OpenSCOR: Framework para Análisis de Performance en Simulaciones de Cadenas de Suministro

Juan L. Sarli[1], Horacio Leone[1], Ma. De los Milagros Gutierrez[2]

[1]INGAR, Instituto de Diseño y Desarrollo CONICET – UTN, Santa Fe, Argentina
{juanleonardosarli, hleone}@santafe-conicet.gob.ar
[2] CIDISI, Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información UTN, Santa Fe, Argentina
mmgutier@frsf.utn.edu.ar

Abstract. Una cadena de suministro es una red de organizaciones que implica relaciones de colaboración entre sus participantes a largo plazo. Para lograr su éxito es necesario coordinar de manera eficiente diversos flujos de materiales, información y dinero. Así el modelo SCOR surge como una herramienta de diseño de cadenas de suministro que realiza una descripción coloquial de las métricas y procesos relevantes de ella. Sin embargo, la simulación emerge como una herramienta adecuada para encontrar el comportamiento más conveniente. Por ello, es necesario determinar qué factores se deben relevar en una simulación para analizar el rendimiento de la misma. Este trabajo presenta un framework que toma como base el modelo SCOR e implementa sus métricas y procesos en componentes de simulación. El objetivo de esta propuesta es brindar un marco de trabajo para la evaluación de la performance en ambientes de simulación de cadenas de suministro.

Keywords: Framework, evaluación cadena de suministro, modelo SCOR, formalismo DEVS, simulación cadena de suministro.

1 Introducción

En los tiempos actuales de la globalización, el constante cambio, los entornos cada vez más competitivos donde la necesidad de información es inmediata, donde existen grandes volúmenes de datos y es necesario determinar qué información es relevante, resulta evidente que una sola organización por sí misma es incapaz de gestionar esta dinámica cambiante. Para adecuarse a esta nueva situación y permanecer competitivas en los mercados es necesario colaborar entre distintas organizaciones. Debido a este hecho, una de las formas de organización más frecuentes es la cadena de suministro (CS) [1–3].

Una CS es una red de organizaciones que posee relaciones de colaboración a largo plazo entre sus miembros, en los procesos y actividades, para producir productos o servicios de valor para los clientes [4, 5]. Lograr el éxito de una CS implica coordinar de una forma eficiente los distintos flujos de materiales, información, dinero entre las organizaciones que participan de la red [6]. En este contexto donde la eficiencia es un

factor clave para el éxito de la CS, la simulación emerge como una herramienta adecuada para encontrar cual es el comportamiento más adecuado para alcanzar dicho factor [7, 8].

Simular una CS permite a las organizaciones miembros analizar y experimentar distintas configuraciones de colaboración en un entorno virtual, lo que reduce en una forma considerable el tiempo y los costos asociados con las pruebas físicas sobre la configuración existente. Además, llevar a cabo una simulación proporciona un modo de prueba para un sistema en un entorno controlado, libre de riesgos, cuyo tamaño, disposición y consideraciones operacionales se validan por medio de un análisis estadístico de los eventos ocurridos. También, es posible entender los factores o componentes que afectan a la CS, evaluar problemas desconocidos, analizar decisiones, realizar gestiones de riesgo y efectuar planificaciones futuras [9–11].

Es claro que la simulación de una CS otorga beneficios importantes pero es necesario definir que relevar en la misma. De este modo, es posible realizar un análisis de los factores que impactan sobre el rendimiento de la CS. Determinar los aspectos relevantes no es una cuestión menor, por lo que diversas organizaciones se han congregado en el Supply Chain Council y han desarrollado el modelo Supply Chain Operation Reference (SCOR). Este modelo tiene como objetivo proveer una terminología común entre CS de diferentes ámbitos; facilitar la medición y evaluación de la performance tanto de forma integral como en los procesos individuales; analizar y fijar objetivos de rendimiento como así también determinar oportunidades de mejora. Por lo tanto, mediante el uso del modelo SCOR es posible especificar qué factores se deben evaluar en una simulación de CS y comprender el comportamiento dinámico de las mismas [12].

