Área: Sistemas de Información e Informática
Categoría: Trabajo de Investigación – Estudiantes de Grado
Regional: Santa Fe

Una Gramática Libre de Contexto para la Especificación de Modelos Conceptuales

A Context-Free Grammar for Building Conceptual Models

Presentación: 04/09/2023

Victoria Meichtry Regner
Universidad Tecnológica Nacional - Facultad Regional Santa Fe Argentina.
meichtryv4@gmail.com

Resumen

Este trabajo tiene como finalidad la construcción de una herramienta que de soporte a la definición por etapas de un modelo de simulación. El objetivo es especificar una sintaxis que permita la definición de un modelo conceptual a partir de la abstracción de una situación del mundo real. Para esto, se establece una gramática que permite reconocer oraciones válidas y, a partir de ello, se permita la visualización de un conjunto de relaciones entre las entidades del modelo a través de un diagrama de clases. Para el desarrollo de cada etapa se utilizan como soporte, primeramente, ANTLR4 para definir reglas sintácticas con el fin de reconocer sentencias válidas. Además, se utilizó USE para realizar un diagrama de clases con el propósito de definir clases y establecer relaciones entre ellas. Por último, a través del lenguaje de programación Java, se pretende desarrollar las clases con sus relaciones propuestas en el diagrama.

Palabras clave: modelo conceptual, gramática libre de contexto, diagrama de clases, simulación de eventos discretos.

Abstract

This research paper is devoted to building a software tool that supports the definition of a simulation model. The objective is to specify a syntax that allows defining a conceptual model from an abstraction of some real-world situation. For this purpose, grammar is defined to recognize valid sentences and then, to get a set of relationships between entities building the model through a class diagram. Several software tools are used for such a development. First, ANTLR4 was used to define syntactic rules to recognize valid sentences in the grammar. Then, USE was used to make a class diagram for the purpose of defining classes and establishing relationships between them. Finally, Java was used to build the classes that allow getting an object diagram.

Keywords: conceptual modeling, context-free grammar, class diagram, discrete event simulation.

Introducción

La simulación es una herramienta poderosa en el proceso de diseño o aprendizaje de un proyecto, que puede utilizarse para resolver cualquier problema relacionado con un sistema real del cual se haya obtenido previamente un modelo. Un modelo es una representación de un objeto, sistema, o idea, cuyo propósito es ayudar a explicarlo, entenderlo o mejorarlo. Precisamente en la tarea de modelado se requiere de la habilidad para analizar un problema: consiste en resumir las características esenciales del problema, seleccionar y modificar las suposiciones básicas que caracterizan al sistema, y luego enriquecer y elaborar el modelo hasta obtener una aproximación útil.

El proceso de modelado consiste en distintas etapas. 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 donde se describe el problema y todos los elementos del mundo real que se relacionan con este. 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 ultima instancia, se especifica el modelo computacional el cual es la traducción de los modelos vistos a una herramienta de software especifica. Este proceso es iterativo, se repite tantas veces como sea necesario hasta obtener la aproximación del modelo que se está buscando según el objetivo de estudio y los conocimientos previos.

Luego de una breve descripción del proceso de modelado, nos centraremos en el estudio del modelado conceptual que es para el cual, en este trabajo, se pretende realizar una herramienta que facilite su construcción. Para esto, se propone desarrollar una sintaxis como forma de representación textual que permita definir un modelo de simulación asociado a eventos discretos. En función de esto, el objetivo final es, a partir de la definición de la sintaxis, reconocer sentencias 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 un modelo de objetos (definido como un diagrama de clases) a fin de definir las bases necesarias para dar una estructura correcta del modelo. Dicho diagrama de clases es implementado siguiendo la programación orientada a objetos para lograr instanciar las clases y establecer las relaciones entre cada objeto y su descripción en forma de texto.

Como soporte a este proyecto, las herramientas a utilizar serán ANTLR4 (ANother Tool for Language Recognition) (https://www.antlr.org/), USE (https://sourceforge.net/projects/useocl/) y Eclipse (https://eclipseide.org/). En primer lugar, ANTLR4 se utiliza puesto que toma una especificación formal de gramática como entrada y genera un analizador sintáctico. En segundo lugar, se decidió usar USE ya que permite realizar la especificación del diagrama de clases y, en último lugar, Eclipse ya que proporciona un conjunto de herramientas y características avanzadas para desarrollar software orientado a objetos.

