Definición de un framework para la realización de Modelos de Diseño para Simulación basada en Eventos Discretos

Victoria Elizabeth Meichtry Regner[1]

[1] Universidad Tecnológica Nacional, Facultad Regional Santa Fe (UTN-FRSF), Santa Fe, Argentina
meichtryv4@gmail.com

Resumen. Este trabajo tiene como finalidad la construcción de una herramienta que dé soporte a la definición de modelos para la simulación basada en eventos discretos. El objetivo es especificar reglas que permitan la definición sintáctica de un modelo a partir de la abstracción de una situación del mundo real y el desarrollo de un metamodelo que explique dicha abstracción semánticamente. Para esto, se establece una gramática que permite reconocer expresiones válidas y, a partir de ello, se permita la visualización de un conjunto de relaciones entre los conceptos del modelo a través de una instancia del metamodelo. Para el desarrollo de cada etapa se utilizan como soporte, primeramente, Xtext, plugin de Eclipse para definir reglas sintácticas con el fin de reconocer sentencias válidas. Además, se utilizó Eclipse Modeling Framework (EMF) para realizar un metamodelo base con el propósito de definir conceptos y establecer relaciones entre ellos, describiendo así la semántica. Por último, a partir de Atlas Transformation Language (ATL) se realizó una conversión desde la sintaxis a partir del metamodelo autogenerado por la gramática hacia la semántica a partir del metamodelo de dominio generado.
Palabras clave: modelo, gramática, metamodelo, simulación.

1 Introducción

La simulación es una herramienta poderosa que puede utilizarse como complemento en el proceso de diseño de un proyecto de software [1], como así también para estudiar múltiples problemas relacionados con un sistema real del cual se haya obtenido previamente un modelo [2]. Un modelo es una representación de un objeto, sistema, o idea, cuyo propósito es ayudar a explicarlo, entenderlo o mejorarlo [3]. Precisamente en la tarea de modelado se requiere de la habilidad para analizar un problema, lo que consiste en resumir las características esenciales de éste, seleccionar y modificar las suposiciones básicas que caracterizan al sistema, y luego enriquecer y elaborar el modelo hasta obtener una aproximación útil.

Según Robinson, el proceso de modelado en el área de simulación consiste en distintas etapas [3]. La Fig. 1 esquematiza estas etapas y sus relaciones. En primer lugar, se establece una situación del mundo real de la cual se requiere un modelo. Luego, a

través de la adquisición de conocimiento acerca del problema o situación, se obtiene la descripción del sistema. En dicha descripción, se explica el problema y todos los elementos del mundo real que se relacionan con éste. En tercer lugar y a partir de una abstracción, se desarrolla el modelo conceptual el cual es una representación no basada en software del modelo de simulación computacional que como mínimo incluye objetivos, entradas y salidas, contenido del modelo, suposiciones y simplificaciones que hayamos hecho. A partir de esto, se realiza un modelo de diseño en donde, a raíz del tipo de simulación que se elige para desarrollar, se realiza un diseño de los elementos y la lógica que se va a utilizar para el modelo computacional. En última instancia, se especifica el modelo computacional, el cual es la traducción de los modelos previos en una herramienta de software de simulación especifica. Luego, la construcción de un modelo de simulación conlleva el desarrollo de tres modelos: modelo conceptual, modelo de diseño y modelo computacional.

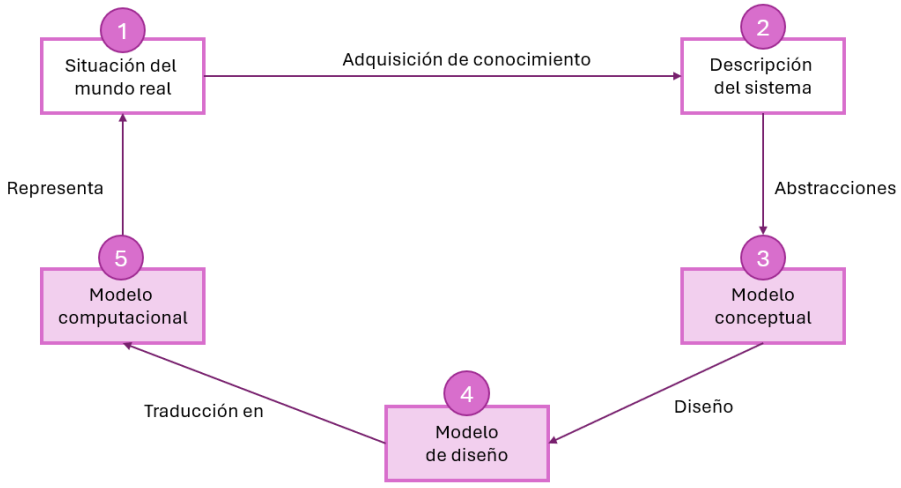

Fig. 1. Etapas del proceso de modelado dentro de un proyecto de simulación. Este proceso es iterativo, por lo que se repite tantas veces como sea necesario hasta obtener una buena aproximación del modelo que se está buscando según el objetivo de estudio y los conocimientos previos.