Actualmente se encuentran simuladores comerciales o educativos que poseen facilidades para diseñar simulaciones y realizar un análisis estadístico de los resultados, entre los cuales se destacan: Simio [13], AnyLogic [14], FlexSim [15], Implexa [16]. Todos ellos brindan distintas facilidades para simular por medio de componentes visuales 3D a los que se les añade el comportamiento deseado. Además, presentan componentes para realizar simulaciones con agentes, con eventos discretos o para simulaciones continuas. Si bien las funcionalidades previstas por estos simuladores son muy potentes, ninguno de ellos presenta herramientas para evaluar los indicadores de rendimiento propuestos en el modelo SCOR. En cualquiera de estos simuladores, la alternativa es diseñar e implementar nuevos componentes que releven los factores de rendimiento importantes. Cabe destacar que FlexSim posee un conjunto de métricas personalizadas basadas en tiempos y costos que permiten evaluar la CS pero ninguno de las mismas se basa en el modelo SCOR.

En este contexto, resulta evidente que es de utilidad contar con componentes de simulación para evaluar el rendimiento de la CS basado en métricas ampliamente aceptadas por la comunidad de modelado, de practicantes y empresarial. En este sentido, este trabajo propone un framework denominado *OpenSCOR* que implementa un conjunto de métricas y procesos definidos en el modelo SCOR a través de modelos de simulación DEVS, para la evaluación de la performance de CS. Esta propuesta es una solución de software parcial que permite añadir métricas y procesos SCOR particulares en base a los componentes de simulación de una CS definidos en el framework.

Para el desarrollo de los componentes del framework se utiliza el formalismo Discrete EVents System specification (DEVS) [17].

El resto del trabajo se organiza de la siguiente manera. Primero se describen los conceptos de modelo SCOR y formalismo DEVS. Luego se presenta la arquitectura definida para el desarrollo del framework y una explicación de cada uno de sus componentes. Finalmente el trabajo se concluye.

2 Conceptos preliminares

2.1 Modelo SCOR

Este modelo fue desarrollado por el Supply Chain Council y actualmente se encuentra en su versión once. Es un modelo conceptual que exhibe los conceptos más relevantes de una CS y provee una terminología común para analizar el rendimiento de las CS. En la Figura 1, se aprecia que SCOR se encuentra organizado alrededor de seis grandes procesos: Plan, Source, Make, Deliver, Return y Enable. Además se visualiza que los procesos del modelo son útiles tanto para gestión inter como intra organizacional; debido a que se alienta desde su organización fundadora el uso del modelo desde los proveedores de los proveedores de una organización hasta los clientes de los clientes de la misma.

Fig. 1. Procesos Nivel Uno de SCOR.

Cada uno de estos procesos se divide en tres niveles jerárquicos los cuales son: Alcance, Configuración y Pasos respectivamente. Para evaluar los procesos se definen métricas de rendimiento las que respetan una estructura jerárquica con tres niveles al igual que los procesos. Las relaciones entre estos niveles de métricas son diagnósticas, es decir que la medición de una métrica de nivel i+1 permite ahondar en los factores que determinan la evaluación de una métrica de nivel i. Este tipo de análisis se conoce como descomposición de métricas.

Según el modelo SCOR, para delimitar cuál es la dirección estratégica de la CS, se definen cinco atributos de performance: Confiabilidad (Reliability), Capacidad de Respuesta (Responsiveness), Agilidad (Agility), Costos (Cost) y Gestión de Activos (Asset Management). Los primeros tres se encuentran enfocados en el cliente o la

visión externa, mientras que los últimos dos se centran en una visión interna de la CS. Una vez definida la dirección estratégica, es posible seleccionar que métricas se desean para evaluar el rendimiento. En la Figura 2 se visualizan las métricas de nivel 1 asociadas con cada atributo de performance.

