Especificación de GLCs para dar soporte al modelado y simulación mediante SES y DEVS

José Ignacio Flores, Victoria Elizabeth Meichtry Regner,
Santiago Andrés Mercanti, Francisco Fernando Pallotti

Universidad Tecnológica Nacional, Facultad Regional Santa Fe (UTN-FRSF)

{josefloresicr, meichtryv4,santi.mercanti15,
franpallotti}@gmail.com

Resumen. El presente trabajo propone el diseño e implementación de tres gramáticas libres de contexto (GLC) como herramientas de soporte para modelado y simulación (M&S) basado en eventos discretos. En primer lugar, se describen los aspectos generales y desafíos del M&S de sistemas reales basados en eventos y la ontología System Entity Structure (SES) para la construcción de modelos que describen las relaciones jerárquicas entre las entidades del sistema. A continuación, se provee una descripción de los componentes de un modelo conceptual de alto nivel, que incluyen entidades, atributos, estados, eventos, actividades y funciones, los cuales pueden ser especificados usando el formalismo Discrete Event System Specification (DEVS). Finalmente, se propone una GLC que contribuye a la definición formal de modelos acoplados DEVS. Las tres gramáticas fueron implementadas usando la herramienta ANTLR4. Se presenta también un ejemplo de modelado común a los tres niveles, a fin demostrar el funcionamiento de cada gramática. A futuro, estas GLC serán utilizadas como base para el desarrollo de una herramienta de software que facilite el mapeo de los distintos niveles de modelados necesarios durante el desarrollo de modelos de simulación.

Palabras clave: Gramática libre de contexto, SES, DEVS, ANTLR4.

1 Introducción

El modelado y simulación de sistemas reales basados en eventos es una tarea compleja [1]. Normalmente, dicha tarea parte de la especificación de un aspecto de un sistema dinámico de interés (fenómeno bajo estudio), el cual es trasladado a un modelo de alto nivel (es decir, una abstracción o modelo del fenómeno). Luego, este modelo de primer nivel puede ser especificado como un modelo de simulación formal haciendo uso de un formalismo en particular. Este proceso de modelado y definición de un sistema (esquematizado en la Fig. 1) puede observarse como un modelo unificado en capas.

System Entity Structure (SES) es un framework ontológico para la construcción de modelos que describen relaciones jerárquicas entre entidades de un sistema. Fue introducido para la representación del conocimiento sobre la descomposición, taxonomía y

acoplamiento de sistemas. Tiene aplicaciones en modelado, simulación, diseño de sistemas e ingeniería de datos [2]. Sobre la base de una definición SES, y dentro de la simulación de eventos discretos, es posible identificar múltiples elementos que se consideran importantes para la especificación de un modelo conceptual de alto nivel [3], tales como: *i) entidades* que son aquellos objetos de interés en el sistema para un determinado fin, *ii) atributos* que son características de las entidades, *iii) estado* que describe los *valores* que tienen los distintos atributos de la entidad en un momento dado, *iv) eventos* que son los que producen el cambio instantáneo del estado, *v) actividades* que indican las operaciones que producen las transformaciones en los estados del sistema, y *vi) funciones* que establecen las relaciones entre los atributos. A su vez, sobre la base de un modelo general, es posible definir modelos de simulación de eventos discretos formales haciendo uso del formalismo Discrete Event System Specification (DEVS) [4]. DEVS es el formalismo más general para la representación de sistemas de eventos discretos, orientado a problemas de modelado y simulación. Se dice que es el formalismo para eventos discretos universal, ya que absorbe a otros más populares como las Redes de Petri, las Statecharts, Grafcet y Grafos de Eventos.

La definición de un modelo puede comprender entonces (como se observa en la Fig.1) la definición del entorno y los conceptos a modelar mediante la utilización del framework SES, procediendo luego a la caracterización mediante una abstracción de alto nivel y concluyendo con una formalización a través de la especificación de un modelo formal DEVS.

