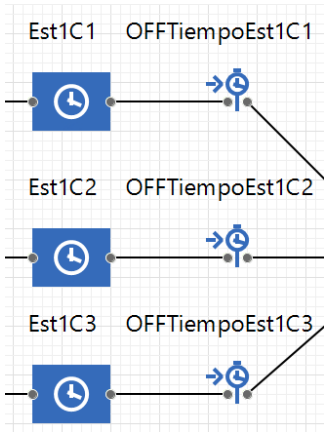

Figura 60: bloque OFFTiempoEst$_i$C$_j$ Anylogic

- Para finalizar el registro del tiempo total de cada coche para ambos tipos de demanda, se utiliza el bloque TiempoTotal. Localizado justo antes del bloque Sink que finaliza la producción.

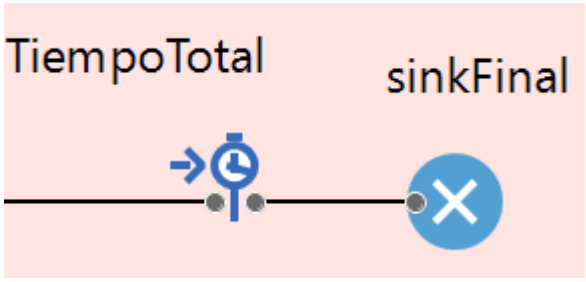

Figura 61: bloque TiempoTotal Anylogic

Todos estos resultados serán almacenados una vez se finalice la simulación dentro del registro que genera Anylogic en su base de datos, en el subregistro 'statistics_log' que se explicará en este mismo capítulo en el subapartado 4.13.1Registro automático Anylogic

4.6.3 Colas estaciones

Las colas de las estaciones vienen dadas por el bloque 'Wait', este permite retener los agentes hasta

que se cumpla una condición. Para este caso la condición que se ejecutará en las acciones de entrada del bloque será:

- **Acción a la entrada del bloque:**

Este bloque de código verifica si la estación está en su estado inicial o si no hay coches esperando en ninguna de las tres colas de la estación. Si alguna de estas condiciones es verdadera, libera al agente de la espera en la estación.

```
if (IndiceInicialEst1==0 || (Est1C1.size() == 0 && Est1C2.size() == 0 &&
Est1C3.size() ==0)) {
waitEst1.free(agent);
}
```

Esta cola tendrá una capacidad determinada por el usuario en el Excel. Si esta cantidad se excede, es decir, si se excede la cantidad de agentes en espera, la variable 'ColaSaturadaEst1' (en el caso de la estación 1) se incrementará una unidad. El código en Java dentro del bloque es el siguiente:

```
else if (waitEst1.size() ==(int) selectFrom(datos_salida_any)
   . firstResult (datos_salida_any. capacidad_de_las_estaciones)) {
   ColaSaturadaEst1++; }
```

- **Acción a la salida del bloque:**

Se controla y registra el tiempo de inactividad de una estación. Su principal lógica consiste en determinar si la estación está ocupada o libre en el momento en que el agente sale del bloque y registrar el tiempo de inactividad correspondient. Además, al final se realiza un registro en la hoja de cálculo de salida con los resultados cuando se detecta que ha habido tiempo de inactividad y se registra el acumulado de esa estación. Por otro lado también se indica el coche que es, el tiempo de simulación en que se produce e indica mediante una cadena de caracteres de qué estación se trata.

```
IndiceInicialEst1++;
if (IndOcupaEst1==0){
// Para que cuente el Idle time inicial
IdleTimeEst1 = time();
IdleTimeTotalEst1=IdleTimeTotalEst1+IdleTimeEst1;
```

```
}
//Se contabiliza el idle time al inicio ya que se empieza a producir y quedan coches
por
//producir, en cambio no se contabiliza una vez ha pasado el último coche por la
//estación ya que se considera que la estación se ha cerrado.
else if (IndOcupaEst1!=0) {
IdleTimeEst1 = time() - StartTimeEst1;
IdleTimeTotalEst1=IdleTimeTotalEst1+IdleTimeEst1;}

else if (IdleTimeTotalEst1 > 0) {
//Se introduce en el excel de resultados exportados la estación y tipo de coche,
tiempo total del idle time, el
//tiempo de la simulación en el que se graba el dato y el idle time total de esa
estación
      insertInto(res_idletime_simu)
          .columns(res_idletime_simu.estacion,res_idletime_simu.id_coche,
res_idletime_simu.total_idle_time,
res_idletime_simu.tiempo_simu,res_idletime_simu.idle_total_est1)
          .values("Est1",agent.IdCoche, IdleTimeEst1, time(),IdleTimeTotalEst1)
          .execute();
}
```

4.6.4 Producción estación

Para simular el tiempo que tarda cada coche en fabricarse en cada estación, se utiliza el bloque 'delay' que tiene los siguientes parámetros:

- **Tipo delay:**

  Se indica que el tiempo que está dentro el coche será un tiempo especificado en el código de tiempo de producción.

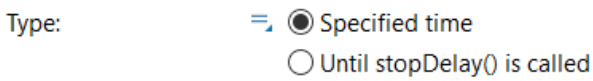

Figura 62: opciones Delay Anylogic

- **Tiempo de producción:**

  Tiempo que el coche tarda en montarse en la estación.

```
(double) selectFrom(datos_salida_any).firstResult(datos_salida_any.tiempos_c1)
```

- **Capacidad**:

  Número de coches que puede tener el bloque en su interior.

Figura 63: capacidad bloque Delay Anylogic

- **Acción a la salida del bloque:**

  Se sobrescribe la variable RatioOcupacion

```
RatioOcupacionEst1=IdleTimeTotalEst1/time(); //Se sobrescribe la variable
RatioOcupacion
```

4.7 Final producción

Una vez el coche ha pasado por las estaciones que le corresponde, entra por el bloque 'enterFinal' (bloque Enter) y termina en un bloque Sink 'sinkFinal'.

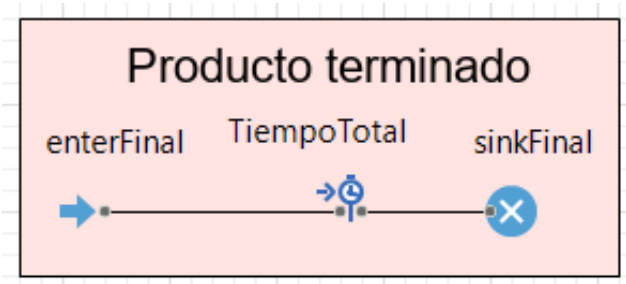

Figura 64: zona producto terminado Anylogic

Este último bloque sirve para contabilizar el número de coches terminados según su tipo, el total de coches y parar el motor de la simulación una vez se alcanza el total de coches producidos previamente

establecidos.

El código a la entrada del bloque es el siguiente:

```
if (agent.TipoDeCoche.equals("Coche1")) {
	//Contabilizar coches terminados de tipo 1
	IndiceTerminadosC1++;
	IndiceTotalCochesTerminados++;
}
else if (agent.TipoDeCoche.equals("Coche2")) {
	//Contabilizar coches terminados de tipo 2
	IndiceTerminadosC2++;
	IndiceTotalCochesTerminados++;
}
else if (agent.TipoDeCoche.equals("Coche3")) {
	//Contabilizar coches terminados de tipo 3
	IndiceTerminadosC3++;
	IndiceTotalCochesTerminados++;
}
else if((int)
selectFrom(datos_salida_any).firstResult(datos_salida_any.tipo_de_demanda)==0){
//Detener el motor para el caso de demanda especifica cuando se llega al total de
coches

	if (IndiceTotalCochesTerminados==(int) selectFrom(datos_salida_any)
	.firstResult(datos_salida_any.total_coches)) {
	getEngine().stop();//Se detiene el motor de la simulación al llegar al total
de coches
	}
}
else if((int)
selectFrom(datos_salida_any).firstResult(datos_salida_any.tipo_de_demanda)==1){
//Detener el motor para el caso de demanda aleatoria cuando se llega al total de
coches

	if (IndiceTotalCochesTerminados==(int) selectFrom(datos_salida_any)
	.firstResult(datos_salida_any.total_coches_aleatorio)) {

	getEngine().stop();//Se detiene el motor de la simulación al llegar al total
de coches
```

```
        }
}
```

4.8 Eventos

Para detener la simulación y poder observarla al llegar al límite de coches se han creado dos eventos. Uno relacionado con la cantidad total de coches en demanda específica y otro con la demanda aleatoria (estos valores se toman de la hoja de cálculo rellenada por el usuario).