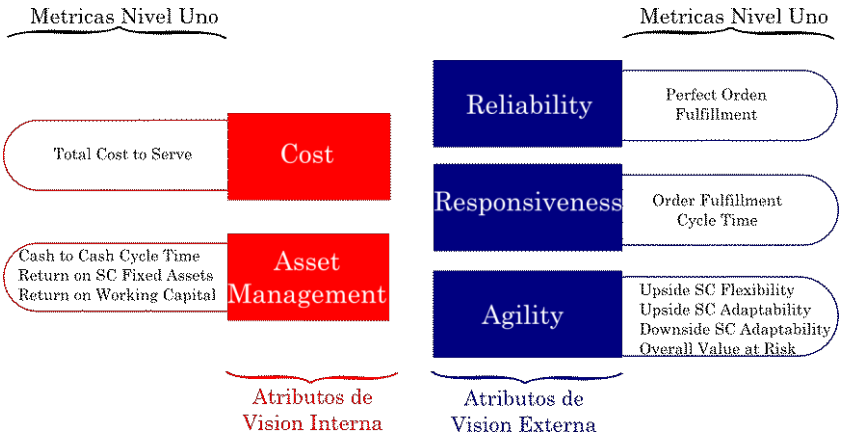

Fig. 2. Atributos de Performance y Métricas.

2.2 Formalismo DEVS

El formalismo DEVS, desarrollado por Bernard Zeigler a mediados de los 70, permite representar cualquier sistema que tenga un número finito de cambios en un intervalo finito de tiempo. Surge, en parte, debido a las limitaciones que poseen los formalismos gráficos como las Redes de Petri y los Grafos de Transición de Estados para modelar y simular sistemas de eventos discretos de propósito general [17, 18].

Un modelo DEVS procesa una secuencia de eventos de entrada y de acuerdo a su condición inicial produce una secuencia de eventos de salida. Por lo tanto, un modelo DEVS atómico con puertos se define por la estructura:

$$M = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta) \quad (1)$$

Donde:

X = {(p, v)} p ϵ conjunto de puertos de entrada y v ϵ conjunto de valores de entrada.

Y = {(p, v)} p ϵ conjunto de puertos de salida y v ϵ conjunto de valores de salida.

S = Es el conjunto de estados.

$\boldsymbol{\delta_{int}}$, $\boldsymbol{\delta_{ext}}$, **λ** y **ta** = Son las funciones que definen la dinámica del sistema.

En la práctica modelar sistemas de eventos discretos con modelos atómicos resulta impracticable, por tal motivo el formalismo define la idea de acoplamiento, es decir, modelar a los sistemas como una composición de subsistemas más simples. Entonces, un modelo DEVS acoplado con puertos se define por la estructura:

$$N = (X, Y, D, \{M_d | d \epsilon D\}, EIC, EOC, IC) \quad (2)$$

Donde:

X e **Y** = Representan los mismo conjuntos que los definidos en (1).

D = Conjunto de identificadores de componentes.

Requerimiento de D: $\forall d \in D$, $M_d = (X_d, Y_d, S, \delta_{int}, \delta_{ext}, \lambda, ta)$ es un modelo DEVS, atómico/acoplado con puertos.

EIC (External Input Coupling) = Es el conjunto de conexiones desde las entradas de N hacia las entradas de los M_d.

EOC (External Output Coupling) = Es el conjunto de conexiones desde las salidas de los M_d hacia las salidas de N.

IC (Internal Coupling) = Es el conjunto de conexiones desde las salidas de los M_d hacia las entradas de los M_d. No se permite feedback directo.

3 Desarrollo del Framework

El creciente avance de la tecnología basada en componentes en el desarrollo de software ha permitido la construcción de entornos computacionales flexibles, donde sus principales ventajas son el reuso y la construcción de modelos al estilo de rompecabezas. La aplicación de esta tecnología a la construcción de modelos de simulación, provee la capacidad de reusar componentes y de esta manera agilizar la construcción del mismo. El formalismo DEVS propone la construcción de modelos de simulación a partir de componentes, conocidos también como bloques de construcción, es por esta razón que se utiliza este formalismo en la definición de los componentes que van a formar parte de los modelos de simulación. El principal desafío en este trabajo, es la definición correcta de los componentes, en este sentido, es necesario identificar los conceptos a ser modelados y en función a esto, identificar el comportamiento dinámico de los mismos. Dado que el principal objetivo de este trabajo es brindar una herramienta flexible que permita construir y ejecutar modelos de simulación de CS, se seleccionó el modelo SCOR, como fuente de conocimiento para la identificación de conceptos, y el formalismo DEVS para la implementación del concepto como modelo dinámico. Finalmente el framework que se presenta, permite construir modelos de simulación utilizando dichos componentes.