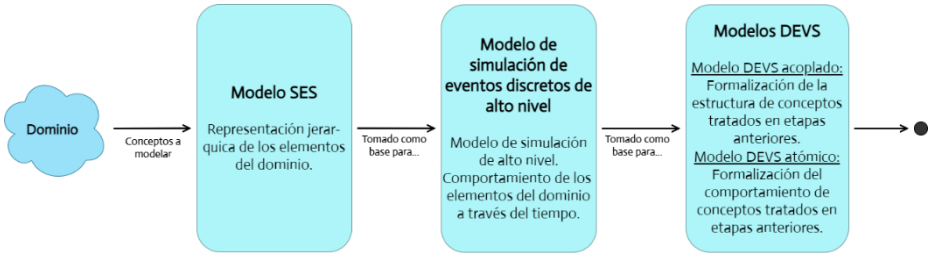


Fig. 1. Cómo la ontología SES y el formalismo DEVS colaboran para el modelado de un dominio.

En este contexto, en este trabajo se propone el diseño de tres gramáticas libres de contexto (GLC)[1] como soporte al M&S basado en eventos discretos. Las gramáticas que se presentan han sido implementadas haciendo uso de ANTLR (ANother Tool for Language Recognition) [5]. En la Sección 2 se trabaja sobre especificaciones de modelos System Entity Structure como sentencias particulares del idioma inglés que permiten describir diferentes componentes de dichos modelos. En la Sección 3 se presenta una GLC para la especificación de modelos de simulación de alto nivel basados en eventos. Finalmente, en la Sección 4 se desarrolla una GLC para la especificación formal de modelos acoplados DEVS. En conjunto, las gramáticas se usarán como soporte

[1] Una gramática libre de contexto es una estructura capaz de definir un lenguaje (libre de contexto) mediante reglas que describen como pueden estar formadas las sentencias del lenguaje [6].

en el desarrollo de una herramienta de software integral que facilite el mapeo de los distintos niveles de modelado requeridos en el desarrollo de modelos de simulación.

2 Definición de Modelos System Entity Structure

El bloque fundamental de un modelo SES [2] son las entidades, que representan tanto objetos físicos como conceptuales. Estas entidades pueden organizarse mediante tres tipos de relaciones: *i)* de aspecto, que representa un criterio para la descomposición de una entidad en diferentes partes, *ii)* de multiaspecto, que representa un criterio para descomponer una entidad en múltiples partes de un mismo tipo (y podría considerarse un caso particular de aspecto), y *iii)* de especialización, que representa el vínculo entre una entidad y una variante particular de la misma, perteneciente a una categoría dada.

Asimismo, otro aspecto constitutivo de SES es la posibilidad de asociar variables a una entidad. Estas variables representan propiedades asociadas a la entidad mediante un identificador. También es posible definirles un rango o universo de valores.

En la Tabla 1 se presentan las restricciones sobre la sintaxis de las sentencias que deben ser reconocidas como validas en un modelo SES.

Tabla 1. Restricciones impuestas sobre una sentencia para que describa parte de un modelo SES (adaptado de [2]). Estas sentencias son reconocidas por la GLC propuesta ignorando artículos como “a”, “an” y “the”.

Concepto	Restricción
Especialización	**entidad** can be **variante**$_0$, …, **variante**$_{n-1}$ or **variante**$_n$ in **criterio_de_especialización** *Ejemplo: “A book can be red, blue or green in color”*
Aspecto	From **criterio_de_descomposición** perspective, **entidad** is made of **componente**$_0$, …, **componente**$_{n-1}$ and **componente**$_n$ *Ejemplo: “From a physical perspective, a book is made of a front cover, a back cover and a core”*
Multiaspecto	From **criterio_de_descomposición** perspective, **entidad** is/are made of more than one **entidad** *Ejemplo: “From a physical perspective, the core is made of more than one page”*
Variables	**entidad** has **variable**$_0$, …, **variable**$_{n-1}$ and **variable**$_n$ *Ejemplo: “A page has a page number and an amount of lines”*
Rango	The range of **entidad**’s variable is **rango** *Ejemplo: “The range of a page’s page number is integer”*

En la Fig. 2 se muestra un subconjunto de las reglas que forman parte de la definición de la GLC. En particular, se visualizan las reglas usadas para reconocer sentencias que describan una relación de aspecto entre entidades. Mediante la herramienta ANTLR4, junto a la gramática especificada, es posible procesar textos compuestos por una o más sentencias que definan de manera parcial (o total) un modelo SES. El parser generado con ANTLR4 toma como entrada un texto y da como salida un árbol de derivación, descomponiendo la entrada en sus diferentes partes sintácticas (en base a la gramática

que fue indicada). Cada una de dichas partes define un elemento del modelo SES que la entrada describe.