- **Condición StopEngineEspecifica:**

  Detiene la simulación cuando se produzcan todos los coches en demanda específica

```
IndiceTotalCochesTerminados==(int) selectFrom(datos_salida_any)
    .firstResult(datos_salida_any.total_coches) &&
((int)selectFrom(datos_salida_any)
    .firstResult(datos_salida_any.tipo_de_demanda)== 0)
//Que se produzcan todos los coches y que esté en modo demanda Específica
```

- **Condición StopEngineAleatorio:**

  Detiene la simulación cuando se produzcan todos los coches en demanda aleatoria

-

```
(IndiceTotalCochesTerminados==(int) selectFrom(datos_salida_any)
    .firstResult(datos_salida_any.total_coches_aleatorio)) && ((int)
selectFrom(datos_salida_any)
    .firstResult(datos_salida_any.tipo_de_demanda)== 1)
//Que se produzcan todos los coches aleatorios y que esté en modo demanda Aleatoria
```

En la interfaz de simulación, los eventos aparecen visualmente de la siguiente manera:

Figura 65: eventos Anylogic

Cuando un evento se active, el icono del rayo se colorea de rojo:

Figura 66: evento activo

4.9 Gráficos

Para poder apreciar con facilidad los resultados de la simulación, se han implementado los siguientes gráficos.

- **Gráficos Idle Time:** se puede observar cómo evoluciona el idle time total de cada estación a lo largo del tiempo. Los valores se pueden observar en las leyendas de los gráficos.

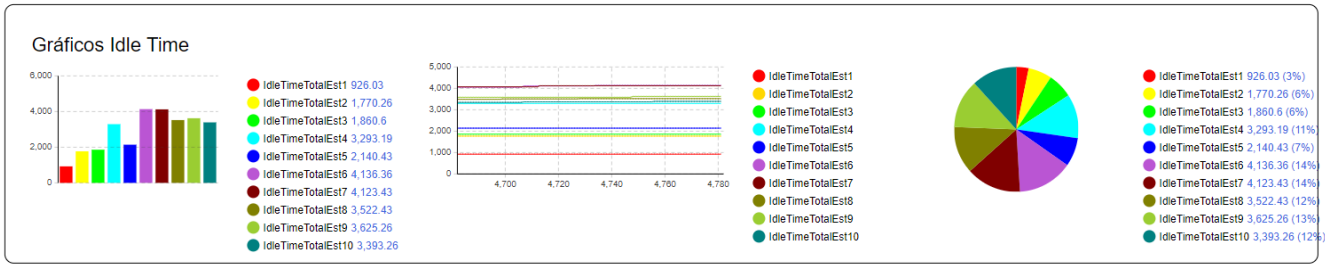

Figura 67: gráficos Idle Time Anylogic

- **Gráficos Saturación buffer:** permiten observar la evolución de las colas de las estaciones, se utiliza la variable ColaSaturadaEst$_i$ que viene directamente relacionada con la capacidad que el usuario ha decido darles a las estaciones (como se explica en el apartado **¡Error! No se e ncuentra el origen de la referencia. ¡Error! No se encuentra el origen de la referencia.**).

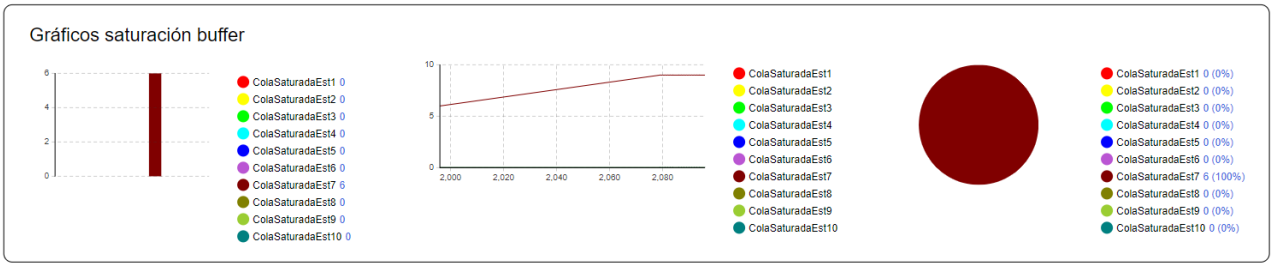

Figura 68: gráficos saturación buffer Anylogic

- **<u>Gráficos uso estaciones:</u>** permite observar que estación es la más utilizada.

Figura 69: gráfico circular uso estaciones Anylogic

- **<u>Ratio ocupación estaciones:</u>** representa el idle time total de cada estación respecto al tiempo de simulación.

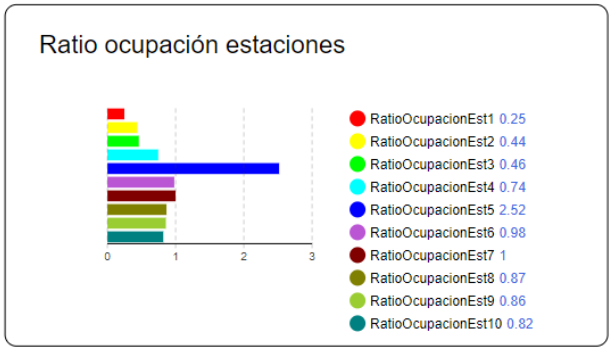

Figura 70: gráfico ratio ocupación estaciones Anylogic

- **Gráficos coches restantes:** permite observar el número de coches que quedan por producir (en la versión pro existe un bloque de representación para observar el avance de simulación, como se ha realizado en la versión estudiante esta es la manera de representarlo). En la interfaz de la simulación aparecerá un gráfico circular a la izquierda o a la derecha dependiendo del tipo de demanda seleccionada.

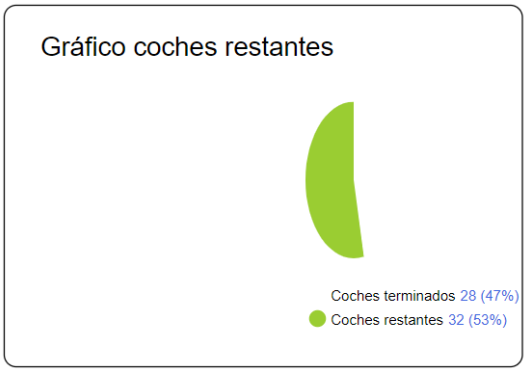

Figura 71:gráfico coches restantes demanda específica

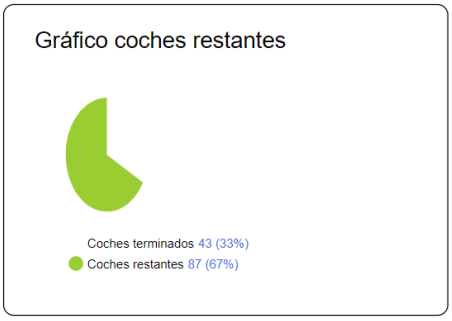

Figura 72: gráfico coches restantes demanda aleatoria

- **Ocupación de las estaciones por tipo de coche:** permite observar que tipo de coche es el que más ocupa cada estación (el eje 'y' indica el número total de coches)

