Desarrollo de una herramienta de software basada en Java para la captura de eventos en la simulación de modelos RDEVS

Mateo Toniolo[1]

Supervisores: Dra. María Julia Blas[1,2], Dr. Silvio Gonnet[1,2]

1 Universidad Tecnológica Nacional-Facultad Regional Santa Fe
2 Instituto de Desarrollo y Diseño INGAR (UTN-CONICET)

mateojustotoniolo@gmail.com
mariajuliablas@santafe-conicet.gov.ar
sgonnet@santafe-conicet.gov.ar

Resumen. Routed DEVS (RDEVS) es una subclase del formalismo de modelado y simulación Discrete Event System Specification que facilita el tratamiento de problemas centrados en el ruteo de eventos. Estos formalismos tienen un carácter abstracto, siendo de suma necesidad poder diseñar y ejecutar modelos basados en ellos, pero también disponer de herramientas que ayuden al procesamiento de sus resultados. En este trabajo se explica el modelado e implementación de una herramienta para la estructuración y captura de los eventos que tienen lugar cuando se ejecuta la simulación de modelos RDEVS. El proyecto se desarrolla utilizando Java para el seguimiento de los componentes sumando el uso de la herramienta GWT para visualizar toda la información recopilada. El objetivo del producto de software final es crear registros de datos que permitan mejorar la comprensión del proceso de simulación junto con el comportamiento de los modelos componentes.

1 Introducción

El formalismo Discrete Event System Specification (DEVS) [1] es un formalismo de modelado basado en la teoría de sistemas que proporciona una metodología general para la construcción jerárquica de modelos de simulación reutilizables de forma modular. El núcleo de DEVS incluye un framework de Modelado y Simulación (M&S) estructurado en base a tres componentes: modelo, simulador y marco experimental. Cada uno de estos componentes debe ser visto como una entidad independiente que actúa con el objetivo de dar lugar al proceso de M&S. En este sentido, el modelo describe la especificación del sistema según su comportamiento y estructura. Por su parte, el simulador refiere al sistema computacional que se encarga de ejecutar las

instrucciones del modelo. Finalmente, el marco experimental representa las condiciones bajo las cuales el sistema es observado, dando lugar al contexto de experimentación y validación del modelo.

Este trabajo centra su atención en el formalismo Routed DEVS (RDEVS) [2], el cual ha sido desarrollado recientemente como una nueva extensión del formalismo DEVS. Esta extensión facilita el modelado y simulación de problemas centrados en el ruteo de eventos sobre modelos de simulación discretos. En este contexto, RDEVS actúa como una capa sobre los modelos DEVS que provee funcionalidad de ruteo al modelador sin requerir "bajar" al modelo DEVS para especificar dicho comportamiento [1]. El objetivo de este proyecto es generar una herramienta de software que permita visualizar el estado de una simulación RDEVS en tiempo real. Teniendo en cuenta que los eventos en modelos RDEVS son ruteados en tiempo de ejecución, además de los resultados de simulación propios del problema (requeridos por el modelador), es de interés obtener información referida a los eventos intercambiados por los distintos modelos. El flujo de eventos que tiene lugar en un proceso de ruteo dado puede brindar información valiosa referida al comportamiento dinámico de las políticas de ruteo definidas y/o de la forma de interacción establecida entre los modelos de simulación ubicados en los distintos niveles.

En este trabajo se presenta el modelado e implementación en Java (específicamente, como un plugin de Eclipse [3]) de un módulo de software que permite crear rastreadores (es inglés, trackers) asociados a los distintos componentes de una simulación RDEVS. Teniendo en cuenta la división propuesta en el framework de M&S que acompaña a DEVS, estos rastreadores son embebidos en los modelos de simulación a fin de garantizar un funcionamiento transparente para el simulador. Este módulo de software es complementado con GWT [4] como herramienta de soporte para la generación de gráficos. Es decir, para visualizar la información es necesario haber capturado todos los datos necesarios previamente. En este sentido, el módulo de captura de datos actúa como núcleo del funcionamiento de la herramienta final. De esta manera, el fin del proyecto es mejorar la comprensión de los eventos que tienen lugar durante un proceso de simulación discreta y, en este caso particular, de modelos RDEVS.