Las reglas que se observan en la Fig. 2 formalizan las restricciones establecidas en la Tabla 1, específicamente las restricciones para una sentencia que describe una relación de aspecto. La regla *asp* establece que la oración debe estar compuesta por la palabra "from" seguido de un identificador *asp_type* que indica la perspectiva de descomposición, la palabra "perspective" seguida por una coma, un identificador entity para la entidad que se está descomponiendo, un verbo to-be ("is" o "are"), y un conjunto de componentes que, como las dos reglas subsiguientes indican, consiste de una sola entidad o varias entidades separadas debidamente por comas y el conector "and".

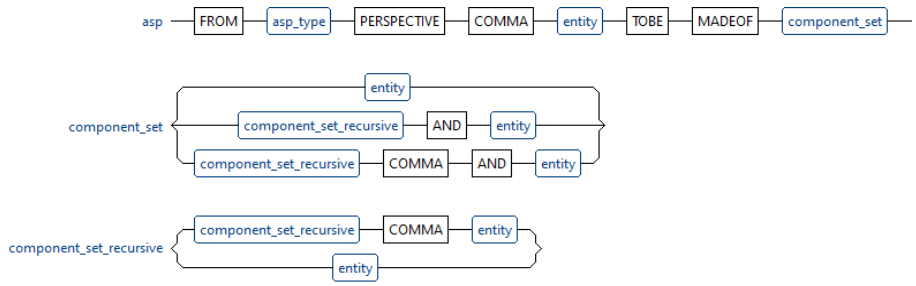


Fig. 2. Reglas gramaticales correspondientes a una sentencia que describe una relación de aspecto. Los identificadores *asp_type* y *entity* admiten letras, numeros y espacios.

Una vez que se cuenta con una especificación textual de un modelo SES (correcta), debe verificarse su estructura. Por ejemplo, no tiene sentido definir una variable para una entidad que no existe en el modelo. Por ello, surge la necesidad de definir un modelo conceptual de los elementos de un modelo SES y las relaciones entre ellos. Este modelo se presenta en la Fig. 3.

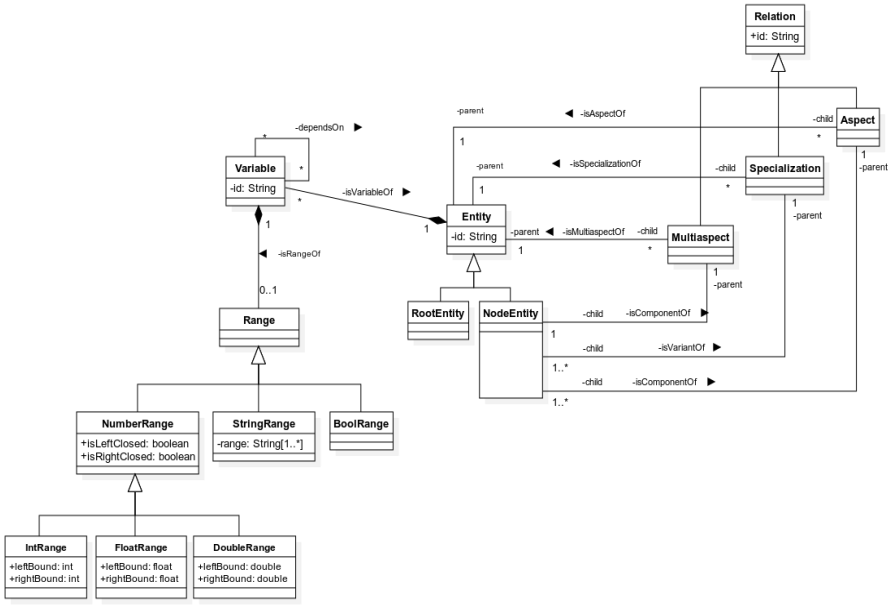
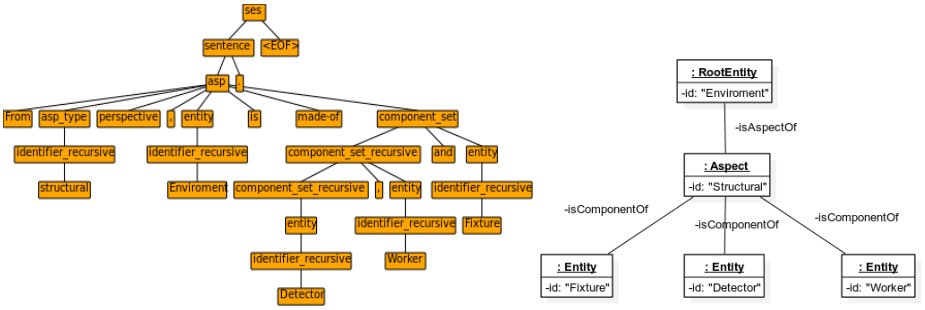