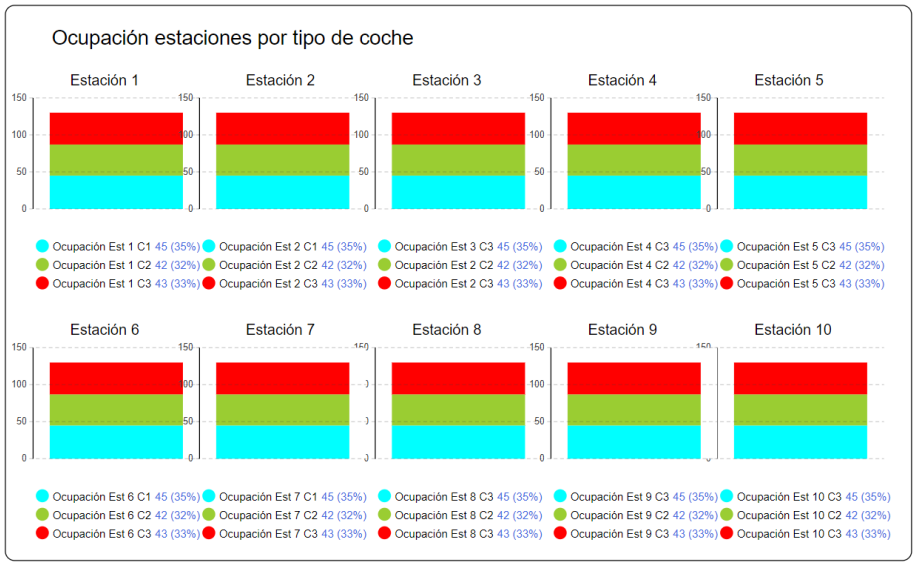

Figura 73: gráficos ocupación estaciones por tipo de coches

- **Gráficos coches terminados:** representa el total de coches terminados según su tipo

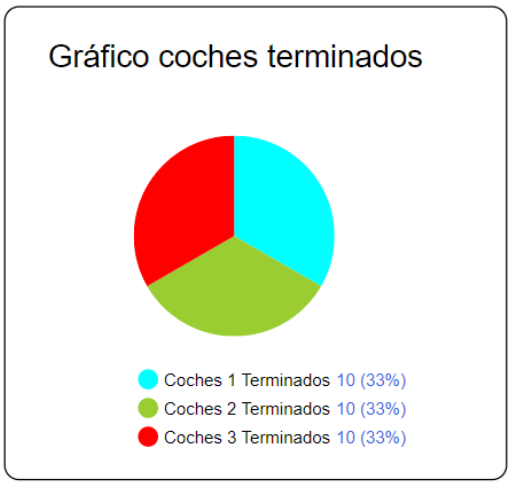

Figura 74: gráfico coches terminados

4.10 Finalizar, pausar y reanudar la simulación

Se puede detener la simulación con la propia interfaz de Anylogic, pero para que sea más sencillo y visual para el usuario se han implementado tres botones dentro de la interfaz de simulación del modelo.

- **Finalizar simulación:**

Para finalizar la simulación en el momento que el usuario desee, se ha incluido un botón interactivo dentro de la interfaz de simulación.

Figura 75: botón finalizar simulación Anylogic

Dentro de este botón, en el apartado acción se encuentra el código que detiene la simulación:

```
stopSimulation();
```

- **Pausar y reanudar simulación:**

Para detener la simulación, en el apartado acción de su respectivo botón se encuentra el siguiente código:

```
pauseSimulation();
```

Al igual ocurre con el botón Reanudar Simulación con el siguiente código:

```
runSimulation();
```

Figura 76: botones pausar y reanudar simulación Anylogic

4.11 Tiempo de simulación

Anylogic permite ver el tiempo de simulación desplegando el panel de simulador en su interfaz propia de simulación. Para que sea más sencillo se ha incluido en la interfaz del modelo un texto que se actualiza mostrando el tiempo de simulación como se puede observar en la siguiente Figura 77:

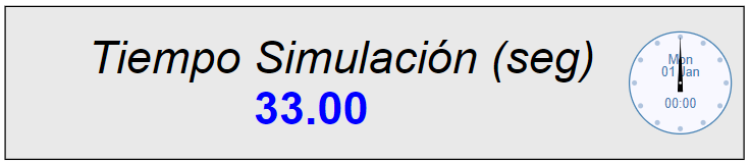

Figura 77: tiempo de simulación Anylogic

Para que este texto se actualice se utiliza el siguiente código dentro de la opción 'Dynamic value' del bloque texto:

```
String.format("%.2f", time()); //Se muestran solo dos decimales
```

El reloj que se muestra a la derecha en la anterior Figura 78 se va actualizando a lo largo de la simulación, indicando en su parte superior la fecha de la simulación y en la inferior las horas y minutos (se recuerda que la simulación comienza el día 1 de enero de 2024 a las 00:00)

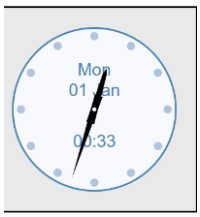

Figura 78: reloj dinámico simulación Anylogic

4.12 Indicadores utilizados

4.12.1 Índices ocupación estaciones

Existen cuatro variables por cada estación, tres que son las respectivas a los tipos de coche y una que engloba toda la estación. Todas son de tipo 'int' y su valor inicial es 0.

Permite conocer el número de veces que un tipo de coche ha entrado en una estación. Sus valores aumentan en una unidad a la salida de los bloques selectOutput$_i$ (justo antes de entrar en los bloques delay). Por ejemplo, para la estación de trabajo 1 si el parámetro TipoDeCoche del agente es 'Coche1' se cumple la condición de la salida 1 (del bloque selectOutputEst1) y a su salida IndOcupaEst1C1 e IndOcupaEst1 aumentan en una unidad.

Figura 79: indicadores índices ocupación Anylogic

4.12.2 Relacionada con la demanda

Se utiliza para llevar un conteo de los coches que han empezado a ser producidos, su valor inicial es 0 y su tipo es ‘int’.

Su valor aumenta una unidad en los bloques SourceRandom y sourceEspecífico en sus respectivas acciones de salida cada vez que se genera un agente.

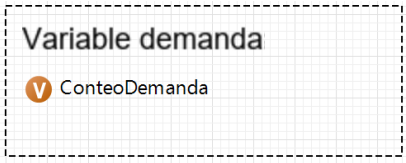

Figura 80: variable demanda Anylogic

4.12.3 Índices coches terminados

Para contabilizar de manera precisa los vehículos producidos (ya sea por tipo de vehículo o en total), se han declarado estos indicadores con un valor inicial de 0 y de tipo ‘int’. El valor de la variable 'IndiceTerminadosC$_i$' se incrementa una unidad en los bloques Sink, diferenciando cada variable por el tipo de vehículo terminado. A su vez, la variable 'IndiceTotalCochesTerminados' aumenta uno su valor independientemente del tipo de vehículo, con el fin de contabilizar el total de vehículos terminados.

Figura 81: indicadores índices coches terminados Anylogic

4.12.4 Índices iniciales estaciones

Su principal función es la de liberar del bloque Wait del primer agente que entra en una estación, también permite contabilizar cuantos coches entran en las estaciones a lo largo de la simulación. Sus valores iniciales son 0 y son de tipo 'int'.

Figura 82: indicadores índices iniciales estaciones Anylogic

4.12.5 Parámetros cola saturada estaciones

Permiten identificar cuando una estación alcanza su capacidad máxima, pudiendo así analizar cuáles son las más congestionadas. Sus valores iniciales establecidos son 0 y son de tipo 'int'.

Cuando en un bloque wait de una estación se alcanza el valor de la capacidad dada por el usuario, esta variable se incrementa en una unidad.

Figura 83: indicadores colas saturadas Anylogic

4.12.6 Variable ratio ocupación

Esta variable relaciona el idle time total de cada estación con el tiempo de simulación siguiendo la siguiente fórmula:

$$RatioOcupacionEst_i = \frac{IdleTimeTotal_i}{time} \text{ siendo i la estación } i \in 1..10$$

Sirve para identificar cuáles son las estaciones más ocupadas y posibles problemas que esto puede conllevar (cuanto menor sea este valor, más ocupada está la estación).

Estos valores se actualizan en la salía de los bloques delay respectivos de cada estación.

