Herramienta de Modelado Textual como Soporte al Formalismo RDEVS

Clarisa Espertino

Universidad Tecnológica Nacional – Facultad Regional Santa Fe
cespertino@frsf.utn.edu.ar

Resumen. En este trabajo se presenta una herramienta de software de modelado textual implementada como un complemento para Eclipse que facilita la definición de procesos de enrutamiento utilizando especificaciones textuales. Estas especificaciones en formato texto definen un modelo de simulación que admite la generación de código Java para situaciones de enrutamiento, a fin de que puedan ser ejecutadas en simuladores Discrete Event System Specification (DEVS) como modelos Routed DEVS (RDEVS). De esta manera, haciendo uso de una especificación textual un modelador no experto en RDEVS podrá obtener modelos de simulación de eventos discretos ejecutables. El núcleo de esta herramienta es una gramática libre de contexto que define la estructura del texto. Además, incluye un editor de texto que permite crear especificaciones textuales basadas en dicha gramática y ofrece ayudas de escritura para asistir al usuario durante la edición, cuenta con una herramienta para crear archivos de especificación y una opción de validación que permite procesar y validar el contenido escrito, para garantizar la consistencia de los modelos a obtener.

Palabras Clave: gramática libre de contexto, modelos de simulación, Routed DEVS.

1 Introducción

En la actualidad, muchos modelos de simulación son definidos a través de lenguajes de programación lejanos al formalismo matemático sobre el que están creados. En particular, para el formalismo Routed DEVS (RDEVS) [1] se deben utilizar librerías Java durante el proceso de modelado y simulación (M&S) para obtener modelos ejecutables. Es decir, el modelador debe generar código Java para definir sus modelos de simulación. Por esta razón, los modeladores que no cuentan con los conocimientos de programación necesarios pueden presentar dificultades al momento de especificar modelos RDEVS. En este contexto, es fundamental contar con entornos de M&S que brinden a los modeladores la posibilidad de especificar y editar modelos en distintos lenguajes, facilitando la obtención de sus contrapartes ejecutables (implementaciones) que luego puedan ser ejecutadas en simuladores concretos.

En este trabajo se presenta una herramienta de software para M&S implementada como un complemento para el entorno de desarrollo Eclipse [2], que permite especificar la estructura de procesos de enrutamiento de forma textual. El núcleo de esta

herramienta es una gramática libre de contexto acompañada de un metamodelo que define la estructura requerida. Además, la herramienta de software incluye: *i)* un editor de texto que da soporte al uso de la gramática ofreciendo ayudas de escritura, *ii)* un asistente para la creación de archivos que contienen especificaciones estructuradas y poseen una extensión determinada, y *iii)* un proceso de validación que deriva en una instancia del metamodelo propuesto (a partir de la cual se podrá generar el código Java asociado al escenario de simulación especificado, sin la necesidad de codificar en un lenguaje de programación específico).

Una gramática libre de contexto es una forma de describir lenguajes mediante reglas recursivas llamadas producciones. Consta de un conjunto de variables, un conjunto de símbolos terminales y una variable inicial, así como de producciones. Cada producción consta de una variable de cabeza y un cuerpo, formado por una cadena de cero o más variables y/o símbolos terminales [3]. Como se ha enunciado con anterioridad, la herramienta presentada se basa en una gramática libre de contexto que ha sido implementada con ANTLR4 [4] y ofrece al modelador una variedad de estructuras permitidas, tanto en inglés como en español. A través de ella se analiza sintácticamente el contenido creado con el editor de texto. Por otro lado, un metamodelo es una herramienta de modelado que permite asegurar la correctitud de la estructura de modelos. Se trata de un modelo que define el lenguaje utilizado para diseñar un modelo [5]. Como parte del desarrollo de esta herramienta se implementó una versión Ecore de un metamodelo de un proceso de enrutamiento. Luego, este metamodelo permite instanciar modelos de procesos de enrutamiento teniendo en cuenta un conjunto de restricciones requeridas que garantizan la posterior obtención de los modelos RDEVS equivalentes.

El resto del trabajo se encuentra estructurado de la siguiente manera. La Sección 2 presenta los conceptos de procesos de enrutamiento y modelos de red restringidos, y la relación entre ellos. La Sección 3 presenta la gramática libre de contexto sobre la que se basa la herramienta. La Sección 4 presenta el metamodelo utilizado para instanciar procesos de enrutamiento. La Sección 5 describe la herramienta propuesta para definir los procesos de enrutamiento junto con un ejemplo ilustrativo y detalles sobre las validaciones realizadas. Por último, la Sección 6 está dedicada a las conclusiones y trabajos futuros.