Metodología

A fin de lograr una mejor comprensión del modelado conceptual, se van a desarrollar algunos conceptos necesarios para entender el desarrollo del trabajo. Dentro de la simulación de eventos discretos se encuentran 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.

Como se mencionó anteriormente, para lograr el desarrollo de este proyecto, se utilizaron una serie de herramientas. En un principio, se utilizó ANTLR4 que es una herramienta de generación de analizadores léxicos y sintácticos desarrollada por Terence Parr. ANTLR4 facilita la definición de gramáticas formales y la generación de analizadores automáticos a partir de esas gramáticas. Sobre la base de esta herramienta, y con el fin de aceptar sentencias

correctas que describan los elementos que se representan en un modelo de simulación de eventos discretos, se desarrolló una gramática libre de contexto.

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. Una gramática libre de contexto consta de un conjunto de reglas de producción que definen cómo se pueden combinar símbolos no terminales para formar cadenas de símbolos terminales. Cada regla se compone de un símbolo no terminal, el cual se reemplazará y una secuencia de símbolos terminales y no terminales que indican cómo se puede reemplazar el símbolo no terminal en la cadena.

En primer lugar, a partir de reglas léxicas se generó un analizador léxico que permitió reconocer tokens de un archivo de entrada. Luego, se desarrolló un analizador sintáctico (también conocido como “parser”), el cual construye una estructura de datos llamada árbol sintáctico que registra la forma en que se reconoció la estructura de la cadena de entrada (secuencia de tokens). Es decir, el árbol sintáctico se utiliza como representación del resultado de ejecución de una gramática con el fin de visualizar la relación entre las reglas y las sentencias de prueba.

En la gramática desarrollada solo se definieron algunos elementos de simulación, ya que resulta más sencillo trabajar con un conjunto reducido de elementos para lograr observar y corregir los distintos errores que pueden ir surgiendo. En cuanto a los elementos de simulación que se desarrollaron se encuentra entidad, estado y atributo. En el futuro, el objetivo es añadir los elementos restantes a la gramática.

Para comprender la gramática especificada, en la Figura 1 se puede observar las reglas de producción y el diagrama de sintaxis para el símbolo no-terminal *atributo*. Las reglas de producción nos permiten ver como fue escrita la sintaxis. En cambio, el diagrama de sintaxis representa la estructura que aplica una gramática a las cadenas de su lenguaje.

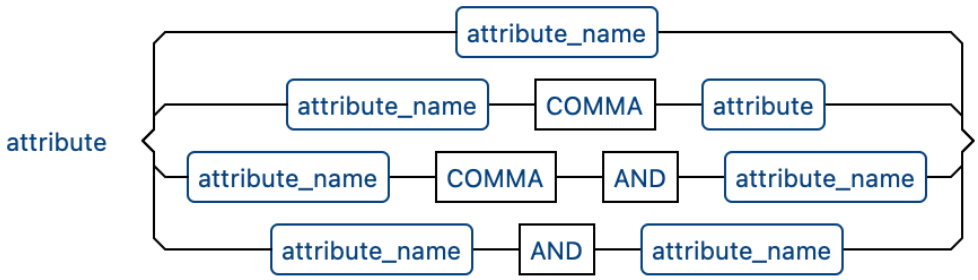


Figura 1. Diagrama de sintaxis para el símbolo no-terminal *atributo*.

Como complemento de los elementos incluidos en la gramática, se diseñó un modelo de clases para lo cual se utilizó USE. Esta es una herramienta que permite diseñar clases con sus atributos, operaciones y relaciones. Un diagrama de clases es una representación gráfica de la estructura estática de un sistema o una parte de él, que muestra las clases, sus atributos, métodos y las relaciones entre ellas. Una clase es un componente fundamental en la programación orientada a objetos que define un conjunto de propiedades (atributos) y comportamientos (métodos) comunes para un grupo de objetos. Luego, para lograr entender la relación que existe entre las distintas entidades, se utilizó un modelo basado en un diagrama de clases. Este diagrama se muestra en a Figura 2.