Este trabajo se organiza de la siguiente manera. En la sección 2 se explica el formalismo RDEVS y se presenta la librería utilizada para su simulación. Luego, en la sección 3, se explica el desarrollo e implementación de la herramienta donde se describe el uso de rastreadores en los módulos de captura de datos y posterior visualización de la información. Por último, en la sección 4 se presenta una conclusión acerca del trabajo junto con posibles ideas para el avance del proyecto a futuro.

2 Formalismo Routed DEVS (RDEVS)

El formalismo RDEVS es una subclase del formalismo DEVS que busca dar solución a la identificación de eventos como funcionalidad embebida dentro de modelos de simulación basados en eventos discretos. Frecuentemente, el objetivo de los modelos de simulación DEVS plantea la necesidad de identificar el origen y/o indicar el destino de los eventos a fin de garantizar su correcto procesamiento. Este problema se

conoce como problema de ruteo de eventos (o, simplificado, problema de ruteo). Un problema de ruteo afecta el diseño del modelo de simulación, pero no se encuentra específicamente vinculado al comportamiento de los componentes a diseñar. Es decir, a nivel del escenario a modelar (dominio), los componentes no presentan una característica propia de ruteo de mensajes. Sin embargo, a nivel del modelo de simulación (diseño), los modelos que representan los distintos componentes requieren funcionalidad de ruteo para los eventos a intercambiar. En este contexto, RDEVS actúa como una capa sobre DEVS que provee funcionalidad de ruteo sin necesidad de que el modelador recurra a nuevas especificaciones DEVS para lograr estas funcionalidades. De esta forma, la incorporación de dicha funcionalidad como parte del formalismo de simulación, reduce la complejidad de diseño ya que el modelador no debe incorporar nuevos módulos de simulación para el manejo de las rutas. Esto es, RDEVS actúa como una abstracción entre el dominio y el diseño que ayuda a generar modelos de simulación reutilizables reduciendo el tiempo de trabajo requerido para la obtención del modelo de simulación final [5].

El formalismo DEVS define dos tipos de modelos: modelos atómicos y modelos acoplados. Por su parte, el formalismo RDEVS define tres tipos de modelos de simulación: modelo esencial, modelo de ruteo y modelo de red. Un **modelo esencial** representa el comportamiento de un componente. Este tipo de modelos se los define mediante un DEVS atómico. Un mismo modelo esencial se corresponde con uno o más componentes. Cuando hablamos de componentes hacemos referencia a los **modelos de ruteo**. Un modelo de ruteo define una entidad capaz de definir el destino/origen de sus entradas/salidas por medio de una política de ruteo. Es decir, un modelo de ruteo es una estructura que asocia un modelo esencial con una política de ruteo definida. El formalismo indica que los modelos de ruteo se deben conectar mediante acoplamientos todos contra todos (excepto consigo mismos). El alcance de las rutas formadas por estos acoplamientos se limita en el **modelo de red.** El modelo de red es una entidad compuesta por modelos de ruteo, siendo el encargado de recibir mensajes provenientes de otros modelos de red y distribuirlos a los modelos internos en caso de ser aceptados.