Para la construcción del framework se utiliza la herramienta PowerDEVS [19], sobre la que se implementaron los modelos DEVS atómicos y acoplados que representan los conceptos provenientes de SCOR. Luego, el usuario podrá combinar estos modelos y acoplarlos para producir una CS particular.

3.1 Definición de la Arquitectura

En la Figura 3 se visualiza la arquitectura del framework compuesta de dos capas. La capa inferior corresponde a PowerDEVS, la cual implementa los componentes del formalismo DEVS. En esta capa se encuentran definidos modelos atómicos y acoplados, funciones de transición internas y externas, puertos de entrada/salida y las máquinas de simulación correspondientes a los simuladores atómicos y acoplados como también el simulador general que implementa el ciclo de simulación DEVS. En

la capa superior se definen los componentes de OpenSCOR en los que se encuentran los procesos y métricas del modelo SCOR definidos como bloques de construcción, los que serán utilizados en los modelos de simulación de CS definidos por el usuario.

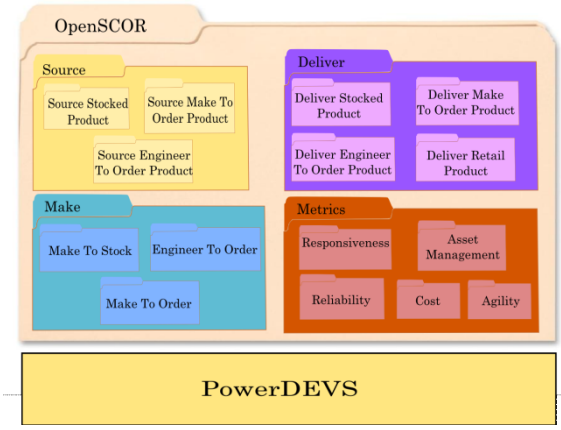

Fig. 3. Arquitectura del Framework.

3.2 Componentes de OpenSCOR

En esta sección se describen los componentes del framework. Cada uno de los paquetes lleva el nombre de los procesos de primer nivel de SCOR y contiene los bloques de construcción asociados al proceso o métrica del modelo SCOR que representan. De este modo, se busca una forma sencilla y fácil de identificar los componentes para diseñar un marco de evaluación a la simulación de una CS. Además, los componentes que representan procesos SCOR tales como *Make to order*, *Make to stock*, entre otros, se han diseñado como modelos DEVS acoplados con el objetivo de que sus componentes sean modelos de simulación particulares que se identifiquen con la CS concreta a evaluar. Para definir los componentes se determinaron puertos de entrada/salida y valores en estos puertos. Para el caso de componentes atómicos se identificaron también funciones de transición interna y externa y función de salida.

A continuación se describen cada uno de los componentes de esta capa, en *Make To Stock* se presenta una explicación de cómo hacer uso del mismo y un diagrama para describir el comportamiento de la actividad *Produce*. Los demás componentes se implementan de la misma forma que *Make To Stock*.

Source. Este paquete contiene los procesos asociados a las actividades de ordenar, entregar, recibir y transferir las materias primas, productos y/o servicios. Este proceso puede utilizar una estrategia para stock, por orden o por proyecto de ingeniería. Dentro de este paquete se encuentran definidos los siguientes componentes:

Source Stocked Product. Este componente se encuentra diseñado para mantener un determinado nivel de inventario de las materias primas o productos. No existe ninguna referencia al cliente que se intercambie con el proveedor, sino que se solicita una cierta cantidad de un tipo de producto. Aquí los productos son estándar, es decir, no poseen parámetros configurables por lo que las órdenes solo varían en la cantidad.