Figura 84: indicadores ratio ocupación Anylogic

4.12.7 Indicadores Idle Time

Se han creado 3 tipos de indicadores para recopilar datos del tiempo ocioso de cada estación (30 indicadores en total). Es importante destacar que se ha considerado tiempo ocioso de la estación desde el instante que se inicia la simulación. Por ejemplo, si no llega ningún coche a la estación 10 hasta los 90 segundos, se considera como tiempo ocioso.

Los indicadores se detallan a continuación:

- StartTimeEst$_i$: inicia la contabilización del tiempo ocioso y el código se encuentra en las acciones de salida de los bloques 'exitE$_i$'. Existe una variable para cada estación, son de tipo 'double' con valores iniciales 0.
- IdleTimeEst$_i$: su valor indica el tiempo ocioso de la estación desde ha salido un coche hasta que entra otro. No acumula el tiempo total, ya que se actualiza a lo largo de la simulación. Valores iniciales 0 y tipo 'double'. El código que permite actualizar la variable se encuentra en las acciones de salida de los bloques 'waitEst$_i$ '.
- IdleTimeTotalEst$_i$: contabiliza el tiempo ocioso total de cada estación. Su valor se obtiene sumando su valor anterior con el actual de la variable IdleTimeEst$_i$ como se puede ver en el código que se encuentra en las acciones de salida de los bloques 'waitEst$_i$ '. Son de tipo 'double' con valores iniciales 0.

Figura 85: indicadores Idle time Anylogic

4.13 Resultados exportados de la simulación

Para poder ver fácilmente algunos resultados obtenidos en la simulación, se ha creado un Excel

llamado 'ResultadoSimulacionAnyLogic' , este se sobrescribirá cada vez que se realice una simulación completa. Se exportarán datos creados automáticamente al activar la opción de registro de Anylogic, así como datos sobre el tiempo ocioso generados mediante código y sus indicadores asociados. Se recomienda cerrar la ventana de la simulación para exportar correctamente los resultados en el excel.

4.13.1 Registro automático Anylogic

Anylogic permite de forma nativa genera algunos registros relacionados con eventos y datos obtenidos de los bloques utilizados. Solo se exportarán aquellos realmente útiles para comprender el modelo, son los siguientes:

- Agents_log: permite ver cuando el agente ha sido creado con la columna 'birthdate' y cuando ha llegado al final con la columna 'deathdate', para saber de qué agentes se trata la columna 'Type' dice qué tipo es y en la columna 'name' se pueden ver que son distintos agentes (el primer agente comienza con el valor 651).

	type	name	birthdate	deathdate
1	Main	root	22-11-2022 00:00:00	
2	TipoCoche	<population>[0] : 651	22-11-2022 00:00:00	22-11-2022 00:04:20
3	TipoCoche	<population>[1] : 652	22-11-2022 00:00:35	22-11-2022 00:03:34
4	TipoCoche	<population>[2] : 653	22-11-2022 00:01:10	22-11-2022 00:04:05
5	TipoCoche	<population>[3] : 654	22-11-2022 00:01:45	22-11-2022 00:04:44
6	TipoCoche	<population>[4] : 655	22-11-2022 00:02:20	22-11-2022 00:05:55
7	TipoCoche	<population>[5] : 656	22-11-2022 00:02:55	22-11-2022 00:06:35
8	TipoCoche	<population>[5] : 658	22-11-2022 00:04:05	22-11-2022 00:07:04

Tabla 14: registro 'agent_log' Anylogic

- Event_log: cada vez que un evento se ejecuta a lo largo de la simulación su información se registra en esta tabla. Los nombres de los eventos se pueden ver en la columna 'event_name'. En la columna 'date' representa la fecha dentro del tiempo de simulación en la que se ejecuta.

	event_name	date	agent_type	agent
1	StopEngineEspecifica	22-11-2022 00:13:35	Main	root

Tabla 15: registro 'Event_log' Anylogic

- Flowchart_process_states_log: cada vez que un agente atraviesa un bloque Wait y Delay en este modelo, se registra en esta sección. La columna 'agent_type' especifica el tipo de agente, 'agent' indica con el número de su parte derecha, a qué agente se refiere (comenzando desde el número 651). 'block_type' identifica el tipo de bloque, mientras que 'block' proporciona el nombre específico del bloque. La columna 'activity_type' indica la actividad que realiza el bloque, para los bloques Delay siempre estará en estado de trabajo y para los bloques Wait siempre en estado de espera. Por último, las columnas 'start_date' y 'stop_date' indican el tiempo de inicio y finalización de la acción del bloque dentro del tiempo de la simulación, respectivamente. Estas últimas son particularmente interesantes al filtrar los bloques Wait y al observar las estaciones donde los agentes pasan más tiempo esperando.

	agent_type	agent	block_type	block	activity_type	start_date	stop_date
1	TipoCoche	<population>[0] : 651	Delay	Est1C1	WORK	22-11-2022 00:00:00	22-11-2022 00:00:32
2	TipoCoche	<population>[0] : 651	Delay	Est3C1	WORK	22-11-2022 00:00:32	22-11-2022 00:01:02
3	TipoCoche	<population>[1] : 652	Delay	Est1C2	WORK	22-11-2022 00:00:35	22-11-2022 00:01:00
4	TipoCoche	<population>[1] : 652	Delay	Est2C2	WORK	22-11-2022 00:01:00	22-11-2022 00:01:21
5	TipoCoche	<population>[0] : 651	Wait	waitEst2	WAIT	22-11-2022 00:01:02	22-11-2022 00:01:21
6	TipoCoche	<population>[2] : 653	Delay	Est1C3	WORK	22-11-2022 00:01:10	22-11-2022 00:01:39
7	TipoCoche	<population>[1] : 652	Delay	Est4C2	WORK	22-11-2022 00:01:21	22-11-2022 00:01:36
8	TipoCoche	<population>[0] : 651	Delay	Est2C1	WORK	22-11-2022 00:01:21	22-11-2022 00:01:49
9	TipoCoche	<population>[1] : 652	Delay	Est3C2	WORK	22-11-2022 00:01:36	22-11-2022 00:02:04
10	TipoCoche	<population>[2] : 653	Wait	waitEst3	WAIT	22-11-2022 00:01:39	22-11-2022 00:02:04

Tabla 16: registro ‘Flowchart_process_states _log’ Anylogic

- Flowchart_stats_time_in_state_log: Para este modelo en particular, se registran varios parámetros relacionados con los tiempos de los bloques Wait y Delay. La columna 'block_type' indica cuál de estos dos tipos se trata, mientras que 'block' especifica el bloque específico del modelo.

  La columna 'activity_time' indica la actividad que están realizando estos bloques, siendo siempre 'WAIT' para los bloques Wait y 'WORK' para los bloques Delay.

  Para determinar la cantidad de agentes que pasan por cada bloque, se consulta la columna 'n_agent'.

  El resto de las columnas, están relacionadas con parámetros temporales específicos de cada

bloque en segundos. Para los bloques Delay, los tiempos mínimo, máximo y medio siempre serán los mismos, ya que provienen de la hoja Excel de datos de entrada del coche (columnas 'min_second', 'max_seconds' y 'mean_seconds', respectivamente). El tiempo total que esos bloques han realizado su actividad se puede observar en la columna ‘total_seconds’. La información más relevante que ofrece este registro se obtiene al filtrar o examinar los bloques Wait, permitiendo analizar las estaciones que más tiempo esperan, identificar posibles cuellos de botella y determinar aquellas más propensas a la saturación, al igual que si se observan los bloques Delay que más tiempo pasan trabajando.

	block_type	block	activity_type	mean_seconds	total_seconds	min_seconds	max_seconds	n_agents
1	Delay	Est10C1	WORK	26	156	26	26	6
2	Delay	Est10C2	WORK	5	30	5	5	6
3	Delay	Est10C3	WORK	13	78	13	13	6
4	Delay	Est1C1	WORK	32	192	32	32	6
5	Delay	Est1C2	WORK	25	150	25	25	6
6	Delay	Est1C3	WORK	29	174	29	29	6
7	Delay	Est2C1	WORK	28	168	28	28	6
8	Delay	Est2C2	WORK	21	126	21	21	6
9	Delay	Est2C3	WORK	24	144	24	24	6
10	Delay	Est3C1	WORK	30	180	30	30	6
11	Delay	Est3C2	WORK	28	168	28	28	6
12	Delay	Est3C3	WORK	11	66	11	11	6
13	Delay	Est4C1	WORK	21	126	21	21	6
14	Delay	Est4C2	WORK	15	90	15	15	6

Tabla 17: registro ‘Flowchart_stats_time_in_state _log’ Anylogic

- Statistics_log: se recopilan datos estadísticos de los bloques Delay, TimeMeasureEnd y Wait, incluyendo su tipo de bloque y el nombre específico del bloque, así como sus valores medios, mínimos, máximos, desviación estándar, confianza promedio y el número de agentes que atraviesan el bloque.