En la Figura 1 se muestra como ejemplo el proceso de verificación de productos en una fábrica. El proceso inicia con la primera máquina de selección. Por cada producto elegido hay dos opciones posibles: *i)* si el paquete se encuentra en buen estado se envía a la máquina de embalaje para luego ser enviado a la salida (escenario en azul); o *ii)* si el paquete debe ser reparado se envía a la máquina de reparación para luego ser nuevamente seleccionado y continuar con el proceso de embalaje y salida (escenario en rojo). En la Figura 2 se observa un esquema de los modelos RDEVS requeridos para este ejemplo. Para construir este esquema primero se debe definir un modelo DEVS atómico para cada tipo de máquina (es decir, las etapas que involucran las mismas máquinas tendrán el mismo modelo esencial). Siguiendo el ejemplo, se repite en este caso la máquina de selección y por lo tanto se muestra dos veces el modelo esencial N°1. Luego por cada etapa del proceso se crea un modelo de ruteo y, por definición del modelo de red, se conectan todos los modelos de ruteo mediante acoplamientos todos contra todos. Las políticas de ruteo quedan establecidas según los vínculos definidos en el escenario.

En la Figura 2 se representan los modelos de ruteo mediante los óvalos verdes, conformados por un modelo esencial (cuadrado blanco) y las políticas asociadas mediante diplomas. Algunos modelos de ruteo poseen más de una política, esto se da cuando un modelo forma parte de dos caminos distintos (por ejemplo, la máquina de embalaje forma parte de ambos caminos por lo que en su modelo asociado se observan dos diplomas ya que posee dos políticas de ruteo diferentes, una asociada al camino rojo y otra al camino azul). Estos componentes entonces definen el modelo de red. La política del modelo asociado a la máquina de salida es distinta a las demás ya que conecta su salida directamente con la salida del modelo de red.

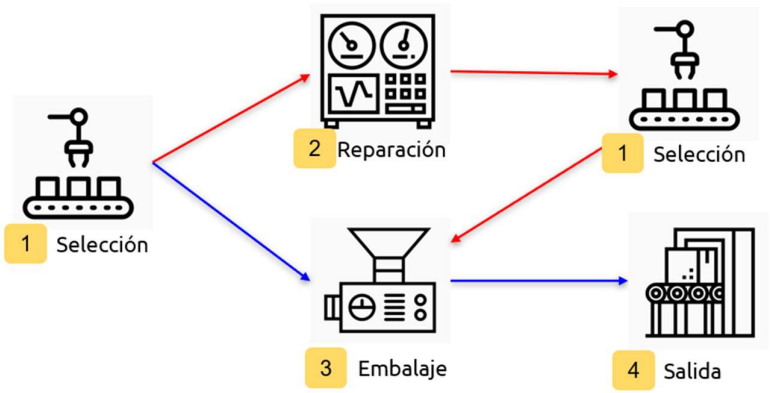


Fig. 1. Proceso de verificación de productos en una fábrica

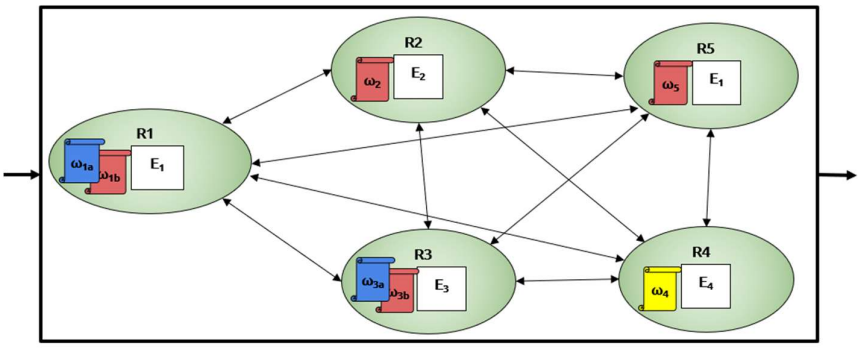


Fig. 2. Modelo del ejemplo de la figura 1 utilizando RDEVS

2.1 Librería Java

Dado que el formalismo RDEVS se encuentra diseñado como una subclase de DEVS, las implementaciones disponibles del formalismo original pueden ser utilizadas como base para desarrollar implementaciones propias de RDEVS. En este contexto, con el objetivo de proporcionar un entorno de software que ayude al modelado y simulación en este nuevo formalismo, en [6] se propuso un framework de software basado en Java para RDEVS. Esta herramienta brinda el respaldo necesario para la implementación y simulación de estructuras de ruteo en modelos de sistemas de eventos discretos basado en el formalismo RDEVS.