Este trabajo se centra en las actividades para la obtención de las etapas 4 y 5 del proceso de modelado, proponiendo el desarrollo de una herramienta que facilitará la definición de modelos computacionales a partir de modelos de diseño para simulaciones basadas en eventos discretos. Para esto, se propone una gramática como forma de representación textual que permita definir sintácticamente un modelo de diseño de simulación asociado a eventos discretos. Además, se pretende verificar la relación entre los distintos conceptos sintácticos a partir de un metamodelo semántico del dominio. Se incluye un conjunto de reglas que permiten vincular la gramática y el metamodelo para poder transformar cada expresión sintácticamente correcta en una relación entre elementos del modelo semánticamente valida.

En función de esto, el objetivo final es, a partir de la definición de la sintaxis, reconocer expresiones válidas con el fin de estructurar los principales conceptos que componen un modelo de diseño. El resultado de dicha validación (es decir, las sentencias correctamente identificadas por la sintaxis) se vincula a una instancia de metamodelo (definida a partir del metamodelo de dominio) a fin de definir las bases necesarias para dar una estructura semánticamente correcta del modelo de diseño propuesto. La definición de cada etapa de esta herramienta se realiza a través de Eclipse [4]. Eclipse es una plataforma de software que engloba un conjunto de herramientas de programación de código abierto y multiplataforma, diseñadas para facilitar el desarrollo de aplicaciones. Dentro de las herramientas de soporte de dicha plataforma, aquellas que fueron utilizadas en el desarrollo del trabajo corresponden a Xtext [5], Atlas Transformation Language (ATL) [6] y Eclipse Modeling Framework (EMF) [7].

2 Modelo para simulación basada en eventos discretos

El fin principal de realizar un modelo de simulación es ejecutarlo para analizar su comportamiento, teniendo los parámetros y restricciones necesarias. Se realiza un modelo ya que resulta más económico y sencillo que realizar el sistema para simularlo. Luego, si el modelo se comporta de acuerdo con lo esperado, pueden obtenerse conclusiones del sistema del cual se obtuvo el modelo.

Dentro de la simulación de eventos discretos [8], un modelo de diseño consiste en la definición de varios elementos que se consideran importantes tales como:

- *Entidades*: Son aquellos objetos de interés en el sistema para un determinado fin.
- *Atributos*: Son características de las entidades.
- *Estado*: Describe los valores que obtienen los distintos atributos de la entidad.
- *Evento*: Es el cambio instantáneo del estado.
- *Actividad*: Es la operación que produce las transformaciones en los estados del sistema.
- *Funciones*: Son las que establecen las relaciones entre los atributos.

Estos elementos, tal como se expresa son parte del modelo de diseño (etapa 4 del proceso de modelado dentro de la Fig. 1) por lo tanto es necesario identificarlos a partir del modelo conceptual (etapa previa) donde ya se logra definir gran parte del modelo de simulación que se busca representar.

A partir de esto, los elementos descriptos se consideran relevantes puesto que son los que utilizará el usuario para definir los modelos de simulación que realice, tanto sintáctica como semánticamente.

3 Arquitectura de la herramienta propuesta

Con el objetivo de facilitar la definición de modelos de diseño basados en los conceptos descriptos en la Sección 2, se diseñó una herramienta basada en descripciones textuales. Los principales módulos de esta herramienta son:

- Módulo “Sintaxis”: necesario para la representación textual del modelo de diseño.
- Módulo “Semántica”: útil para la representar las relaciones del modelo.
- Módulo “Transformación Sintaxis-Semántica”: utilizado para realizar la conversión del modelo textual a una instancia del metamodelo.