Source Make To Order Product. Se ordenan y reciben productos o materiales solo cuando es requerido por una orden. Este componente mantiene el inventario ordenado. Los productos son ordenados, recibidos e identificados en inventario mediante el número de referencia de la orden. Las órdenes poseen variaciones en la configuración de ciertos parámetros que los define cada uno de los clientes específicos.

Source Engineer To Order Product. Para este tipo de abastecimiento, el proceso se encarga de identificar y seleccionar proveedores, negociar, validar, planificar, ordenar y recibir partes, diseñar productos especializados basados en los requerimientos de una orden especifica. Cada orden es especial y distinta, no existen dos órdenes iguales y cada una se trata como si fuera la primera vez que se realiza.

Make. Este paquete contiene los procesos para agregar valor a los productos a través de mezclar, separar, utilizar máquinas y procesos químicos. Como en el caso de Source, es posible utilizar tres estrategias: para inventario, por orden o por proyecto de ingeniería. Los componentes que forman parte de este paquete son:

Make To Stock (MTS). Este es un proceso de manufactura que agrega valor a los productos por medio de distintas técnicas. Los productos que se envían son bienes finales o estándares que se producen antes de la recepción de órdenes de clientes y en concordancia con los pronósticos de ventas. No existe referencia a los clientes u órdenes.

En la Figura 4 se puede observar cómo se comunican los subprocesos de este componente mientras que en la Tabla 1 se presenta su especificación formal. Este proceso se compone de siete actividades que se representan con modelos acoplados compuestos de actividades simples que modelan las tareas de fabricación para stock de una CS. Las entradas al modelo se presentan con flechas en color rojo, mientras que las salidas se visualizan con flechas en color verde.

Este tipo de proceso de fabricación es de uso común en las fábricas de yerba mate, gaseosas, electrodomésticos básicos (tostadoras, licuadoras), donde existen tipos de productos pero cada uno de esos tipos se produce en base a pronósticos de demanda. Las órdenes solo establecen el tipo y cantidad del producto.

Table 1. Especificación Formal Modelo MTS.

Elemento	Estructura
MTS	$\{X_{MTS}, Y_{MTS}, D_{MTS}, \{M_{MTS,d} \mid d \in D_{MTS}\}, EIC_{MTS}, EOC_{MTS}, IC_{MTS}\}$

X_{MTS}	{(p,v) | p ϵ MTSIP, v ϵ R_0^+-} donde MTSIP = {Production, Receipts, Inventory, Delivery}
Y_{MTS}	{(p,v) | p ϵ MTSOP, v ϵ R_0^+ } donde MTSOP = {Production, Inventory, Replenishment, Products}
D_{MTS}	{Schedule, Issue, Produce, Package, Stage, Release, Waster}
EIC_{MTS}	{ ((MTS,Production), (Schedule,Production)), ((MTS,Production), (Stage,Production)), ((MTS,Receipts), (Schedule,Receipts)), ((MTS,Inventory), (Issue,Inventory)), ((MTS,Delivery), (Stage,Delivery)) }
EOC_{MTS}	{ ((Schedule,Production), (MTS,Production)), ((Issue,Inventory), (MTS,Inventory)), ((Issue,Replenishment), (MTS,Replenishment)), ((Release,Products), (MTS,Product)) }
IC_{MTS}	{ ((Schedule,Production), (Issue,Production)), ((Issue,End), (Produce,Start)), ((Produce,End), (Package,Start)), ((Produce,Waste),(Waster,Waste)), ((Package,End), (Stage,Start)), ((Stage, End), (Release, Start)) }