El framework propuesto utiliza DEVSJAVA [7] como herramienta de software subyacente. Las clases Java implementadas como conceptos propios de RDEVS se vinculan con las clases propuestas en la herramienta original a fin de garantizar su compatibilidad con fines de simulación. Luego, el diseñador debe instanciar una serie de clases (pertenecientes a la librería) de acuerdo con las necesidades requeridas como parte de su modelo RDEVS a fin de implementar un modelo de simulación ejecutable en el entorno Java.

Cada tipo de modelo incluido en el formalismo RDEVS queda definido haciendo uso de una clase específica. Siguiendo el ejemplo presentado en el apartado 2 los modelos esenciales asociados a las máquinas serían definidos a partir de la clase EssentialModel.java, los modelos de ruteo asociados a cada máquina ubicada en un componente serían definidos a partir de la clase RoutingModel.java, y el modelo de red sería definido a partir de la clase NetworkModel.java. El resto de los elementos requeridos con relación al formalismo fueron definidos en clases especificas a fin de dar soporte al proceso de M&S.

Además de la implementación a nivel de código Java, la herramienta de software diseñada tiene la posibilidad de ser complementada con representaciones gráficas de los modelos desarrollados. DEVS Suite [8] es un entorno gráfico que se utiliza para representar modelos de DEVS implementados con DEVSJAVA. Teniendo en cuenta que el proyecto RDEVS extiende un subconjunto de las clases definidas en el proyecto DEVSJAVA, esta herramienta gráfica también puede aplicarse para la representación visual de los modelos de simulación RDEVS.

3 Implementación y Desarrollo

3.1 Definición de los Trackers

La vista estática de los modelos RDEVS se corresponde con las representaciones descriptas en el ejemplo de la sección anterior. Por su parte, la vista dinámica se vincula no sólo con la visualización del modelo durante el proceso de simulación sino también con la forma en la cual el modelo evoluciona a través del tiempo. Aunque el visualizador de DEVS Suite [8] puede ser utilizado para contemplar la forma en la cual evoluciona la simulación a lo largo del tiempo, este simulador brinda soporte a la ejecución de modelos DEVS. Luego, su utilidad para el caso de modelos RDEVS se ve limitada por la introducción de las políticas de ruteo.

Con el objetivo de capturar los datos asociados a este tipo de simulación, tomando como punto de partida la librería detallada en la sección 2.1, se decidió utilizar trackers (o rastreadores) asociados a los distintos componentes del formalismo. Estos trackers se definen en un proyecto como clases Java con el objetivo de almacenar información estructurada a lo largo de una simulación. La Figura 3 refleja en el diagrama de clases UML implementado para dar soporte al esquema de trackers.

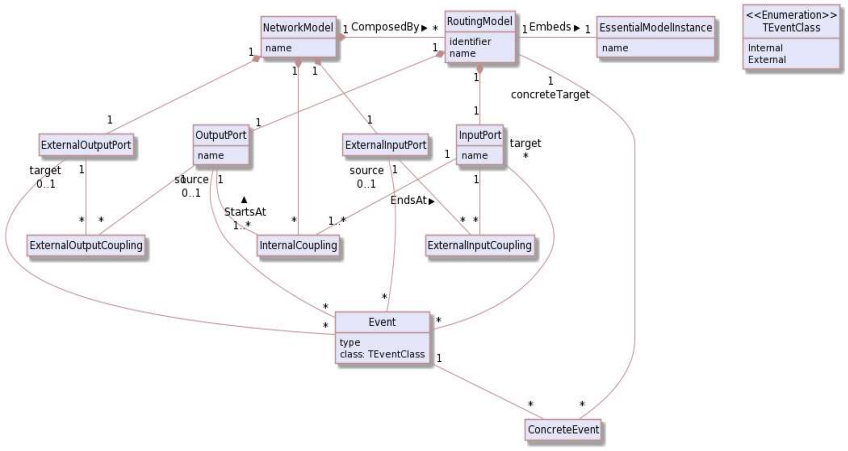


Fig. 3. Diagrama de clases sobre la estructura de los trackers y las relaciones entre ellos.