Como puede observase, el diagrama incluye como clases los mismos conceptos que los elegidos para armar la gramática. Además, como se puede visualizar en la Figura 2, se entiende que la entidad que se modela (Entity) está compuesta de un estado (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).

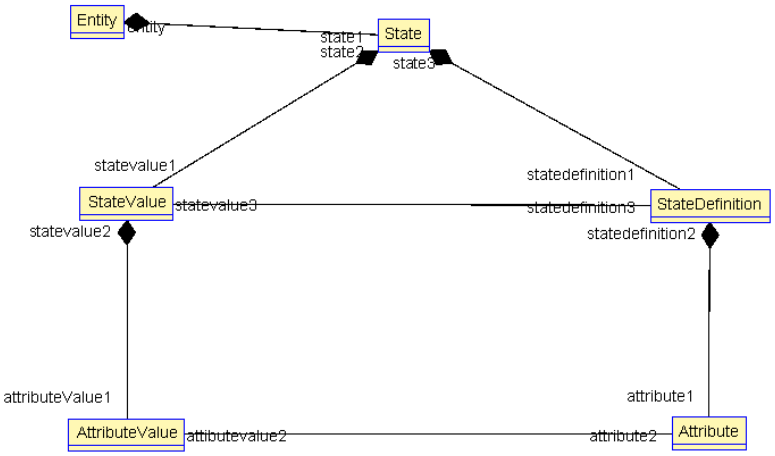


Figura 2. Diagrama de clases.

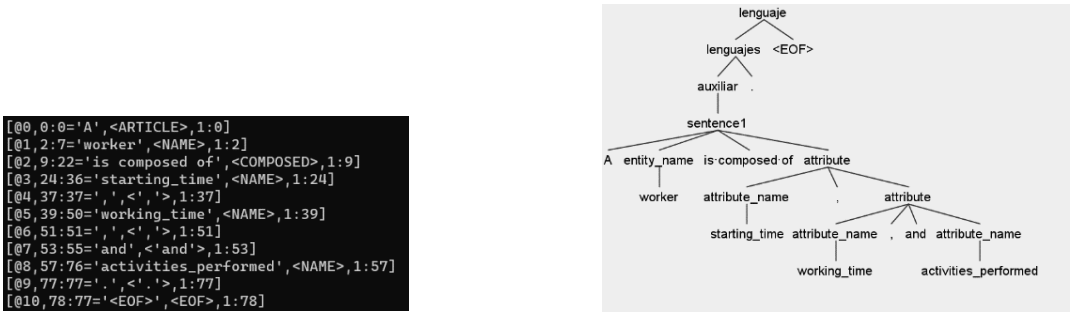
Sobre ambas definiciones (gramática y modelo de clases), la intención es mapear las sentencias de entrada (reconocidas a partir del parser generado con la gramática) con una instanciación del diagrama de clases, construyendo así un nuevo diagrama. En este caso, este es un diagrama de objetos ya que se centra en las interacciones entre instancias de las clases definidas en el diagrama de clases. Haciendo uso de un diagrama de objetos, es mas sencillo comprender las asociaciones entre entidades del modelo definido en la representación textual.

Resultados y Discusión

A continuación, se desarrolla un ejemplo a partir de un elemento denominado “Worker”, con el fin de entender el propósito y 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 léxico 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 Figura. 3(a). Por otra parte, el analizador sintáctico de dicha sentencia genera el árbol sintáctico que se presenta en la Figura. 3(b).



(a) (b)

Figura 3. (a) Tokens generados para la sentencia de ejemplo. (b) Árbol generado para la sentencia de ejemplo.

En la (Figura 3(b)), se puede observar las distintas partes que componen la derivación del árbol resultante que consta de los siguientes conceptos:

- Raíz del árbol (en la parte superior de la imagen) es el símbolo de inicio.
- Nodos internos (parte intermedia de la imagen) son los distintos símbolos no-terminales utilizados en la generación de las reglas de la gramática.
- Nodos hoja (parte inferior de la imagen) se logra visualizar los símbolos terminales, la sentencia de prueba planteada al comienzo.

Por lo tanto, la gramática logró reconocer que la sentencia era válida a partir de la derivación, es decir, de aplicar las reglas de la gramática. A partir del diagrama de clases indicado en la Figura 2, el ejemplo desglosado en el árbol sintáctico de la Figura 3(b) se mapea al diagrama de objetos indicado en la Figura 4.