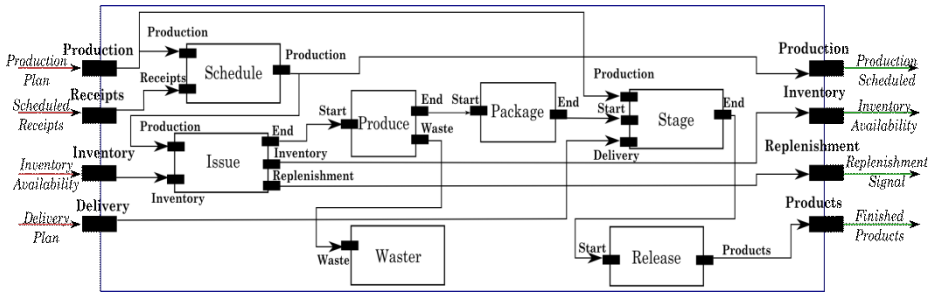

Fig. 4. Componentes del modelo MTS.

En la Figura 5 se presenta el modelo de la actividad *Produce*. Se exhiben cuatro tareas: *Manufacture*, *Quality Control*, *Waste Collector* y *Produce and Test Cycle Time*. La primera se encarga de fabricar la cantidad de producto solicitada con los materiales recibidos. Su especificación formal se presenta en la Tabla 2. La segunda efectúa el control de calidad sobre los productos fabricados. La responsabilidad de la tercera es recolectar los residuos de fabricación y los productos defectuosos del control de calidad. La última actividad realiza el cálculo del tiempo estimado de producción y control de calidad, es decir, permite efectuar el cálculo de la métrica cuyo nombre es el mismo que el de la actividad y pertenece al atributo *Responsiveness*.

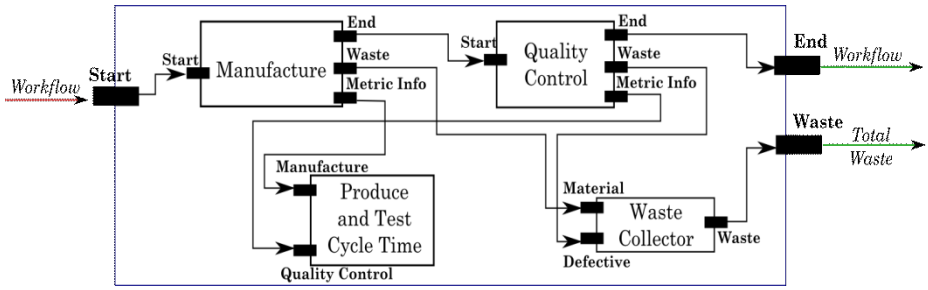

Fig. 5. Componentes del Modelo Produce.

Table 2. Especificación Formal Modelo M (tarea Manufacture).

Elemento	Estructura
M	$\{X_M, Y_M, S_M, \delta_{int,M}, \delta_{ext,M}, \lambda_M, ta_M\}$
X_M	$\{(p,v) \mid p \in MIP, v \in R_0^+\}$ donde MIP = {Start} y $X_{M,p} = R_0^+ \ \forall p$ / $p \in MIP$
Y_M	$\{(p,v) \mid p \in MOP, v \in R_0^+\}$ donde MOP = {End, Waste, Metric Info}
S_M	phase x σ x q^{TP} x pt^{CT} donde phase ∈ {waiting, manufacturing, removingWaste, sendingMetric}, $\sigma \in R_0^+$, $q^{TP} \in R_0^+$, $pt^{CT} \in R_0^+$ donde q^{TP} es la cantidad a producir, pt^{CT} es el tiempo efectivo de realizar la tarea.
$\delta_{int,M}$ (S)	$\delta_{int,M}$(manufacturing, σ, q^{TP}, pt^{CT}) = (removingWaste, rwTime, q^{TP}, pt^{CT} + rwTime) donde rwTime es el tiempo de eliminación de los residuos. $\delta_{int,M}$(removingWaste, σ, q^{TP}, pt^{CT}) = (sendingMetric, informTime, q^{TP}, pt^{CT} + informTime) donde informTime es el tiempo para informar pt^{CT} $\delta_{int,M}$(sendingMetric, σ, q^{TP}, pt^{CT}) = (waiting, ∞, q^{TP}, pt^{CT})
$\delta_{ext,M}$ (s,e,(port,x))	$\delta_{ext,M}$ ((waiting, σ, q^{TP}, pt^{CT}),e,(Start, x_1)) = (manufacturing, t_N, x_1, t_N) donde t_N = getRandomTime(q^{TP}) $\delta_{ext,M}$ ((phase, σ, q^{TP}, pt^{CT}),e,(Start, x_1)) = (phase, σ - e, q^{TP}, pt^{CT}) en cualquier otro caso
λ_M(S)	λ_M(manufacturing, σ, q^{TP}, pt^{CT}) = (End, q^{TP}) λ_M(removingWaste, σ, q^{TP}, pt^{CT}) = (Waste, wasteMaterial) donde wasteMaterial = δ * q^{TP} y δ = porcentaje de desperdicio λ_M(sendingMetric, σ, q^{TP}, pt^{CT}) = (Metric Info, pt^{CT})
ta_M(S)	ta_M(S) = σ