Tal como se ha dicho con anterioridad, para el desarrollo de la herramienta se utilizó Eclipse como tecnología base. En principio, se usó Xtext [4], plugin de Eclipse para el desarrollo de la gramática. Luego, Eclipse Modeling Framework [6] se usó como soporte para el desarrollo del metamodelo. Por último, se empleó Atlas Transformation Language [5] para realizar la transformación de la gramática en una instancia del metamodelo.

La Fig. 2 muestra la forma en la cual estas herramientas dan soporte a cada componente de la herramienta bajo desarrollo.

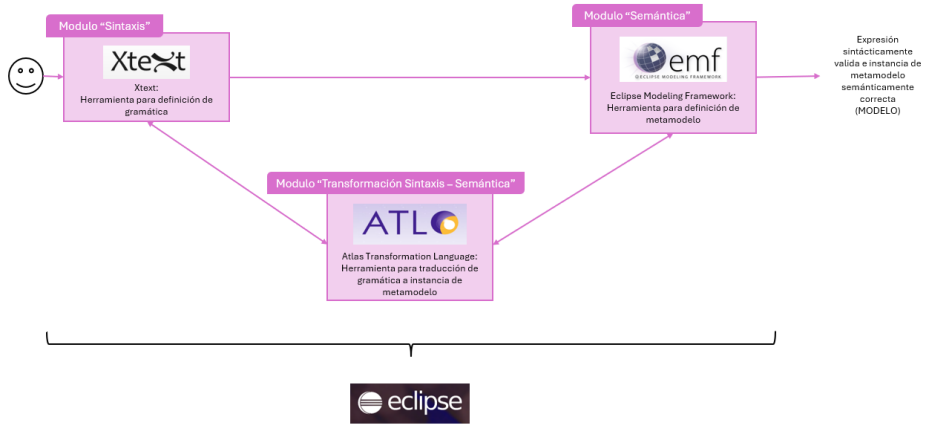

Fig. 2. Módulos y herramientas que dan soporte al proyecto.

3.1 Módulo “Sintaxis”

En primer lugar, se utilizó Xtext para la definición de la gramática. Xtext es un framework de código abierto que se utiliza para desarrollar lenguajes específicos de dominio (DSL) y sus respectivos editores en Eclipse. Este marco permite definir la sintaxis, semántica y reglas de validación de un lenguaje de manera sencilla. Xtext utiliza una gramática libre de contexto (GLC) internamente para definir la sintaxis de los lenguajes específicos de dominio (DSL) que se crean con él.

A través de reglas se definió la gramática utilizada para validar expresiones que definan el modelo que se quiere representar. A modo de ejemplo, la Fig. 3 desarrolla la regla para especificar los atributos de una entidad.

```
AttributesDefinition:
    'Attributes' '(' entity = Entity ')' '=' '{' attributesEst += AttributesEst+ '};'
;
```

Fig. 3. Regla sintáctica para definir los atributos de una entidad.

Como se puede ver en la Fig. 3, la regla consta de un encabezado en el cual se define el nombre con el que va a ser invocada la regla dentro de otra regla. Luego, se decidió establecer que para definir atributos es necesario escribir la palabra literal *"Attributes"* seguida por la entidad de la cual se definen los atributos entre paréntesis. Después, se debe tipear el signo *"="* con los atributos de la entidad separados por coma y entre llaves. Luego, si el usuario define una entidad con sus atributos de la manera en que fue establecido, la expresión será correcta sintácticamente.

Al momento de pensar cómo definir las reglas y qué tipo de expresiones se considerarían como válidas, se buscó establecer que éstas fueran cercanas a un lenguaje de programación, basadas en estructuras legibles y claras. La Tabla 1 presenta la definición de las expresiones que se consideran como válidas dentro de la gramática.

Tabla 1. Definición de expresiones sintácticamente válidas.