  La información relacionada con los bloques TimeMeasureEnd resulta especialmente interesante, al observar el bloque 'TiempoTotal', donde se puede analizar el tiempo promedio que tarda un coche en producirse, así como identificar los coches que requieren menos y más tiempo.

	agent_type	agent	name	mean	deviation	minimum	maximum	mean_confidence	number
1	TimeMeasureEnd	TiempoTotal	distribution	212.794	41.078	144.386	285.994	15.314	30
2	TimeMeasureEnd	OFFTiempoEst9C3	distribution	14.818	3.401	12	23	2.257	11
3	TimeMeasureEnd	OFFTiempoEst9C2	distribution	21.625	5.041	19	32	4.11	8
4	TimeMeasureEnd	OFFTiempoEst9C1	distribution	9.455	4.503	8	23	2.988	11
5	TimeMeasureEnd	OFFTiempoEst8C3	distribution	10.385	6.966	3	22	4.623	11
6	TimeMeasureEnd	OFFTiempoEst8C2	distribution	15	0	15	15	0	8
7	TimeMeasureEnd	OFFTiempoEst8C1	distribution	20.517	1.713	20	25.683	1.137	11
8	TimeMeasureEnd	OFFTiempoEst7C3	distribution	4.636	3.325	3	14	2.206	11
9	TimeMeasureEnd	OFFTiempoEst7C2	distribution	8.75	5.339	6	20	4.352	8
10	TimeMeasureEnd	OFFTiempoEst7C1	distribution	15	0	15	15	0	11
11	TimeMeasureEnd	OFFTiempoEst6C3	distribution	11.727	1.618	11	15	1.074	11
12	TimeMeasureEnd	OFFTiempoEst6C2	distribution	8	0	8	8	0	8
13	TimeMeasureEnd	OFFTiempoEst6C1	distribution	6.636	2.111	6	13	1.401	11
14	TimeMeasureEnd	OFFTiempoEst5C3	distribution	34.785	23.353	12	70	15.498	11

Tabla 18: registro ‘Statistics _log’ Anylogic

- Library_block_parameters_log: muestra los valores iniciales de los parámetros de todos los bloques del modelo. Sus columnas representan el tipo de bloque, El nombre de ese bloque, el nombre del parámetro y el valor del parámetro.

	block_type	block	parameter_name	parameter_value
1	Delay	Est10C1	forceStatisticsCollection	false
2	Delay	Est10C1	type	TIMEOUT
3	Delay	Est10C1	capacity	1
4	Delay	Est10C1	maximumCapacity	false
5	Delay	Est10C1	entityLocation	null
6	Delay	Est10C1	pushProtocol	false
7	Delay	Est10C1	restoreEntityLocationOnExit	true
8	Delay	Est10C2	type	TIMEOUT
9	Delay	Est10C2	capacity	1
10	Delay	Est10C2	maximumCapacity	false
11	Delay	Est10C2	entityLocation	null
12	Delay	Est10C2	pushProtocol	false

Tabla 19: registro ‘Library_block_parameters_log ‘Anylogic

4.13.2 Generados mediante código

Para poder exportarlo al Excel, se ha creado desde cero una tabla en la base de datos llama ‘res_idletime_simu’, en la que se recogen datos relacionados con el tiempo ocioso de cada estación y tipo de coche.

Está compuesta por 15 columnas:

- Estación: indica la estación en la que se registra ese tiempo ocioso.
- Id_coche: permite identificar que coche (mostrando su número de identificación único) entra en la estación e interrumpe este tiempo ocioso.
- Tipo_coche: permite diferenciar entre los tres tipos de coche.
- Total_idle_time : relacionada con los indicadores $IdleTimeEst_i$, muestra el tiempo ocioso que ha transcurrido en la estación hasta que ha entrado el siguiente coche.
- Tiempo_simu: muestra el tiempo de simulación en el que se ha introducido esa fila con sus respectivos datos.
- Idle_total_Est_i : son un total de 10 columnas, muestran por cada estación el valor acumulado de su tiempo ocioso. Están relacionadas con los indicadores $IdleTimeTotalEst_i$.

estacion	id coche	tipo coche	total idle time	tiempo simu	idle total est1	idle total est2	idle total est3	idle total est4	idle total est5	idle total est6	idle total est7	idle total est8	idle total est9	idle total est10
Est3	1	Coche1	32	32			32							
Est1	2	Coche2	3	35	3									
Est2	2	Coche2	60	60		60								
Est1	3	Coche3	10	70	13									
Est2	1	Coche1	0	81		60								
Est4	2	Coche2	81	81				81						
Est3	2	Coche2	34	96			66							
Est1	4	Coche3	6	105	19									
Est4	1	Coche1	13	109				94						
Est5	2	Coche2	124	124					124					
Est2	3	Coche3	26	135		86								
Est1	5	Coche2	6	140	25									
Est10	2	Coche2	147	147										147
Est9	2	Coche2	152	152									152	
Est2	4	Coche3	0	159		86								
Est8	2	Coche2	171	171								171		
Est1	6	Coche1	10	175	35									
Est6	1	Coche1	179	179						179				

Tabla 20: resultados excel Idle time (‘res_idletime_simu’)

Esta tabla, al no estar generada automáticamente, requiere la eliminación de los resultados obtenidos en la anterior simulación. Para que esto ocurra, se ha incorporado el siguiente código en las acciones de java de la configuración de la simulación (Ventana de proyecto → Simulation→ Java Action en la ventana de propiedades → Before each experiment run):

```
deleteFrom(res_idletime_simu)
  .execute();//Borra los datos de las tablas de la base de datos de res_idletime_simu
```

De esta manera, cada vez que se inicia una simulación, este código se ejecutará y la tabla ‘res_idletime_simu’ solo almacena los valores correspondientes a esa simulación.

4.13.3 Documentación adicional

Si llegase a ser necesaria aún más información del modelo, se puede generar una documentación con todos los parámetros, bloques utilizados, acciones, etc. Para generarlo hay que seleccionar la opción Tools en la zona superior izquierda y hacer clic en 'Create Documentation', por último, seleccionar el destino y formato de salida del documento.

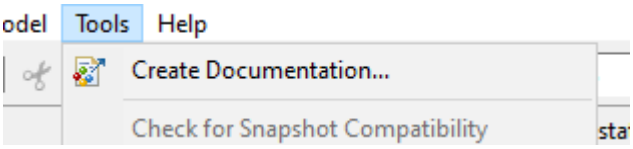

Figura 86: crear documentación adicional Anylogic

5 EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se desarrollará un ejemplo práctico del empleo de la herramienta desarrollada para comprobar la viabilidad de una propuesta de diseño de planta. Para ello se seguirán los pasos descritos anteriormente apuntando todos los parámetros en el Excel 'DatosEntradaCoche', realizando la simulación en Anylogic y viendo los resultados obtenidos tanto en Anylogic como en el Excel de datos de salida.