Fig. 3. Diagrama de clases UML que modela la estructura de un modelo SES.

En la Fig. 4 se muestra una instanciación de este modelo conceptual, usando un ejemplo sintácticamente correcto de especificación de un modelo SES. El ejemplo se corresponde con la siguiente descripción:

"From a structural perspective, an Enviroment is made of a Detector, a Worker and a Fixture."



(a) (b)

Fig. 4. (a) Arbol sintáctico de la sentencia producido por ANTLR4. (b) Instanciación del modelo conceptual.

3 Definición de Alto Nivel de un Modelo de Simulación basado en Eventos Discretos

Hasta aquí, hemos descrito la implementación de una GLC para la representación del conocimiento sobre la descomposición de sistemas. A nivel modelo de simulación de eventos discretos y con el fin de aceptar sentencias correctas que describan los elementos que se representan en un modelo de alto nivel, se desarrolló otra GLC utilizando ANTLR. Al igual que en el caso previo, a partir de reglas léxicas se generó un analizador léxico que permite reconocer tokens de un archivo de entrada. Luego, se desarrolló un analizador sintáctico o parser el cual construye una estructura de datos llamada árbol sintáctico que registra como el analizador reconoció la estructura de la palabra de entrada (secuencia de tokens).

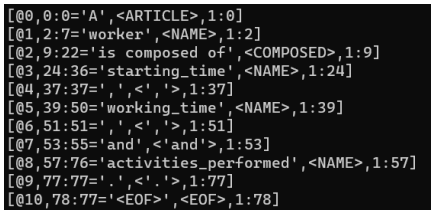
Un árbol sintáctico, se utiliza para representar el resultado de ejecución de una gramática con el fin de obtener de manera más visible la relación entre las reglas con las sentencias de prueba.

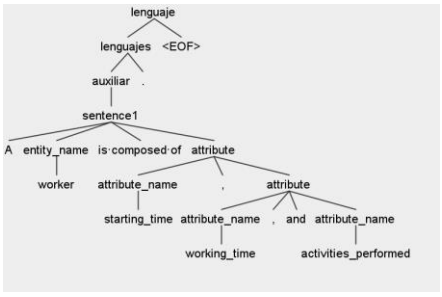
En la gramática, solo se definieron algunos elementos de simulación, ya que resulta más sencillo trabajar con un conjunto más pequeño de elementos para lograr observar y corregir los distintos errores que pueden ir surgiendo. Luego, el objetivo es añadir los elementos restantes a la gramática.

A continuación, se desarrolla un ejemplo a partir del "Worker" (incluido como entidad en la sección previa), con el fin de entender el modo de ejecución y el funcionamiento de la gramática implementada. Se utiliza como ejemplo de prueba la siguiente sentencia:

"A worker is composed of starting_time, working_time, and activities_performed."