Los rastreadores implementados se dividen en cuatro tipos: Modelo, Puerto, Acople y Evento (o Mensaje). Las clases que representan a los modelos (Esencial, Ruteo y Red) tienen la función de almacenar los datos que identifican al modelo y los trackers de las entidades del nivel inferior que lo componen. Excepto por el esencial, el resto de los modelos tienen asociado un puerto de entrada (Input Port) y uno de salida (Output Port) para la recepción y el envío de mensajes. Por lo tanto, cada uno de estos mismos posee un tracker que lleva registro de los mensajes que han entrado/salido por el mismo. Como ya se mencionó antes, el formalismo RDEVS ordena que los modelos de ruteo se conecten por medio de acoplamientos. Cada uno de estos acoplamientos (Couplings) también obtiene su rastreador el cual permite almacenar los datos de los dos puertos (Port) que conecta (siempre conecta un puerto de salida con uno de entrada). Por último, también se crean rastreadores para los distintos mensajes (Event) que se envían a lo largo de la simulación. En este caso, los trackers permiten conocer los puertos origen y destino del mensaje. Además, almacenan información que identifica si el mensaje fue aceptado o rechazado de acuerdo con la política de ruteo definida en el modelo. El hecho de que dentro de un modelo de red pueda haber más de un modelo de ruteo implica hacer nuevas clasificaciones dentro de los componentes ya mencionados, por un lado, a los rastreadores de puertos de entrada/salida se le agregan los puertos de entrada/salida “Externos” que se asocian a los puertos de entrada (External Input Port) y salida (External Output Port) de un modelo de red. Por otro

lado, también se dividen los acoplamientos en dos grupos: internos y externos. Los acoplamientos internos (Internal Couplings) son aquellos vínculos internos al modelo de red que se dan entre componentes. Es decir, se utilizan para conectar modelos de ruteo. A su vez, los acoplamientos externos (External Couplings) conectan los modelos de red con sus modelos de ruteo componentes. Esto es, vinculan un puerto de entrada/salida externa de un modelo de red con un puerto de salida/entrada interna de un modelo de ruteo.

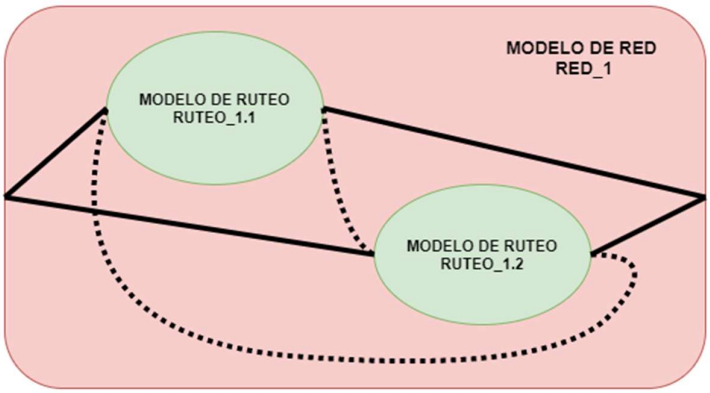
3.2 Captura de Datos

La librería mencionada en el apartado 2.1 permite definir y ejecutar modelos utilizando el formalismo RDEVS por medio de la implementación de las clases y funcionalidades requeridas como extensión de DEVSJAVA. La herramienta presentada en este proyecto se encuentra embebida en la librería RDEVS.