5.1 Parámetros entrada

Coche 1	Coche 2	Coche 3
VERDADERO	VERDADERO	VERDADERO

COMPROBACIÓN PRECEDENCIAS
CORRECTO

Figura 87: resultados comprobación precedencias ejemplo práctico

	Orden estaciones		
	Coche 1	Coche 2	Coche 3
Paso 1	1	1	1
Paso 2	3	2	3
Paso 3	2	4	2
Paso 4	4	3	5
Paso 5	5	5	4
Paso 6	-1	-1	-1
Paso 7			
Paso 8			
Paso 9			
Paso 10			
FINAL			

Tabla 21: orden estaciones ejemplo práctico

TIEMPO ESTACIÓN	
COCHE 1	
1	35
2	25
3	29
4	22
5	38
6	0
7	0
8	0
9	0
10	0

TIEMPO ESTACIÓN	
COCHE 2	
1	30
2	19
3	27
4	10
5	31
6	0
7	0
8	0
9	0
10	0

TIEMPO ESTACIÓN	
COCHE 3	
1	22
2	31
3	11
4	10
5	16
6	0
7	0
8	0
9	0
10	0

Tabla 22: tiempos por estaciones y tipo de coches del ejemplo práctico

5.2 Simulación

Se producen un total de 30 coches, se ha introducido como parámetro que el número de secuencias repetidas sea diez y que se produzcan los coches tipo 1,2,3 en el vector tipo coche demanda.

Vector Tipo coche demanda
1
2
3

Tabla 23: vector tipo coche demanda del ejemplo práctico

5.2.1 Demanda máxima

Para este caso, el takt time es de 40 segundos y la capacidad de las estaciones es de 1. Al iniciar la simulación, en el segundo 149, se recibe un aviso del software indicando que el agente en el bloque 'Enter' de la estación 5 no puede abandonarlo. Esto se debe a que la capacidad es 1 y, con ese takt time, la estación se colapsa, lo que hace inviable este diseño de layout con estos tiempos.

Para resolver este problema, lo más sencillo sería aumentar el takt time, pero esto incumpliría el requisito establecido en las instrucciones de entrega del trabajo de clase. Además, las gráficas de saturación de buffer muestran una alta congestión en relación con el poco tiempo que lleva la simulación.

Número de estaciones	5
Selecciona tipo de Demanda	Especifica
Takt Time	40
Numero Secuencias Repetidas	10
Capacidad de las estaciones	1

Tabla 24: parámetros demanda máxima ejemplo práctico

Figura 88: modelo simulado demanda máxima ejemplo práctico

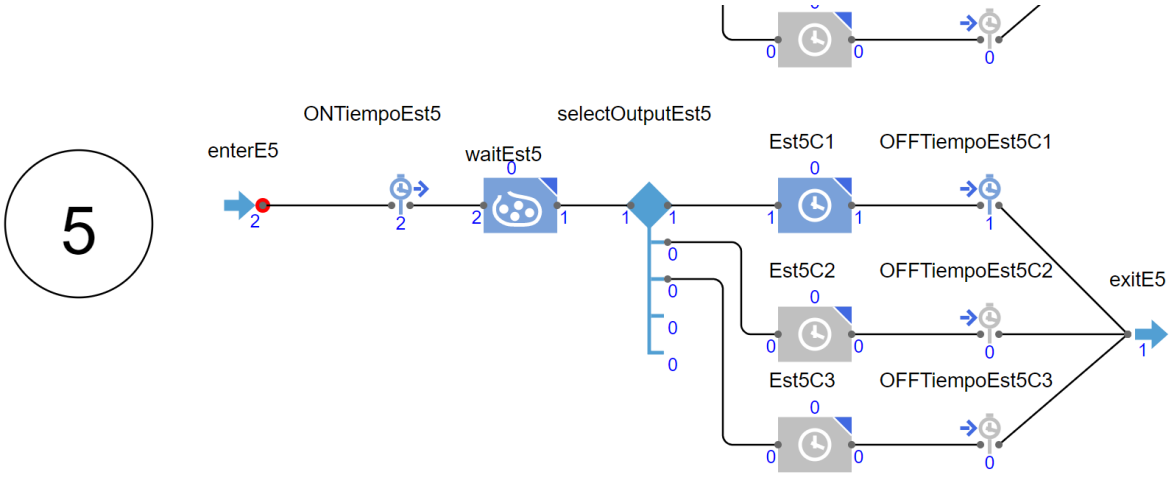

Figura 89: fallo estación caso demanda máxima ejemplo práctico

En la Figura 89, se puede observar cómo Anylogic indica con un círculo rojo que el agente no puede abandonar el bloque 'enterE5'.

5.2.2 Demanda mínima

Para este caso, el takt time es de 160 segundos y la capacidad de las estaciones es de 1. En esta situación, la simulación se puede realizar sin que el software muestre ningún error. Con este valor tan alto de takt time, se observa en las gráficas que ninguna estación tiene los buffers saturados a lo largo de la simulación. Los idle time de todas las estaciones son valores bastante altos y se destaca que la estación 4 es la que menos carga de trabajo tiene. En las gráficas de ocupación de estaciones por tipo de coche se puede observar que los tres tipos pasan por las cinco estaciones, aunque cada uno sigue un orden de paso por las estaciones diferente.

Número de estaciones	5
Selecciona tipo de Demanda	Especifica
Takt Time	160
Numero Secuencias Repetidas	10
Capacidad de las estaciones	1

Tabla 25: parámetros demanda mínima ejemplo práctico

Figura 90: resultado modelo simulado demanda mínima ejemplo práctico

5.3 Resultados simulación en 'DatosSalidaAny'

Para el caso de demanda mínima, ya que se ha podido realizar la simulación correctamente si se han podido exportar los resultados obtenidos. Algunos de los más relevantes son:

- **Agent_log:** Este registro muestra el inicio y finalización de la producción de cada coche. Es importante destacar que, debido al alto valor del takt time, nunca se produce más de un coche simultáneamente durante la simulación.

type	name	birthdate	deathdate
Main	root	0:00:00	
TipoCoche	<population>[0] : 651	0:00:00	0:02:29
TipoCoche	<population>[0] : 652	0:02:40	0:04:37
TipoCoche	<population>[0] : 653	0:05:20	0:06:50
TipoCoche	<population>[0] : 654	0:08:00	0:10:29
TipoCoche	<population>[0] : 655	0:10:40	0:12:37
TipoCoche	<population>[0] : 656	0:13:20	0:14:50
TipoCoche	<population>[0] : 657	0:16:00	0:18:29
TipoCoche	<population>[0] : 658	0:18:40	0:20:37
TipoCoche	<population>[0] : 659	0:21:20	0:22:50
TipoCoche	<population>[0] : 660	0:24:00	0:26:29
TipoCoche	<population>[0] : 661	0:26:40	0:28:37
TipoCoche	<population>[0] : 662	0:29:20	0:30:50
TipoCoche	<population>[0] : 663	0:32:00	0:34:29
TipoCoche	<population>[0] : 664	0:34:40	0:36:37
TipoCoche	<population>[0] : 665	0:37:20	0:38:50
TipoCoche	<population>[0] : 666	0:40:00	0:42:29
TipoCoche	<population>[0] : 667	0:42:40	0:44:37
TipoCoche	<population>[0] : 668	0:45:20	0:46:50
TipoCoche	<population>[0] : 669	0:48:00	0:50:29
TipoCoche	<population>[0] : 670	0:50:40	0:52:37
TipoCoche	<population>[0] : 671	0:53:20	0:54:50
TipoCoche	<population>[0] : 672	0:56:00	0:58:29
TipoCoche	<population>[0] : 673	0:58:40	1:00:37
TipoCoche	<population>[0] : 674	1:01:20	1:02:50
TipoCoche	<population>[0] : 675	1:04:00	1:06:29
TipoCoche	<population>[0] : 676	1:06:40	1:08:37
TipoCoche	<population>[0] : 677	1:09:20	1:10:50
TipoCoche	<population>[0] : 678	1:12:00	1:14:29
TipoCoche	<population>[0] : 679	1:14:40	1:16:37
TipoCoche	<population>[0] : 680	1:17:20	1:18:50

Tabla 26: resultados agent_log demanda mínima del ejemplo práctico

- **flowchart_stats_time_in_state_:** Estos datos proporcionan información sobre los agentes que pasan por cada uno de los bloques de espera en cada estación, así como su tiempo total en segundos. Es relevante mencionar que los tiempos máximos, mínimo y medio corresponden a los valores introducidos en la hoja de datos de entrada.