Esta es una sentencia válida. Al ejecutar el análisis de la gramática, se obtiene por cada token una línea del tipo `@numeroDeToken, PosicionInicio:PosicionFin = 'SecuenciaDeCaracteresIdentificados', <Token> Linea:posicionInicioRelativaaLaLinea.` Se puede observar el resultado de dicha ejecución en la Fig. 5(a). El árbol sintáctico generado se presenta en la Fig. 5(b).





(a) (b)

Fig. 5. (a) Tokens generados para la sentencia de ejemplo. (b) Árbol generado para la sentencia de ejemplo.

En el árbol resultante (Fig. 5(b)), se puede observar en la raíz del árbol (en la parte superior de la imagen) el símbolo de inicio, en los nodos internos (parte intermedia de la imagen) los distintos no-terminales utilizados en la generación de las reglas de la gramática y en los nodos hoja (parte inferior de la imagen) se logra visualizar los terminales, la sentencia de prueba planteada al comienzo. Por lo tanto, la gramática permitió reconocer que la sentencia era válida a partir de la derivación, es decir, de aplicar las reglas de la gramática.

En último lugar, en este ejemplo se tiene solo una sentencia. Si se tuviese una cantidad mayor de oraciones, el tiempo de revisión manual sería grande por lo que, en esos casos, la herramienta reduce el tiempo de revisión ya que solo es necesario ingresar un conjunto de comandos ANTLR para establecer la validez de las sentencias.

4 Definición Formal de un Modelo de Simulación basado en Eventos como un Modelo DEVS Acoplado

Ya definida una representación de descomposición de sistema y un modelo de simulación de alto nivel, abordamos el problema de definir modelos formales DEVS. En primer lugar, es necesario comprender la definición formal de un modelo acoplado DEVS, como sigue:

$$N = (X, Y, C, D, EIC, EOC, IC)$$

donde:

$X = \{(p,v) / p \in InputPorts, v \in X_p\}$ es el conjunto de puertos y valores de entrada.

$Y = \{(p,v) / p \in OutputPorts, v \in Y_p\}$ es el conjunto de puertos y valores de salida.

$C = \{Component1, Component2, \ldots\}$ es el conjunto de los nombres de los componentes.

$D = \{Comp1 \text{ is } ExampleModel1, Comp2 \text{ is } ExampleModel2, \ldots\}$ es la definición de los componentes establecidos en C.

$EIC = \{(N;iPort1), (CompX; iPort1)), \ldots\}$ es el conjunto de entradas externas que conectan componentes externos con componentes internos al modelo.

$EOC = \{((CompX;oPort1), (N; oPort1)), \ldots\}$ es el conjunto de salidas externas que conectan componentes internos con componentes externos al modelo.

$IC = \{((CompA;port1),(CompB;port1)), \ldots\}$ es el conjunto de las conexiones internas.

Al igual que en los casos previos, haciendo uso de ANTLR se construyeron las reglas del lenguaje del interprete a partir de descripciones gramaticales.

Como primera medida, se establecieron todos los elementos por los que va a estar conformado el archivo que contenga el modelo a analizar por la gramática. Entre ellos:

el nombre del archivo, los componentes que pueda importar, el nombre del modelo y su consecuente definición. De esta manera se estipula, por ejemplo, que el nombre del modelo es interpretado por ANTLR como un "ElementName", el cual puede ser una combinación alfanumérica con posibilidad de incluir guiones y guiones bajos. Así, un nombre que respete la gramática podría ser "Modelo-Ejemplo1" o "Prueba_Termodinámica"; no así "Modelo$%$" o "###Prueba###". Este módulo queda explicitado en nuestra gramática de la siguiente manera:

```
ElementName:[A-Za-z]([A-Za-z0-9_]|'- ')*( '(' [A-Za-z0-9_]* ')' )*;
```

De forma análoga, se establecen todos los elementos que componen nuestra estructura gramatical para que el intérprete delimite si el modelo está escrito de manera correcta o no.