2 Procesos de Enrutamiento como una Conceptualización de Modelos de Red Restringidos

Un proceso de enrutamiento puede ser definido como “un sistema de componentes que interactúan, donde la operación de un componente y el ruteo de sus salidas depende de lo que sucede a lo largo del proceso” [6]. Luego, la operación interna de los distintos componentes que se utilizan como parte de proceso es independiente de la estructura del proceso que definen. Así, los componentes pueden decidir los destinos de sus salidas y tomar decisiones sobre el ruteo.

El formalismo RDEVS ha sido definido en [1] como una extensión del formalismo DEVS [7] que facilita la definición de procesos de enrutamiento en modelos de simulación basados en eventos discretos. Este formalismo estructura tres tipos de modelos:

modelo esencial, *modelo de ruteo* y *modelo de red*. La Tabla 1 presenta la descripción de cada uno de ellos.

Tabla 1. Descripción de los elementos RDEVS.

Elemento RDEVS	***Descripción***
Modelo Esencial	Funcionamiento de un componente elemental.
Modelo de Ruteo	Definición de un modelo esencial junto con una política de ruteo. Un mismo modelo esencial puede componer a múltiples modelos de ruteo.
Política de ruteo	Formalización de interacciones vinculadas a los componentes.
Modelo de Red	Conjunto de modelos de ruteo acoplados.

La teoría de redes propone modelar un sistema como un conjunto de nodos conectados a través de enlaces [8]. Luego, una red consiste en nodos conectados por un conjunto de enlaces. De acuerdo con lo definido en [9], si un proceso de enrutamiento es correctamente definido como un modelo de red restringido, los modelos RDEVS pueden ser obtenidos a partir de un conjunto de reglas de traducción. Es decir, es posible establecer una correspondencia entre los elementos de la teoría de redes con los modelos RDEVS (Tabla 2). A partir de esto, se puede establecer que cualquier definición de un modelo de red que cumple con ciertas restricciones para modelar un proceso de enrutamiento puede tomarse como base para la obtención de modelos RDEVS.

Tabla 2. Mapeo de elementos RDEVS a elementos de la Teoría de Redes (Conceptualización).

Elemento RDEVS	***Elemento Teoría de Redes***	***Descripción***
Modelo Esencial	Nodo	Un nodo puede ser mapeado a un modelo esencial, que define su comportamiento o funcionalidad.
Modelo de Ruteo	Nodo	Un nodo puede ser mapeado a un modelo de ruteo, que define su estructura o ubicación en la red.
Modelo de Red	Red	Un modelo de red representa un conjunto de nodos conectados todos contra todos. *
Política de ruteo (por nodo)	Enlaces	El conjunto de enlaces (entrantes/salientes) de un nodo son formalizados a través de políticas de ruteo.

* Los acoplamientos todos contra todos son requeridos por la definición del modelo de red. Esto permite que el ruteo se delegue al conjunto de funciones de ruteo.

Sobre la base de este mapeo, en este trabajo se propone una herramienta que facilite la traducción de modelos definidos según la teoría de redes como modelos de red restringidos a modelos de simulación RDEVS. La Figura 1 presenta la estructura de la herramienta en desarrollo. Como puede observarse, el modelador (haciendo uso de un editor de texto) realiza una especificación textual del modelo de red restringido que da soporte a su escenario de simulación. La gramática libre de contexto RDEVSNL

permite abstraer, por medio de su sintaxis, la definición de procesos de enrutamiento en representaciones textuales basadas en nodos y enlaces para describir su estructura. El metamodelo conceptualiza procesos de enrutamiento e incluye un conjunto de restricciones OCL para limitar el modelo de red original, dando lugar a un modelo de red restringido. Luego, la descripción textual definida por el modelador es analizada y validada a partir de la gramática y el metamodelo definidos. Posteriormente, la instancia del modelo validada es traducida a código Java para obtener su correspondiente representación Java de RDEVS que podrá ser ejecutada en simuladores DEVS.