block type	block	activity typ	mean secc	total secor	min secon	max secon	n agents
Delay	Est1C1	WORK	35	350	35	35	10
Delay	Est1C2	WORK	30	300	30	30	10
Delay	Est1C3	WORK	22	220	22	22	10
Delay	Est2C1	WORK	25	250	25	25	10
Delay	Est2C2	WORK	19	190	19	19	10
Delay	Est2C3	WORK	31	310	31	31	10
Delay	Est3C1	WORK	29	290	29	29	10
Delay	Est3C2	WORK	27	270	27	27	10
Delay	Est3C3	WORK	11	110	11	11	10
Delay	Est4C1	WORK	22	220	22	22	10
Delay	Est4C2	WORK	10	100	10	10	10
Delay	Est4C3	WORK	10	100	10	10	10
Delay	Est5C1	WORK	38	380	38	38	10
Delay	Est5C2	WORK	31	310	31	31	10
Delay	Est5C3	WORK	16	160	16	16	10

Tabla 27: resultados flowchart_stats_time_in_state demanda mínima del ejemplo práctico

5.4 Corrección caso demanda máxima

Gracias a este modelo de simulación, se puede comprobar que este layout con un takt time de 40 no es viable. Sin embargo, al ajustar fácilmente los parámetros en el Excel de datos de entrada aumentando tres segundos el takt time, al volver a realizar la simulación se puede comprobar que ya sí es factible.

Se obtienen los siguientes resultados:

Figura 91: resultado modelo simulado demanda máxima parámetros corregidos del ejemplo práctico

El tiempo en el que se producen estos 30 coches es de 1367.11, siendo muy inferior al tiempo del caso de demanda mínima. Como era de esperar tras los resultados obtenidos con el takt time de 40 segundos, la estación con los buffers más saturados es la cinco; las estaciones dos y cuatro también presentan una

alta saturación en sus respectivos buffers. Por otro lado, la que presenta mayor tiempo ocioso es la estación cuatro, seguida de la tres y la dos.

Consultando el Excel de datos de salida:

- **Flowchart_stats_time_in_state :** Los valores más destacables son los obtenidos con los bloques wait, siendo el más alto el de la estación cinco con un total de 310 segundos.

block type	block	activity typ	mean secc	total secor	min secon	max secon	n agents
Delay	Est1C1	WORK	35	350	35	35	10
Delay	Est1C2	WORK	30	300	30	30	10
Delay	Est1C3	WORK	22	220	22	22	10
Delay	Est2C1	WORK	25	250	25	25	10
Delay	Est2C2	WORK	19	190	19	19	10
Delay	Est2C3	WORK	31	310	31	31	10
Delay	Est3C1	WORK	29	290	29	29	10
Delay	Est3C2	WORK	27	270	27	27	10
Delay	Est3C3	WORK	11	110	11	11	10
Delay	Est4C1	WORK	22	220	22	22	10
Delay	Est4C2	WORK	10	100	10	10	10
Delay	Est4C3	WORK	10	100	10	10	10
Delay	Est5C1	WORK	38	380	38	38	10
Delay	Est5C2	WORK	31	310	31	31	10
Delay	Est5C3	WORK	16	160	16	16	10
Wait	waitEst2	WAIT	16	160	16	16	10
Wait	waitEst4	WAIT	3	30	3	3	10
Wait	waitEst5	WAIT	15,5	310	1	30	20

Tabla 28:resultados flowchart_stats_time_in_state demanda máxima parametros corregidos del ejemplo práctico

- **Agent log:** Ya no ocurre como en el caso de demanda mínima; con este takt time, sí se producen bastantes coches simultáneamente, terminando el primero de ellos en 2 minutos con 29 segundos.

type	name	birthdate	deathdate
Main	root	0:00:00	
TipoCoche	<population>[0] : 651	0:00:00	0:02:29
TipoCoche	<population>[1] : 652	0:00:43	0:03:00
TipoCoche	<population>[2] : 653	0:01:26	0:03:26
TipoCoche	<population>[2] : 656	0:03:35	0:05:35
TipoCoche	<population>[2] : 659	0:05:44	0:07:44
TipoCoche	<population>[2] : 662	0:07:53	0:09:53
TipoCoche	<population>[2] : 665	0:10:02	0:12:02
TipoCoche	<population>[2] : 668	0:12:11	0:14:11
TipoCoche	<population>[2] : 671	0:14:20	0:16:20
TipoCoche	<population>[2] : 674	0:16:29	0:18:29
TipoCoche	<population>[2] : 677	0:18:38	0:20:38
TipoCoche	<population>[2] : 680	0:20:47	0:22:47
TipoCoche	<population>[3] : 654	0:02:09	0:04:38
TipoCoche	<population>[3] : 655	0:02:52	0:05:09
TipoCoche	<population>[3] : 657	0:04:18	0:06:47
TipoCoche	<population>[3] : 658	0:05:01	0:07:18
TipoCoche	<population>[3] : 660	0:06:27	0:08:56
TipoCoche	<population>[3] : 661	0:07:10	0:09:27
TipoCoche	<population>[3] : 663	0:08:36	0:11:05
TipoCoche	<population>[3] : 664	0:09:19	0:11:36
TipoCoche	<population>[3] : 666	0:10:45	0:13:14
TipoCoche	<population>[3] : 667	0:11:28	0:13:45
TipoCoche	<population>[3] : 669	0:12:54	0:15:23
TipoCoche	<population>[3] : 670	0:13:37	0:15:54
TipoCoche	<population>[3] : 672	0:15:03	0:17:32
TipoCoche	<population>[3] : 673	0:15:46	0:18:03
TipoCoche	<population>[3] : 675	0:17:12	0:19:41
TipoCoche	<population>[3] : 676	0:17:55	0:20:12
TipoCoche	<population>[3] : 678	0:19:21	0:21:50
TipoCoche	<population>[3] : 679	0:20:04	0:22:21

Tabla 29: resultados agent_log demanda máxima parámetros corregidos

6 CONCLUSIONES

En este estudio, se ha abordado la aplicabilidad y las ventajas de la simulación, centrándose especialmente en el contexto de una fábrica multiproducto. Se ha resaltado la versatilidad de la simulación en diferentes áreas, desde la planificación de la planta, su layout, hasta la toma de decisiones en la producción, especialmente enfatizando la implementación de la simulación basada en agentes. A través de la manipulación de parámetros como la capacidad de las estaciones, el takt time y el tiempo de producción o transporte del producto, entre otros, se logran modelar escenarios dinámicos que proporcionan información relevante.

Por otro lado, también se destaca el impacto de las células de fabricación en la mejora continua de la productividad en los sistemas de fabricación. Agrupando máquinas, trabajadores y herramientas para producir diferentes tipos de productos y mejorar la eficiencia operativa del sistema productivo.

Este modelo de simulación basada en agentes desarrollado en el software Anylogic, donde los agentes representan coches en un proceso de ensamblaje, permite recopilar datos destacables de bloques como Delay, TimeMeasureEnd y Wait, proporcionando una comprensión profunda del rendimiento del sistema. La obtención de datos sobre el tiempo ocioso de cada estación, ratios de ocupación y la saturación de los buffers de las estaciones permite realizar ajustes en los parámetros de entrada, buscando constantemente mejorar la eficiencia global de la planta. Por otro lado, la interfaz de simulación creada permite ver con facilidad y sin necesidad de comprender profundamente cómo funciona esta fábrica, pudiendo obtener todo tipo de datos en tiempo real mientras que se realiza la simulación.

Con el ejemplo aplicado en el último capítulo, se puede apreciar la utilidad del modelo creado. En caso de que se produzcan errores y que el diseño no sea factible, gracias a la interfaz gráfica creada se puede detectar fácilmente su causa. Para solucionar estos errores se pueden cambiar fácilmente los parámetros gracias al Excel creado en el que se agrupan todos los datos necesarios en una sola hoja. Una vez que se han realizado dichos cambios y la simulación ha podido ser llevada a cabo, se pueden observar ciertos resultados dentro de Anylogic, si se prefiere se pueden observar estos mismos junto a otros generados mediante código en el Excel de salida.