Otro caso puede ser la definición del set de salidas externas (EOC), el cual como se puede observar en la definición formal, está compuesto por duplas, que a su vez poseen dos elementos: un componente (CompX o N) y el puerto utilizado (oPort1), todo encerrado entre corchetes {}.

Se dictamina entonces indicar esta notación de la siguiente manera:

```
couplingDefinition: ElementName'='{' coupling (',' coupling)* '}';

coupling: '(' couplingPair ',' couplingPair ')';

couplingPair: '(' ElementName ';' ElementName ')';
```

Donde se establece al conjunto de salidas como un `couplingDefinition` el cual está conformado por un nombre (`ElementName`) y por lo menos una dupla (`coupling`), la cual posee sus dos elementos (`couplingPair`) .

4.1 Ejemplo: "Enviroment"

Se presenta como ejemplo el modelo acoplado DEVS "Environment" tomado de [7], su definición formal en base a la GLC diseñada y la respuesta que se obtiene como descomposición de la entrada en relación con el rol que cumple cada elemento.

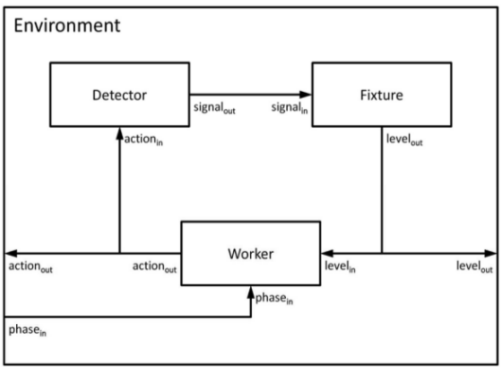


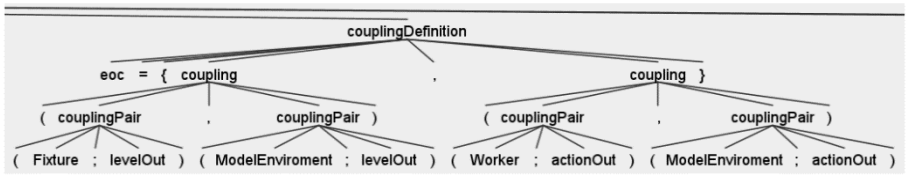
Fig. 6. Modelo DEVS acoplado "Environment" (tomado de [7]). Dicho modelo está formado por tres componentes (Detector, Fixture y Worker) y sus relaciones.

```
include ./modelo_Envi.devs as specification of MOD_Ambiente
import modelo-123 from ./modelo-B.devs

ModelEnviroment_1 = (x,y,componentes,definicionDeComponentes,eic,eoc,ic)

InPort = { phaseIn }
OutPort = { actionOut, levelOut }
x = { (phaseIn, phase) / phase belongs to Phases }
y = y_action union y_level where y_action = { (actionOut, v_actionOut) / v_actionOut belongs to Actions } , y_level = { (levelOut, v_levelOut) / v_levelOut belongs to Levels }
componentes = { Detector, Worker, Fixture }
definicionDeComponentes = { Detector is ModelDetector, Worker IS ModelWorker, Fixture is ModelFixture }
eic = { ( (ModelEnviroment; phaseIn), (Worker; phaseIn) ) }
eoc = { ( (Fixture; levelOut), ( ModelEnviroment; levelOut) ), ( (Worker; actionOut), (ModelEnviroment; actionOut) ) }
ic = { ( (Worker; actionOut), (Detector; actionIn) ), ( (Detector; signalOut), (Fixture; signalIn) ), ( (Fixture; levelOut), (Worker; levelIn) ) }
```

(a)



(b)

Fig. 7. (a) Definición del modelo acoplado propuesto en la Fig. 6 haciendo uso de la gramática diseñada. (b) Árbol sintáctico generado por ANTLR como descomposición del segmento "EOC" del modelo expresado en la Fig. 7(a).