Conceptos	**Definición**
Atributos	Attributes (**Entity**) = {**attribute1, attribute2, …, attributeN**};
Actividades	Activities (**Entity**) = {**activity1, activity2, …, activityN**};
	Activities (**Entity**) = {**activity1 [startEvent1 : finishEvent1], activity2 [startEvent2 : finishEvent2], …, activityN [startEventN : finishEventN]**};
	Activities (**Entity**) = {**[startEvent1 : finishEvent1] activity1, [startEvent2 : finishEvent2] activity2, …, [startEventN : finishEventN] activityN**};
Eventos **** Existe la posibilidad de intercalar ambas definiciones.***	Events (**Entity**) = {**event1** (**eventType1**) = **eventDefinition1** (**entity1**), **event2** (**eventType2**) = **eventDefinition2** (**entity2**), …, **eventN** (**eventTypeN**) = **eventDefinitionN** (**entityN**)};
	Events (**Entity**) = {**event1** (**eventType**), **event2** (**eventType2**), …, **eventN** (**eventTypeN**)};
Estados	For **Entity** states = {**state1** (**attributeValue1, AttributeValue2, …, attributeValueN**), **state2** (**attributeValue1, AttributeValue2, …, attributeValueN**), …, **stateN** (**attributeValue1, AttributeValue2, …, attributeValueN**)};
	For **Entity** states = {(**attributeValue1, AttributeValue2, …, attributeValueN**), (**attributeValue1, AttributeValue2, …, attributeValueN**), …, (**attributeValue1, AttributeValue2, …, attributeValueN**)};
	For **Entity** states = {**state1** (**attribute1** = **attributeValue1, attribute2** = **AttributeValue2, …, attributeN** = **attributeValueN**), **state2** (**attribute1** = **attributeValue1, attribute2** = **AttributeValue2, …, attributeN** = **attributeValueN**), …, **stateN** (**attribute1** = **attributeValue1, attribute2 = AttributeValue2, …, attributeN = attributeValueN**)};
	For **Entity** states = {(**attribute1** = **attributeValue1, attribute2** = **AttributeValue2, …, attributeN** = **attributeValueN**), (**attribute1** = **attributeValue1, attribute2** = **AttributeValue2, …, attributeN** = **attributeValueN**), …, (**attribute1** = **attributeValue1, attribute2 = AttributeValue2, …, attributeN = attributeValueN**)};

En la Tabla 1, se puede observar que aquello que está definido en negrita es lo que debe ser completado por el usuario según lo que desee modelar. Luego, aquello definido sin formato debe ser escrito de forma literal al momento de especificar un modelo.

3.2 Módulo “Semántica”

Eclipse Modeling Framework (EMF) es un conjunto de herramientas y tecnologías desarrolladas por Eclipse Foundation para facilitar la creación, manipulación y gestión de modelos de datos estructurados. Está diseñado para ayudar a construir aplicaciones basadas en modelos, lo que significa que las entidades del sistema (como objetos, relaciones y atributos) se representan como modelos en lugar de codificarse directamente en el código fuente.

Haciendo uso de EMF se especificó el metamodelo para definir la semántica de dominio, es decir, la manera de entender las relaciones entre cada uno de los conceptos del modelo en estudio (los vistos en la Sección 2). La Fig. 4 presenta una vista parcial de dicho metamodelo, ya que solo se definieron algunos elementos relevantes del modelo de diseño.

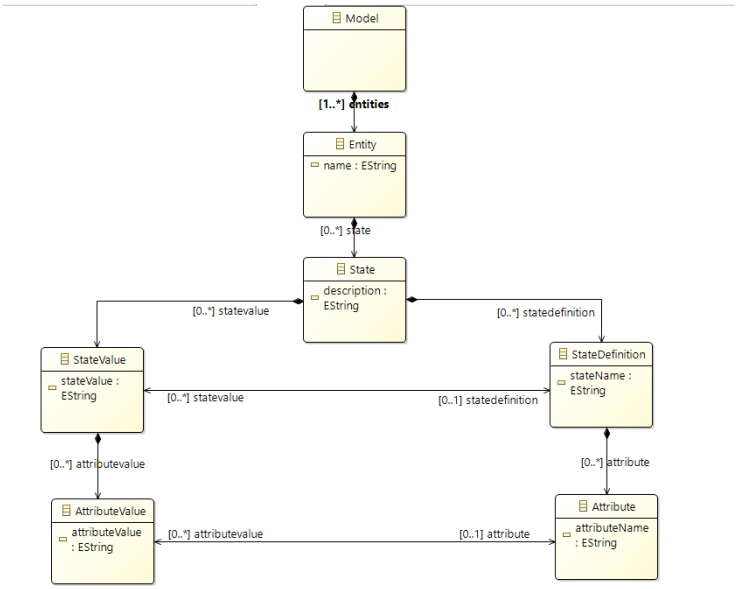

Fig. 4. Metamodelo utilizado para representar el dominio de simulación basada en eventos discretos.

Como se puede visualizar en Fig. 4, un modelo (Model) está compuesto de al menos una entidad. Además, cada entidad que se modela (Entity) está compuesta de un conjunto de estados (State) del cual forma parte una definición de estado (StateDefinition) que hace referencia a un conjunto de valores de estado (StateValue). Por otra parte, la

definición de estado está compuesta por un conjunto de atributos (Attribute) los cuales hacen referencia a un conjunto de valores de atributo (AttributeValue).