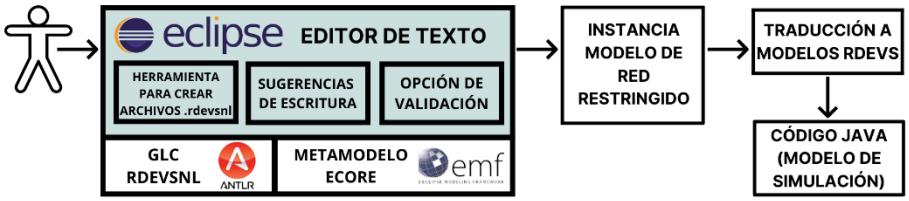

Fig. 1. Esquema del proceso de definición de modelos de simulación a partir del uso del Editor de Texto.

3 Gramática Libre de Contexto RDEVSNL

La gramática libre de contexto RDEVSNL, que ha sido presentada en un trabajo previo [10], se basa en un modelo de red restringido para abordar la definición de procesos de enrutamiento a través del formalismo RDEVS, abstrayendo la definición de dichos procesos en representaciones textuales que emplean nodos y enlaces para definir su estructura.

Se desarrollaron dos versiones de la sintaxis, una en inglés y otra en español, ambas especificadas e implementadas utilizando ANTLR4. En la especificación en inglés se pueden identificar tres bloques de construcción primarios: *network*, *materializes* y *edges*. Es a partir de estos símbolos no terminales que la sintaxis permite definir una red, asignarle un identificador y detallar la lista de nodos que están incluidos en ella (utilizando sentencias del bloque *network*), establecer el comportamiento que ejecutará cada nodo (relacionándolos con la operación de un componente empleando sentencias del bloque *materializes*) y vincular los diferentes nodos (es decir, definir enlaces entre ellos a través del bloque *edges*). Para visualizar los diagramas de sintaxis de la gramática y obtener mayores detalles, se sugiere revisar [10].

4 Metamodelo Ecore (EMF)

Se implementó una versión Ecore del metamodelo que se observa en la Figura 2. Este metamodelo especifica el conjunto de conceptos y relaciones que se mapean con la descripción textual del usuario a fin de instanciar un proceso de enrutamiento válido. Para esto, se utilizó el proyecto EMF de Eclipse [11], que consiste en una herramienta que provee un marco de trabajo para la definición de modelos. El modelo incluye un

conjunto de restricciones invariantes definidas en OCL que garantizan la obtención de un proceso de enrutamiento a partir de una definición de red basada en nodos y enlaces.

La Figura 2 presenta el metamodelo que conceptualiza procesos de enrutamiento, donde los estereotipos son utilizados para indicar el componente de red al que refiere el elemento de ruteo. Una especificación RDEVSNL (*NLSpecification*) incluye un proceso de enrutamiento (*Routing Process*). Este concepto refiere al modelo de red (*Network Model*). El modelo de red restringido utilizado para estructurar un proceso de enrutamiento se define sobre un conjunto de nodos (*Node*) que denotan componentes (*Component*). Cada uno de ellos ejecuta el comportamiento de un tipo de componente (*ComponentType*). A su vez, el proceso de enrutamiento se compone de un conjunto de interacciones dirigidas entre componentes (*Link*), donde uno de ellos actúa como fuente (*source*) y el restante actúa como destino (*destination*). Por otra parte, las referencias resaltadas en color representan las restricciones OCL incluidas en el modelo para garantizar su integridad: *i)* al menos un nodo debe identificarse como inicial, *ii)* al menos un nodo debe identificarse como final, *iii)* los nodos no pueden estar aislados, *iv)* varios enlaces no pueden conectar a los mismos nodos, *v)* los auto-enlaces no están permitidos, *vi)* un componente debe ejecutar un único comportamiento y *vii)* en una especificación RDEVSNL se debe describir a un único proceso de enrutamiento.

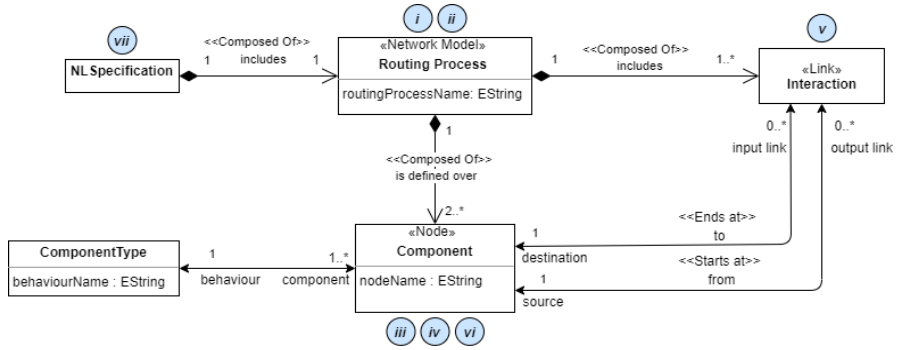

Fig. 2. Metamodelo del dominio instanciado a partir de especificaciones textuales creadas utilizando la sintaxis RDEVSNL.