Make To Order. Este tipo de fabricación busca completar, construir o configurar en respuesta a las órdenes de pedidos. La orden de producción conlleva información del cliente y una vez terminado el producto se transfiere al proceso Deliver.

Engineer To Order. Este componente posee los procesos para desarrollar, diseñar, validar y usar un proceso de manufactura para producir productos o servicios basados en los requerimientos de un cliente específico. En general, esta estrategia necesita que las instrucciones de trabajo se re/definan y que el abastecimiento se añada.

Deliver. En este componente se definen los procesos asociados con realizar la gestión de los pedidos de cara al cliente y las actividades de cumplimiento de orden. Este proceso puede hacer uso de las estrategias: por inventario, por orden, por proyecto de ingeniería y al por menor. Cada una se explica a continuación

Deliver Stocked Product. Este proceso entrega productos fabricados en base a una demanda agregada de las órdenes, las proyecciones de demanda y los parámetros de punto de pedido. Además, el objetivo de este proceso es tener disponibilidad del producto cuando arriben los pedidos y, de este modo, prevenir la perdida de la venta. Para las industrias de servicios, este proceso es predefinido y estandarizado.

Deliver Make To Order Product. Este componente posee los procesos asociados a entregar productos configurados, fabricados, y/o ensamblados desde las materias primas o partes en respuesta a una orden específica. Las referencias a la orden se intercambian con el proceso Source o Make y se adjuntan al producto.

Deliver Engineer To Order Product. Este tipo de entrega tiene los procesos para obtener, responder y reservar los recursos para una orden de requerimientos únicos. Además se entrega productos que están parcial o totalmente re/diseñados, fabricados a partir de una lista de materiales. El diseño comienza solo después de validar la orden.

Deliver Retail Product. En este componente se encuentran los procesos usados para adquirir, promocionar y vender bienes finales a los almacenes minoristas. Estos procesos pueden ser manuales o automáticos y se utilizan para recolectar el pago. Esta estrategia trata de mantener los niveles de inventario en los almacenes minoristas.

Metrics. En este paquete se agrupan las métricas según el atributo de performance al que pertenecen. De este modo, se definen los subcomponentes: Reliability, Responsiveness, Asset Management, Cost y Agility.

Reliability. Esta biblioteca contiene una métrica denominada *Perfect Order Fulfillment*. La misma determina el porcentaje de órdenes perfectas, es decir, aquellas que cumplen con una entrega completa, una documentación precisa, sin daños en la entrega, con todos los ítems y cantidades solicitadas, y en la fecha estipulada por el cliente.

Responsiveness. En este caso existe una métrica denominada *Order Fulfillment Cycle Time*. Esta métrica estipula el ciclo de tiempo promedio para cumplir con las órdenes. El cálculo de esta métrica se realiza para cada orden, en donde el ciclo comienza con su recepción y termina con el envío de la misma.

Asset Management. Se compone de tres métricas: *Cash to Cash Cycle Time, Return on SC Fixed Assets* y *Return on Working Capital*. La primera calcula el tiempo que una inversión demora en retornar a la organización luego de comprar los insumos de fabricación. La segunda, mide el retorno de una organización sobre su capital invertido en activos de la CS. La tercera, es una medida que evalúa el retorno de la inversión en relación al capital de trabajo.