Se puede observar en la Fig. 7(a) que el segmento "EOC" se define como:

```
eoc = { ( ( Fixture; levelOut ), ( ModelEnvironment; levelOut ) ),
        ( ( Worker; actionOut ), (ModelEnvironment; actionOut) ) }
```

Esto se puede interpretar y observar a partir de la Fig. 6 como las flechas que van desde el componente "Fixture" y "Worker" hacia fuera del modelo. Luego en Fig. 7(b) se muestra cómo el parser generado con ANTLR interpreta esta definición identificando ambas duplas y separando sus pares de origen y salida. Por ejemplo, la dupla que hace referencia a la relación de "Fixture" con "ModelEnvironment", el cual se lleva a cabo mediante el enlace "levelOut".

5 Conclusiones y Trabajos Futuros

En conclusión, en este trabajo se ha presentado el diseño de tres gramáticas libres de contexto para apoyar el modelado y simulación de eventos discretos. Estas GLC se implementaron utilizando la herramienta ANTLR y se diseñaron para facilitar el desarrollo de modelos de simulación basados en el marco de la ontología System Entity Structure (SES) y el formalismo Discrete Event System Specification (DEVS). La primera GLC se diseñó para analizar oraciones en inglés que describen modelos SES. La segunda se diseñó para especificar modelos de simulación de alto nivel basados en eventos, mientras que la tercera se diseñó para especificar modelos DEVS acoplados de manera formal.

Las gramáticas propuestas pueden utilizarse como soporte en el desarrollo de una futura herramienta de software integrada que facilite la vinculación entre las diferentes

capas de especificación requeridas durante el desarrollo de modelos de simulación. La herramienta tomaría como entrada una descripción en lenguaje pseudo-natural de un modelo y utilizaría el analizador sintáctico ANTLR para generar una especificación de modelo estructurada que se pueda utilizar para la simulación. Dicho proceso podría realizarse a través de diferentes etapas, donde primero se realice la especificación del entorno de modelado mediante el framework SES. Luego, partiendo de dicho entorno, sería posible definir el modelo en una abstracción de alto nivel del modelo de eventos discretos a partir de la entrada y por último, obtener un modelo formal expresado en términos del formalismo DEVS.

Este proceder descrito puede ser observado en la evolución de los ejemplos utilizados en las secciones de cada gramática, donde a partir de una descripción general del entorno a través de SES en la sección 2 (Fig. 4) para la entidad "Environment", se procede a la definición del modelo de alto nivel para una de las entidades que lo componen, denominada "Worker" en la sección 3 (Fig. 5) y a la posterior formalización en la sección 4 (Fig. 6(a)) mediante DEVS.

En resumen, este trabajo ofrece una contribución al campo de el modelado y simulación de eventos discretos al proporcionar un conjunto de GLC que pueden ayudar a simplificar el proceso de modelado. Futuros trabajos podrían incluir el desarrollo de una herramienta de software como la previamente mencionada y la evaluación de las gramáticas propuestas mediante el desarrollo de modelos de simulación basados en escenarios del mundo real.

Referencias

1. M. J. Blas, S. Gonnet and B. P. Zeigler. (2021). *Towards a Universal Representation of DEVS: A Metamodel-Based Definition of DEVS Formal Specification*. 2021 Annual Modeling and Simulation Conference (ANNSIM), Fairfax, VA, USA, 2021.
2. B. Zeigler, P. Hammonds. (2007). *Modeling & Simulation-Based Data Engineering*. Elsevier.
3. Jerry Banks, John S. Carson II, Barry Nelson. (2021). *Discrete-Event System Simulation – fifth edition*. Ed. Prentice-Hall.
4. Zeigler, B., Muzy, A. y Kofman, E. (2018). *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations – Third Edition*. Londres: Academic Press.
5. Parr, Terence (2022). *ANTLR*. Disponible en: https://www.antlr.org/
6. J. E. Hopcroft, R. Motwani y J. D. Ullman. (2007). *Introduction to Automata Theory, Languages and Computation*. Estados Unidos: Addison-Wesley.
7. Goldstein, R., Wainer, G. A., & Khan, A. (2014). The DEVS formalism. In Formal Languages for Computer Simulation. *Transdisciplinary Models and Applications* (62-102). IGI Global.