5 Plugin de Eclipse para la Construcción de Especificaciones en RDEVSNL

Sobre la base de la gramática introducida en la Sección 3 y el metamodelo presentado en la Sección 4, la herramienta propuesta fue implementada como un plugin para la plataforma Eclipse. Se conforma de un editor de texto que permite instanciar modelos de red restringidos válidos haciendo uso de la sintaxis RDEVSNL, un “wizard” para la creación de archivos con extensiones válidas para el almacenamiento de las descripciones textuales y un proceso de validación que instancia el metamodelo nombrado previamente a fin de verificar la descripción textual. El editor de texto brinda ayudas

durante la edición, respetando la correspondencia con el idioma de trabajo seleccionado por el modelador.

El modelador puede seguir los pasos indicados en el "wizard" para la plataforma Eclipse (que fue incluido en el plugin) para crear un archivo de especificación RDEVSNL, que tendrá extensión ".rdevsnl". Puede, dentro de un proyecto Java en Eclipse, asignar un nombre al archivo y seleccionar el idioma con el que trabajará. Esto último se asocia con el hecho de que la gramática RDEVSNL cuenta con dos versiones: una en inglés y otra en español. Así, esta herramienta soporta la escritura y validación de las especificaciones en ambos idiomas. Si el modelador no selecciona ningún idioma, por defecto trabajará en inglés.

5.1 Definición de la Especificación Textual

El editor incluido en el plugin cuenta con ayudas de escritura que facilitan al modelador la edición de sus especificaciones. Entre ellas se encuentra el resaltado de sintaxis. Para lograrlo, las palabras admitidas se clasifican en: *artículos, símbolos no terminales, acciones, comentarios* e *interacciones*. De esta manera, al escribir la especificación, el modelador puede identificar claramente los componentes de cada sentencia a través de sus colores. Por ejemplo, palabras como "*network*", "*node*" o "*component*" (que representan los símbolos no terminales) se resaltan en azul, mientras que palabras como "*includes*", "*sends*" o "*receives*" (que representan acciones) toman el color rojo. Además, puede incluir comentarios cuyo contenido no será analizado durante el proceso de validación de la especificación y se resaltará en color verde. Los mismos pueden ser definidos en una sola línea o en múltiples líneas. En los comentarios de una única línea, deberán incluirse los caracteres "//" al principio. En cambio, si se trata de un comentario de múltiples líneas, se deben incluir los caracteres "/*" al comienzo y "*/" al finalizar.

Con respecto a las sugerencias de escritura, las mismas pueden obtenerse presionando CTRL+SPACE. El modelador podrá seleccionar de a una palabra por vez sobre una lista de palabras sugeridas, haciendo clic en ella o navegando con las flechas de dirección del teclado y presionando ENTER sobre la palabra deseada. Las listas de palabras que se muestran son almacenadas en archivos de configuración, existiendo uno por cada idioma disponible.

El idioma de trabajo elegido por el modelador durante el proceso de creación de su archivo de especificación es almacenado junto con sus metadatos como un atributo definido por el usuario. Cuando el archivo es creado, se asigna el valor correspondiente al atributo idioma. Esto permite establecer una correspondencia entre las ayudas de escritura y el idioma con el que trabaja el modelador, ya que cada vez que se abre el archivo, se lee de sus metadatos el valor del atributo idioma y se cargan las ayudas requeridas en consecuencia a dicha elección.

Para clarificar la descripción de la herramienta, se presenta un ejemplo. La Figura 3 muestra el proceso de enrutamiento a definir. Puede observarse que la red está conformada por 8 nodos (representados por máquinas), que se relacionan a través de interacciones dirigidas, y cada uno se corresponde con un tipo de máquina (es decir, ejecuta el comportamiento de un tipo de máquina). Luego, en la misma figura se muestra la

definición de su estructura a partir de las sentencias que ofrece la sintaxis RDEVSNL, haciendo uso del editor de texto.