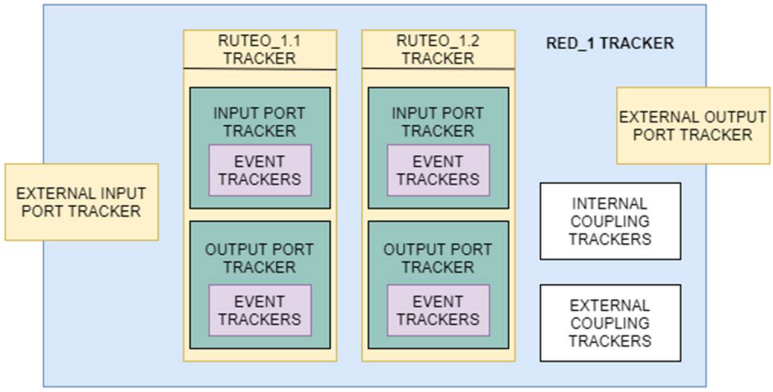
Al inicio de cada simulación se crean los trackers asociados a las clases que representan los componentes del modelo RDEVS. Para el modelo de la Figura 4.a, la Figura 4.b muestra cómo se da el proceso de creación de trackers.

Parte de los datos que se almacena son atributos denominados estáticos ya que se asignan al inicio de la ejecución (es decir, con la creación de los rastreadores y no varían a lo largo de la misma); por ejemplo: id, nombre, puertos asociados a un modelo, acoples asociados a un puerto, modelo esencial de un modelo de ruteo, etc. Por otro lado, la información dinámica es aquella que se obtiene a lo largo de la simulación. Para esta herramienta los datos que se generan durante la simulación son aquellos asociados con el envío de mensajes entre modelos de ruteo. De cada evento creado se debe registrar el tipo de evento, los puertos de origen/destino, y el evento concreto asociado en caso de ser aceptado. Para conocer todos los eventos que entran y salen por un modelo es necesario acceder a sus puertos los cuales poseen una lista de mensajes que llegan/salen por el mismo. Por lo tanto, para obtener los datos dinámicos de la simulación se almacenan los rastreadores de los eventos en una lista dentro de los rastreadores de los puertos, esto se puede observar en la Figura 4.b.

Una vez finalizada la simulación, es necesario generar un reporte para registrar toda la información recolectada de manera estructurada. Se optó por crear un archivo con formato de tipo JSON para crear este reporte. Para este proceso se utiliza la librería Google Gson [9] que es una biblioteca de código abierto para el lenguaje Java que facilita la conversión entre objetos Java y JSON. Se utiliza este tipo de formato ya que se considera fácil de entender y hace más fácil la comunicación de datos, los cuales se utilizan en módulo de visualización que se describe en el siguiente apartado.



(a)



(b)

Fig. 4. (a) Modelo utilizando el formalismo RDEVS. (b) Creación de trackers asociados al modelo.

3.3 Visualización de información

En base a los registros resultantes del proceso de simulación, los datos son estructurados en nuevos modelos de datos según el tipo de gráfico a visualizar. Los gráficos son generados haciendo uso de la herramienta GWT [4]. Este framework de código abierto creado por Google funciona como plug-in de Eclipse [3] y nos permite utilizar una gran variedad de gráficos en HTML y JAVASCRIPT directamente desde código Java.

Este módulo tiene como objetivo una nueva representación de la información recolectada que permita, según el grafico implementado, entender distintos aspectos del proceso de simulación.

Actualmente se está trabajando con diagramas Sankey [10]. El diagrama de Sankey es un tipo específico de diagrama de flujo en el cual el ancho de las flechas se visualiza de forma proporcional a la cantidad de flujo entre dos nodos específicos. Bajo esta premisa, siendo cada nodo del diagrama un modelo de ruteo, se utiliza el ancho de las flechas para representar el intercambio de eventos entre modelos, diferenciando eventos internos/externos, aceptados/rechazados y contenido del mensaje, entre otros.