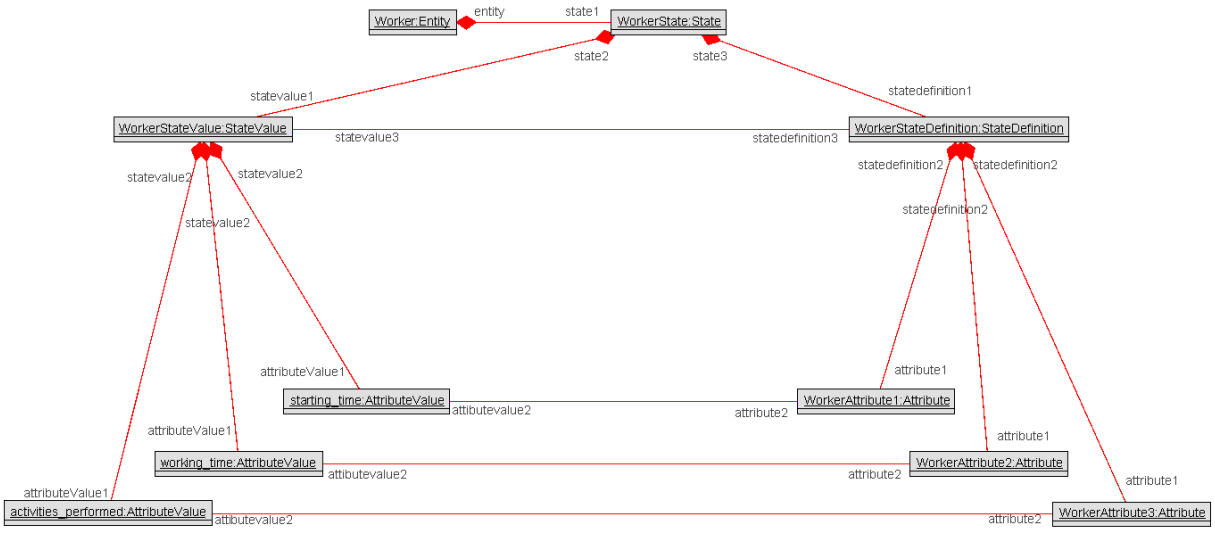


Figura 4. Diagrama de objetos.

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 el modelado conceptual que como se explicó anteriormente, se logra a través de una buena abstracción del modelo de diseño.

Hasta el momento se ha logrado un avance en cuanto a la especificación de las reglas sintácticas para la definición de la gramática que se utiliza para detallar el modelo conceptual y el diseño y mapeo del diagrama de clases para relacionar la definición textual con una representación gráfica bien conocida (UML). Se ha presentado un ejemplo sencillo basado en una única sentencia. Si se tuviese una cantidad mayor de oraciones, el tiempo de revisión manual sería grande. En esos casos, la herramienta reduce ese tiempo ya que solo es necesario ingresar un conjunto de comandos ANTLR para establecer la validez de las sentencias como representación textual del modelo conceptual. En un futuro, se pretende evolucionar el proyecto, agregado los elementos remanentes para lograr una representación más detallada y útil del modelo. Además, se aspira a lograr implementar las clases especificadas en el diagrama a través de Eclipse con el fin de poder instanciarlas en un lenguaje de propósito general.

Referencias Bibliografías

Robinson (2014). *Simulation: The Practice of Model Development and Use.*

Parr, Terence. (28 de septiembre de 2023). *Download*. ANTLR. https://www.antlr.org/

Slashdot Media. (28 de septiembre de 2023). *USE: UML-based Specification Environment*. SOURCE FORGE. https://sourceforge.net/projects/useocl/

Eclipse Foundation. (28 de septiembre de 2023). *ECLIPSE IDE*. Eclipse. https://eclipseide.org/

Jerry Banks, John S. Carson II, Barry Nelson. (2021). *Discret-Event System Simulation*. Estados Unidos: Ed. Prentice-Hall.

J. E. Hopcroft, R. Motwani y J. D. Ullman. (2007*). Introduction to Automata Theory, Languages and Computation.* Estados Unidos: Addison-Wesley.