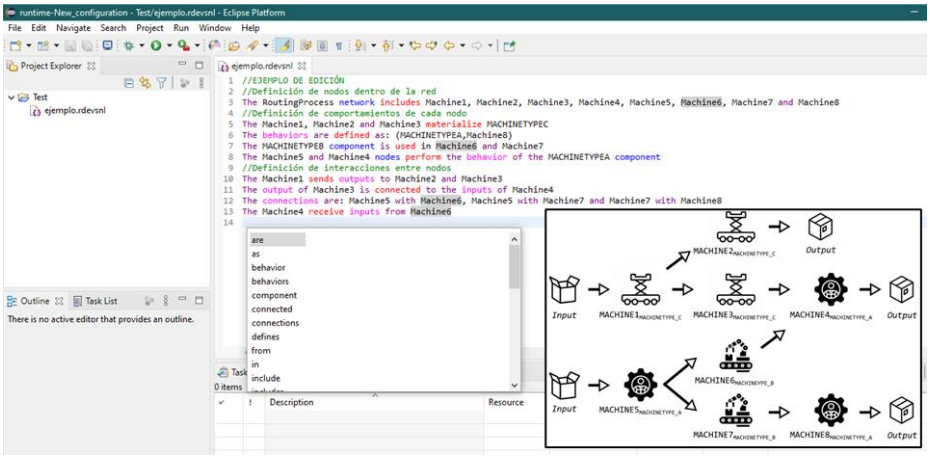

Fig. 3. Captura de pantalla del Editor de Texto incluido en el plugin para admitir la sintaxis. El resaltado de sintaxis y las sugerencias de escritura se hacen visibles, en inglés para este caso. Abajo a la derecha: representación gráfica del proceso de enrutamiento definido en la especificación textual.

5.2 Validación de la Especificación Textual

El plugin implementado cuenta con la opción de validar el archivo de especificación, utilizando el analizador sintáctico (o parser) creado con ANTLR4. Cuando el modelador activa el proceso de validación presionando la opción ofrecida en el menú de opciones (*"Check RDEVSNL Specification"*), el análisis sintáctico de RDEVSNL es ejecutado sobre el contenido actual del archivo "*.rdevsnl". El mismo fue previamente guardado en forma automática para asegurar que el análisis se realiza sobre su versión más reciente. Luego, el parser trata de reconocer las estructuras de las sentencias a partir de un flujo de tokens, dado por el contenido actual de la especificación.

Si dicho análisis es exitoso (es decir, todas las sentencias que el modelador utilizó para crear su especificación son válidas), utilizando los tokens identificados por el parser se crea automáticamente una instancia del metamodelo Ecore presentado en la Figura 2. ANTLR genera, junto con otras clases Java, una interfaz que contiene los métodos necesarios para recorrer el árbol de sintaxis y manejar eventos como ingresar a una regla de producción o salir de ella. Haciendo uso de dicha interfaz se evalúan las estructuras empleadas y se reconocen los elementos que la componen. Así se identifican las clases del metamodelo que deben instanciarse, junto con los valores a asignar a sus respectivos atributos. Cada elemento definido en la especificación es mapeado a un concepto o relación.

Para el ejemplo de la Figura 3, a partir del contenido de la línea 3 se creará una instancia del concepto *NL Specification* que contendrá una instancia de *RoutingProcess* llamada *"RoutingProcess"*, ocho instancias de *Component* que guardan en el atributo

nodeName los valores *"Machine1"* hasta *"Machine8"* y las relaciones *IsDefinedOver* para vincular a cada *Component* con *RoutingProcess*. Luego, se crean las relaciones de cada *Component* con las tres instancias de *ComponentType* (a las cuales se les asignaron los valores *"MACHINETYPEA"*, *"MACHINETYPEB"* y *"MACHINETYPEC"* en el atributo *behaviourName*) analizando las líneas 5 a 8. Por último, se mapean los diferentes enlaces (*Interaction*) que se establecen entre los *Component* en las líneas 10 a 13. A cada relación se le especifica el componente que actúa como origen y el que actúa como destino. Cuando una sentencia incluye una lista de nodos o de conexiones, se utilizan estructuras iterativas para recorrerla y crear todas las instancias referidas en ella, cada una con sus correspondientes atributos. La Figura 4 muestra las diferentes instancias de las clases del metamodelo del dominio que fueron creadas al analizar la especificación textual del ejemplo en análisis.