3.3 Módulo "Transformación Sintaxis-Semántica"

Atlas Transformation Language (ATL) es un lenguaje de transformación desarrollado por Eclipse Foundation. Su principal objetivo es permitir la definición de transformaciones entre diferentes modelos, lo que facilita la conversión y el mapeo de datos entre diferentes representaciones de información. Se utiliza principalmente en el contexto de Ingeniería de Modelos para realizar tareas de transformación entre modelos definidos en lenguajes específicos de dominio o lenguajes de modelado estándar como UML (Unified Modeling Language) o Ecore (Eclipse Modeling Framework).

En este contexto y, sobre ambas definiciones (gramática y metamodelo), a partir de reglas de transformación, se mapea la entrada como expresiones sintácticamente correctas con una instancia del metamodelo, logrando mostrar cómo se relaciona semánticamente lo definido en la expresión que se definió como entrada. Haciendo uso de un metamodelo, es más sencillo comprender las asociaciones entre entidades del modelo definido en la representación textual.

A modo de ejemplo, en la Fig. 5 se observa la regla que permite pasar el concepto "*Attribute*" desde la representación sintáctica al concepto "*Attribute*" de la representación semántica. Como se puede observar, para definir la regla es necesario de un nombre que permita identificarla (en este caso, Attribute2Attribute). Luego, tomando como base el modelo de sintaxis y semántica (elementos definidos como from and to), se establece la forma en la cual se realiza la transformación.

```
rule Attribute2Attribute {
    from
        s: Stx!Attribute
    to
        t: Smt!Attribute
        (attributeName <- s.attribute)

}
```

Fig. 5. Regla de transformación de *"Attribute"*.

4 Resultados

En esta sección se describe un ejemplo sencillo de aplicación basado en el estudio de una línea de despacho de equipaje en aeropuerto. El mismo corresponde al modelo conceptual que utilizan los alumnos de 4to año de la carrera Ingeniería en Sistemas de Información al iniciar el cursado de la catedra "Simulación". La descripción de dicho modelo conceptual se encuentra disponible en [9].

La evolución del modelo conceptual al modelo de diseño se propone en clases de forma conjunta con los estudiantes, generando un modelo de diseño basado en tres entidades: Cola, Pasajero y Mostrador. En la actualidad, dicho modelo es documentado en una planilla de Excel. A modo de ejemplo, utilizaremos únicamente la entidad "Mostrador" (Tabla 2).

La herramienta propuesta permite definir el fragmento de la Tabla 2 según la Fig. 7. Esta especificación es válida sintácticamente puesto que en la Fig. 7, no se observan errores, ni advertencias. Luego, tomando como base el modelo de la gramática, al aplicar las reglas de transformación propuestas en la Sección 3.3, se genera el modelo de conceptos de dominio a partir de los elementos del metamodelo EMF propuesto. La Fig. 8 presenta los resultados que se obtienen para el ejemplo de entrada.

Tabla 2. Especificación de la entidad *"Mostrador"* presentada a los alumnos.

Entidad	Mostrador	
Atributos	Tiempo total ocupado	
	Tiempo total libre	
	Tiempo de fin última atención	
	Duración de atención	
	Empleado	
Actividades	Atender	Comienza la atención
		Finaliza la atención
	Esperar	Finaliza la atención
		Comienza la atención
Eventos	Comienza la atención = Arriba mostrador (pasajero)	Condicional
	Finaliza la atención = Abandona mostrador (pasajero)	Incondicional

```
Attributes ('mostrador') = {
    'tiempoInicioEspera',
    'tiempoFinEspera',
    'tiempoInicioAtencion',
    'tiempoFinAtencion',
    'tiempoArribo'
};

/*Activities ('mostrador') = {
    'atender',
    'esperar'
};
Activities ('mostrador') = {
    'atender' ['comienzaAtencion' : 'finalizaAtencion'],
    'esperar' ['finalizaAtencion' : 'comienzaAtencion']
};*/

Activities ('mostrador') = {
    ['comienzaAtencion' : 'finalizaAtencion'] 'atender',
    ['finalizaAtencion' : 'comienzaAtencion'] 'esperar'
};
Events ('mostrador') = {
    'comienzaAtencion' (conditional) = 'arribaMostrador' ('Pasajero'),
    'finalizaAtencion' (unconditional) = 'abandonaMostrador' ('Pasajero')
};

/*Events ('mostrador') = {
    'comienzaAtencion' (conditional),
    'finalizaAtencion' (unconditional)
};*/
```

Fig. 7. Ejecución de la gramática para la entidad *"Mostrador"*.