En concreto para el caso práctico del último capítulo, se ha podido confirmar la inviabilidad inicial de

un diseño con un tiempo de ciclo determinado, así como demostrar su factibilidad al aumentar solo tres segundos el takt time en el Excel de datos de entrada. Durante este proceso, se ha analizado detalladamente la capacidad de las estaciones y los tiempos muertos. Además, se ha obtenido información relevante del tiempo total necesario para llevar a cabo toda la producción, los tiempos en los bloques de espera y el registro de producción de coches entre otros.

En resumen, la simulación basada en agentes emerge como una muy potente herramienta, no solo para estudiar escenarios y anticipar resultados en fábricas multiproducto, sino también en una gran variedad de sectores como transporte, logística, salud y estrategias de negocio, entre otros. La capacidad de visualizar y analizar dinámicamente los procesos proporciona perspectivas fundamentales que pueden ser aprovechadas para optimizar la eficiencia y la productividad en entornos complejos.

BIBLIOGRAFÍA

Afonso C. Medina, GENOA, & The AnyLogic Company. (2023, febrero 23). *AnyLogic tutorial*. https://www.anylogic.com/resources/books/tutorial-para-anylogic-en-espanol/

Alhammad, M. M., & Moreno, A. M. (2018). Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software*, *141*, 131-150. https://doi.org/10.1016/j.jss.2018.03.065

Askin, R. G. (2013). Contributions to the design and analysis of cellular manufacturing systems. *International Journal of Production Research*, *51*(23-24), 6778-6787. https://doi.org/10.1080/00207543.2013.825745

Bandini, S. and Manzoni, S. and V. G. (2012). Agent Based Modeling and Simulation. En R. A. Meyers (Ed.), *Computational Complexity: Theory, Techniques, and Applications* (pp. 105-117). Springer New York. https://doi.org/10.1007/978-1-4614-1800-9_7

Banks, J. (1999). Introduction to simulation. *Winter Simulation Conference Proceedings*, *1*, 7-13. https://doi.org/10.1145/324138.324142

Chtourou, H., Jerbi, A., & Maalej, A. Y. (2008). The cellular manufacturing paradox: A critical review of simulation studies. *Journal of Manufacturing Technology Management*, *19*(5), 591-606. https://doi.org/10.1108/17410380810877276

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining «gamification». *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011*. https://doi.org/10.1145/2181037.2181040

Egne, J (2011, febrero 9). *What is the difference between STELLA and iThink?* https://blog.iseesystems.com/stella-ithink/what-is-the-difference-between-stella-and-ithink/. https://blog.iseesystems.com/stella-ithink/what-is-the-difference-between-stella-and-ithink/

Escudero-Santana, A., Muñoz-Díaz, M. & Lorenzo-Espejo, A. (2022). *Innovación docente en la asignatura Sistemas Integrados de Producción* En: *Innovación Docente en Ingeniería de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Periodo 2019-2021* . Instituto de Ciencias de la Educación de la Universidad de Sevilla.

García-Valdecasas Medina, J. I. (2011). La simulación basada en agentes: una nueva forma de explorar los fenómenos sociales. *Revista Española de Investigaciones Sociológicas*, *136*. https://doi.org/10.5477/cis/reis.136.91

Grigoryev, I. (2018). AnyLogic 8 in Three Days. *World Health*, *fifth edition*.

isee systems. (2023). *iThink*. https://www.iseesystems.com/store/products/ithink.aspx

Johnson, L., Adams Becker, S., Estrada, V., & Freeman, A. (2014). Horizon Report. En *NMC Horizon Report*.

Kapp, K. M. (2012). The Gamification of Learning and Instruction: Game-based Methods and ... - Karl M. Kapp. En *The Gamification of Learning and Instruction*.

Lorenzo-Espejo, A. (2018). *Dinámica de sistemas para la simulación de técnicas híbridas de control de la producción*.

Macal, C. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, *10*, 144-156. https://doi.org/10.1057/jos.2016.7

Mahdavi, A. (2020). *The Art of Process-Centric Modeling*. https://www.anylogic.com/resources/books/the-art-of-process-centric-modeling-with-anylogic/

Pegden, C. D., Shannon, R. E., & Sadowski, R. P. (1990). Introduction to Simulation Using SIMAN, McGraw-Hill. *New York, USA*, *2*.

Shannon, R. E. (1998). Introduction to the art and science of simulation. *Winter Simulation Conference Proceedings*, *1*, 7-14. https://doi.org/10.1109/wsc.1998.744892

Sharma, P. (2015). Discrete-Event Simulation. *International Journal of Scientific & Technology Research*, *4*(04).

Siebers, P. O., MacAl, C. M., Garnett, J., Buxton, D., & Pidd, M. (2010). Discrete-event simulation is dead, long live agent-based simulation! En *Journal of Simulation* (Vol. 4, Número 3, pp. 204-210). https://doi.org/10.1057/jos.2010.14

Slota, A., & Malopolski, W. (2007). Integration of simulation software arena with FMS control system. *International Journal of Simulation Modelling*, *6*(3), 165-172. https://doi.org/10.2507/IJSIMM06(3)3.088

S. Carson, J. (2005). Introduction to modeling and simulation. *Proceedings of the Winter Simulation*

Conference, 2005., 8 pp.-. https://doi.org/10.1109/WSC.2005.1574235

Stadnicka, D., & Deif, A. (2019). A gamification approach application to facilitate lean manufacturing knowledge acquisition. *Management and Production Engineering Review*, *10*(4), 108-122. https://doi.org/10.24425/mper.2019.131451

The Anylogic Company (2023a). *Anylogic Library Reference Guides*. https://anylogic.help/library-reference-guides/index.html. https://anylogic.help/library-reference-guides/index.html

The AnyLogic Company (2023b). *Anylogic: Variables and parameters*. https://anylogic.help/anylogic/data/parameters.html

The AnyLogic Company (2023c). *Anylogic: visualizing data using charts*. https://anylogic.help/anylogic/analysis/pie-chart.html

The LEGO Group. (2023). *Instrucciones de construcción por 31027, Deportivo Azul, LEGO® Creator.* https://www.lego.com/es-es/service/buildinginstructions/31027?locale=es-es

Younes Sinaki, R., Sadeghi, A., Mosadegh, H., Almasarwah, N., & Suer, G. (2022). Cellular manufacturing design 1996–2021: a review and introduction to applications of Industry 4.0. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2022.2105763

ANEXO 1: Notación

BLOQUES

SourceEspecífico	Bloque Source que genera los agentes en modo demanda específica
OnTiempoDemEsp	Temporizador de inicio tras la creación del agente para demanda específica
exigen	Bloque que permite la salida de los agentes de la zona del comienzo de la producción
SourceEspecífico	Bloque Source que genera los agentes en modo demanda aleatoria
ONTiempoDemEsp	Temporizador de inicio tras la creación del agente para demanda específica
enterE_i	Bloque que permite la entrada de los agentes a la estación de trabajo i, $i \in 1..10$
ONTiempoEst_i	Temporizador de inicio para las estaciones de trabajo i, $i \in 1..10$
waitEst_i	Bloque Wait para dimensionar capacidad de cola en las estaciones de trabajo i, $i \in 1..10$
selectOutputEst_i	Bloque selectOutput5 que envía según el tipo de coche j dentro de la estación de trabajo i, a que bloque delay debe salir el agente $i \in 1..10$; $j \in 1..3$
Est_iC_j	Bloque delay, tiempo de fabricación del coche j en la estación i, $i \in 1..10$; $j \in 1..3$
$\text{OFFTiempoEst}_i\text{C}_j$	Temporizador final para las estaciones de trabajo i, $i \in 1..10$; $j \in 1..3$
exitE_i	Bloque exit que permite la salida de los agentes de la estación de trabajo i, $i \in 1..10$
enterFinal	Bloque enter que permite la entrada de los agentes a la zona de producto terminado.
TiempoTotal	Temporizador final que marca el tiempo total tardado durante toda la producción.
SinkFinal	Bloque sink que marca el final de la producción