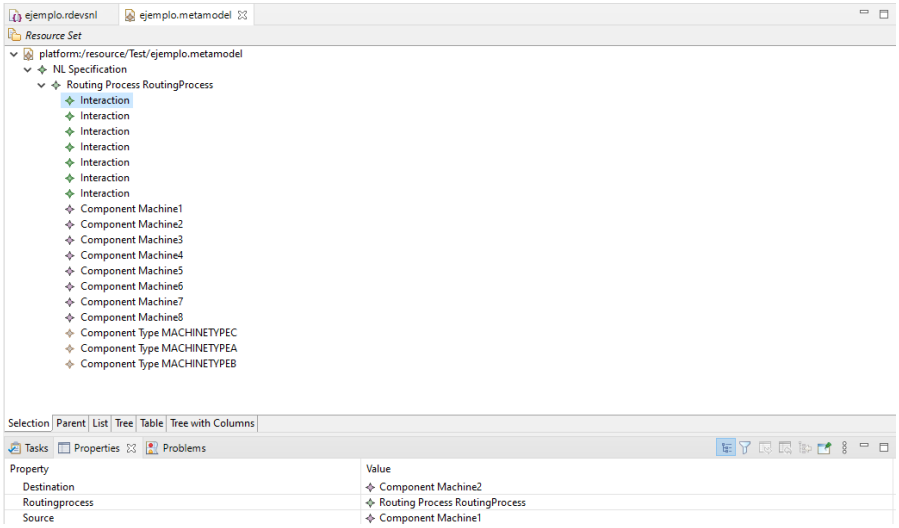

Fig. 4. Instancia del metamodelo Ecore presentado en la Figura 2, obtenido de la validación de la especificación del ejemplo presentado en la Figura 3.

Por otro lado, si se identifican errores sintácticos, se mostrará un mensaje de error. En la Vista *Problems* de Eclipse podrá encontrarse el detalle de estos para localizarlos y posteriormente corregirlos.

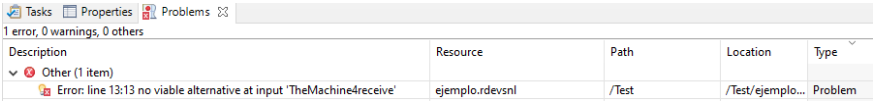

Fig. 5. Vista *Problems* de Eclipse informando el error sintáctico asociado a la línea 13 de la especificación de la Figura 3, que se origina al reemplazar la palabra *"receives"* por *"receive"*.

Teniendo ya creada la instancia del metamodelo a partir de la descripción textual, el plugin ejecuta su validación para asegurar su correctitud. Aquí se verifican los conceptos, relaciones, multiplicidades y restricciones de OCL del metamodelo sobre la

instancia obtenida. Si no se encuentran errores, el modelador recibirá un mensaje de éxito. De lo contrario, podrá visualizar el error identificado (como en la Figura 5). Contando con la posibilidad de acceder a las propiedades del error (como muestra la Figura 6), podrá corregir su descripción del proceso de enrutamiento y volver a realizar las comprobaciones.

En este punto es importante destacar que los elementos utilizados del proceso de enrutamiento corresponden a los modelos RDEVS que definen estructura. El comportamiento interno de las máquinas en el ejemplo presentado no se especifica junto con la descripción textual de la estructura del escenario.

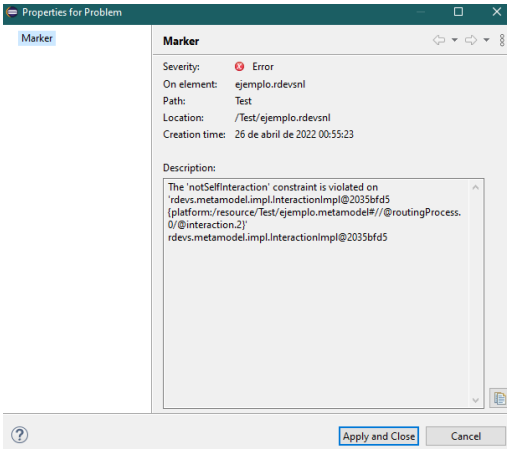

Fig. 6. Propiedades del error mostrado en la Vista Problems de Eclipse, en este caso originado al modificar la línea 11 de la especificación de la Figura 3 por *"The output of Machine3 is connected to the inputs of Machine3"*. Por la restricción *v)* presentada en la Figura 2, esta situación no es admitida.