En la Figura 5 se muestra como ejemplo un gráfico Sankey que representa el flujo de eventos a lo largo de la simulación del modelo definido para el ejemplo de la Figura 1. Cada color se asocia según los caminos posibles (Rojo representa a todos los paquetes que deben ser reparados, Azul indica que el paquete seleccionado se envía directamente al embalaje). Para este caso, la maquina se selección número uno envía 20 paquetes correctos directamente a embalar, mientras que 10 paquetes seleccionados resultan ser defectuosos y se envían a reparación.

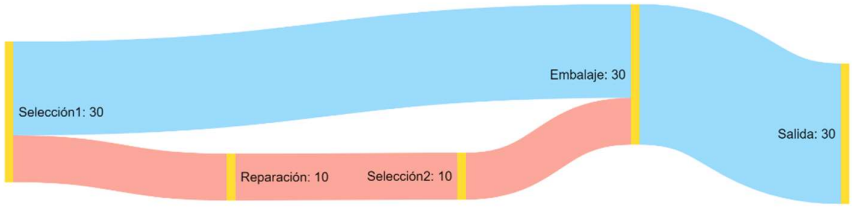


Fig. 5. Diagrama Sankey para el flujo de eventos correspondiente al ejemplo de la Figura 1.

4 Conclusión y Trabajos Futuros

En este trabajo se ha presentado una herramienta desarrollada en Java que tiene como objetivo mejorar la comprensión de la simulación de modelos RDEVS. Esto se logra implementando rastreadores asociados a los distintos componentes del formalismo que almacenan datos referidos al intercambio de eventos. Muchas herramientas de simulación se concentran solo en los resultados finales, por lo que este trabajo tiene como fin conocer el comportamiento de las entidades del formalismo RDEVS a lo largo del proceso de simulación. Además, permite abstraer todos esos datos e información recolectada de manera sencilla para su posterior manipulación.

Actualmente el reporte gráfico de este trabajo se genera de manera estática, esto refiere a que la herramienta GWT genera los gráficos con los datos finales una vez terminada la simulación. Se propone como opción de mejora a futuro de la herramienta encontrar una manera de generar los gráficos de manera dinámica, es decir, lograr que la información se genere y se ilustre en tiempo real.

Referencias

1. Zeigler, B. P., Muzy, A., Kofman, E. Theory of modeling and simulation: discrete event & iterative system computational foundations. Academic press (2018).
2. Blas, M. J., Gonnet, S., Leone, H. Routing structure over discrete event system specification: A DEVS adaptation to develop smart routing in simulation models. In Proceedings of the 2017 Winter Simulation Conference, 774-785 (2017).
3. The Eclipse Foundation. Eclipse URL: https://www.eclipse. org/.
4. Google Web Toolkit. GWT Project. http://www.gwtproject. org/.
5. Blas, M. J., Dalmasso, F., Toniolo, M., Gonnet, S. Diseño e Implementación de una Herramienta de Software para el Modelado y Simulación en RDEVS (WIP). Anales del 8º Congreso Nacional de Ingeniería Informática / Sistemas de Información (2020).
6. Blas, M. J., Un Modelo de Simulación para el Análisis de Arquitecturas de Software de Aplicaciones Web y en la Nube. Tesis Doctoral UTN-FRSF (2019).
7. Arizona Center for Integrative Modeling and Simulation (ACIMS). DEVSJAVA. http://acims.asu.edu/software/ devsjava/.
8. Arizona Center for Integrative Modeling and Simulation (ACIMS). DEVS-Suite. http://acims.asu.edu/software/ devs-suite/.
9. Google Gson Library. https://github.com/google/gson
10. Sankey. https://es.wikipedia.org/wiki/Diagrama_de_Sankey