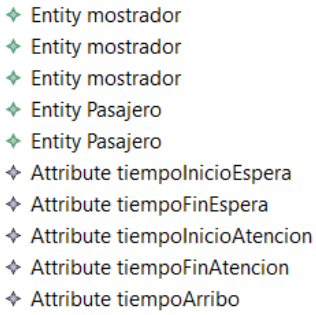

Fig. 8. Resultado de transformación de sintaxis a semántica de la entidad *"Mostrador"*.

Es importante destacar que las sentencias comentadas en la Fig. 7 corresponden a alternativas de definición del mismo elemento, siendo también expresiones sintácticamente correctas en el lenguaje bajo desarrollo.

En función de lo observado en Fig. 7 y Fig. 8 se obtiene que esta herramienta permite especificar textualmente un modelo que representa un sistema o problema de la vida real. A partir de las reglas de transformación, se observa una conversión desde la representación textual a una instancia del metamodelo de dominio.

Se puede observar que todavía no se logra representar un modelo completamente. Esto se debe a que se está completando el desarrollo de reglas según nuevos elementos se incorporan al diseño del metamodelo de dominio. Pero, se espera que, en comparación al método utilizado hasta el momento (es decir, una planilla de Excel), la herramienta propuesta enriquezca el conocimiento tanto respecto a la representación de conceptos como así también a su especificación y visualización.

5 Conclusiones

Para poder simular un sistema real es imprescindible disponer previamente de un modelo. Para lograrlo, es necesario seguir todas las etapas del modelado e iterar hasta conseguir una aproximación útil del sistema en estudio. En este proyecto nos enfocamos en la etapa de modelo de diseño que, como se explicó anteriormente, se logra a través de una buena abstracción de la descripción del sistema.

Hasta el momento se ha logrado especificar las reglas sintácticas para la definición de la gramática que se utiliza para detallar el modelo de diseño y el diseño del metamodelo para relacionar la definición textual con una representación gráfica bien conocida, que logre explicar semánticamente la relación entre los conceptos del modelo. Se ha presentado un ejemplo sencillo a fin de mostrar la aplicabilidad de la herramienta a un caso real trabajado por los estudiantes. Se busca que la herramienta ayude a comprender visualmente el modelo generado.

En un futuro, se pretende evolucionar la herramienta para que, a partir del modelo de dominio que se obtiene del modelo de diseño, realice un mapeo a componentes de herramientas de simulación comercial (como, por ejemplo, SIMIO [10] o ARENA [11]).

Referencias

1. María Julia Blas; Silvio Gonnet.: Using Model-to-Model Transformations for Web Software Architecture Simulation. pp. 545 – 552. IEEE LATIN AMERICA TRANSACTIONS. (2022)
2. Gonzalo Alvarez; María Julia Blas.: An Hybrid Simulation-Optimization Solution for Improving the Safety of Power Grids under Blackout Condition. The 2021 International Conference on Decision Aid Sciences and Applications. (2021)
3. Robinson.: Simulation: The Practice of Model Development and Use. United States. (2014)
4. Eclipse Foundation. https://www.eclipse.org/. Último acceso: 23/03/2024
5. Xtext. https://eclipse.dev/Xtext/. Último acceso: 23/03/2024
6. Eclipse Foundation. https://eclipse.dev/atl/. Último acceso: 23/03/2024
7. Eclipse Foundation. https://eclipse.dev/modeling/emf/. Último acceso: 23/03/2024
8. Jerry Banks, John S. Carson II, Barry Nelson.: Discret-Event System Simulation. Estados Unidos: Ed. Prentice-Hall. (2021).
9. Gonzalo Alvarez; Juan Leonardo Sarli; María Julia Blas.: Una Propuesta de Extensión de un Framework para el Desarrollo de Modelos Conceptuales para Simulación. 11º Congreso Nacional de Ingeniería Informática / Sistemas de Información. San Miguel de Tucumán; (2023).
10. SIMIO. https://www.simio-simulacion.es/. Último acceso: 05/04/2024
11. ARENA. https://info.arenasimulation.com/v16-esd-0. Último acceso: 05/04/2024