6 Conclusiones y Trabajos Futuros

Se ha presentado una herramienta de software para M&S implementada como un complemento para el entorno Eclipse. Sus funcionalidades asisten a los modeladores durante el proceso de M&S, ofreciéndoles un entorno de modelado donde pueden definir y validar la estructura de procesos de enrutamiento a través de especificaciones textuales, abstrayéndolos así de las complejidades matemáticas propias del formalismo y aquellas asociadas a la programación de sus modelos.

A través de esta herramienta los modeladores podrán obtener, desde la creación de especificaciones textuales de procesos de enrutamiento como redes, modelos de simulación RDEVS sin la necesidad de emplear lenguajes de programación. La gramática RDEVSNL ofrece, a partir de las sentencias permitidas, un lenguaje más cercano al modelador que ayuda a entender con más claridad lo que se está modelando y permite compartirlo fácilmente entre los participantes del proceso de M&S, sin que tengan conocimientos específicos.

Es importante destacar que las herramientas existentes que ofrecen funcionalidades similares están asociadas al empleo de DEVS y no son directamente aplicables a RDEVS, por tratarse este último de un formalismo reciente.

El trabajo futuro está dedicado a la generación automática del código Java asociado a los modelos de simulación RDEVS, a partir de la traducción del proceso de enrutamiento descrito en la instancia del modelo de red obtenido. La misma se desarrollará como la ya implementada en [12].

Referencias

1. Blas, M., Gonnet, S. y Leone, H. “Routing Structure over Discrete Event System Specification: A DEVS Adaptation to Develop Smart Routing in Simulation Models”, En Actas de Winter Simulation Conference, Las Vegas, USA, Diciembre 2017, pp. 774-785.
2. The Eclipse Foundation. Eclipse. Disponible: https://www.eclipse.org/. Accedido por última vez el 22/5/2021.
3. Hopcroft, J.E., Motwani, R. y Ulman, J.D. Introduction to Automata Theory, Languages and Computation. 3era ed. Madrid: Pearson, 2007, pp. 143-184.
4. ANTLR4 IDE Eclipse Plugin para ANTLR4. ANTLR. Disponible: https://www.antlr.org/tools. Accedido por última vez el 22/5/2021.
5. OMG. Meta Object Facility (MOF) Specification, versión 1.4, 2002.
6. Alshareef, A., Blas, M.J., Bonaventura, M., Paris, T., Yacoub, A., Zeigler, B.P. (2022). Using DEVS for Full Life Cycle Model-Based System Engineering in Complex Network Design. En: Nicopolitidis, P., Misra, S., Yang, L.T., Zeigler, B., Ning, Z. (eds) Advances in Computing, Informatics, Networking and Cybersecurity. Lecture Notes in Networks and Systems, vol 289. Springer, Cham. https://doi.org/10.1007/978-3-030-87049-2_8
7. Zeigler, B., Muzy, A. y Kofman, E. Theory of modeling and Simulation: Discrete Event & Iterative System computational Foundations, Academic Press, 3era ed., 2018.
8. Newman, M., Barabasi, A.-L. y Watts, D.J. The Structure and Dynamics of Networks, Princeton University Press, 2006.
9. Blas, M., Espertino, C. y Gonnet, S. “Modeling Routing Processes through Network Theory: A Grammar to Define RDEVS Simulation Models”, En Actas de 3rd Workshop on Modeling and Simulation of Software-Intensive Systems, 2021, https://doi.org/10.5753/mssis.2021.17255.
10. Espertino, C., Blas, M. y Gonnet, S. “Especificación de Modelos de Simulación RDEVS: Diseño e Implementación de una Gramática Libre de Contexto”, En Actas de 9° Congreso Nacional de Ingeniería Informática/Sistemas de Información CoNaIISI, 2021. ISBN 978-950-42-0213-4.
11. The Eclipse Foundation: Eclipse Modeling Project. Eclipse Modeling Framework. Disponible: https://www.eclipse.org/modeling/emf/. Accedido por última vez el 22/5/2022.
12. Blas, M. y Gonnet, S. “Computer-aided Design for Building Multipurpose Routing Processes in Discrete Event Simulation Models”, Engineering Science and Technology, an International Journal, 24, 2021, pp. 22–34. https://doi.org/10.1016/j.jestch.2020.12.006.