Cost. Posee una métrica denominada *Total Cost to Serve*. Esta métrica establece cual es el costo de la CS para entregar productos y servicios a los clientes.

Agility. Se compone de cuatro métricas: Upside SC Flexibility, Upside SC Adaptability, Downside SC Adaptability y Overall Value at Risk. La primera calcula los días requeridos para alcanzar un incremento sostenido del 20% en las cantidades entregadas. La segunda, establece el máximo porcentaje de incremento en la cantidad entregada que se puede lograr en 30 días. La tercera, determina que reducción se puede lograr en las cantidades ordenadas 30 días antes de la entrega sin penalidades de inventario o costos. La última, presenta la exposición al riesgo de una cartera de negociación basado en las volatilidades históricas.

Con los componentes de la arquitectura definidos, el usuario que necesite simular una SC deberá seleccionar los procesos correspondientes al caso específico y acoplarlos. Notar que los distintos procesos tienen definidos diferentes puertos, haciendo que no puedan ser conectados de cualquier manera sino que el usuario deberá tener cuidado en cómo realizar esta conexión. Para orientar al usuario, se definieron los mismos nombres de puertos en los modelos que pueden ser conectados.

4 Conclusiones

En este trabajo se ha presentado el framework denominado OpenSCOR que provee una forma fácil e intuitiva de evaluar la performance de una simulación de CS. Como novedad presenta bloques de simulación predefinidos con los procesos y métricas del modelo SCOR. A su vez, los componentes han sido diseñados por medio de modelos DEVS atómicos y acoplados que permiten una composición sencilla entre los procesos de SCOR y los procesos reales de la CS. De esta forma, el marco de trabajo propuesto estimula el uso de los procesos de evaluación del modelo SCOR en la simulación de CS y permite la comparación de las métricas de rendimiento tanto en dentro de la misma industria como entre otras industrias. Como trabajo futuro, se espera incorporar al framework reglas lógicas que asistan al usuario en la correcta implementación de los modelos, de manera de evitar errores en el acoplamiento.

Referencias

1. Mezgár, I., Rauschecker, U.: The Challenge of Networked Enterprises for Cloud Computing Interoperability. Computers in Industry. 65, 657–674 (2014).
2. Friesen, A., Theilmann, W., Heller, M., Lemcke, J., Momm, C.: On Some Challenges in Business Systems Management and Engineering for the Networked Enterprise of the Future. pp. 1–15. Springer Berlin Heidelberg, (2012).
3. Cedillo-Campos, M.G., Martínez-Hernández, A., Villa-Araujo, J.C., Cantu-Sifuentes, M.: Service Supply Chains Performance Improvement Method: The Case of Supplier Selection in the Metallurgical Sector. Nova scientia. 7, 314–338 (2015).
4. Chavez, J.H.: Supply Chain Management (Gestión de la cadena de suministro). RIL Editores (2012).
5. Álvarez, E., Díaz, F., Larrinaga, M.A.: Panorama de la gestión de la cadena de suministro: retos, colaboración y gestión de excepciones. Boletín de Estudios Económicos. 66, 531 (2011).
6. Camarinha-Matos, L.M., Afsarmanesh, H., Galeano, N., Molina, A.: Collaborative Networked Organizations – Concepts and Practice in Manufacturing Enterprises. Computers & Industrial Engineering. 57, 46–60 (2009).
7. Chan, F.T.S., Zhang, T.: The Impact of Collaborative Transportation Management on Supply Chain Performance: A Simulation Approach. Expert Systems with Applications. 38, 2319–2329 (2011).
8. Chu, Y., You, F.: Simulation-based optimization for multi-echelon inventory systems under uncertainty. In: Proceedings of the 2014 Winter Simulation Conference. pp. 385–394. IEEE Press (2014).
9. Moon, J.-H., Lee, Y.-H., Cho, D.-W.: Object-oriented Simulation Modeling for Service Supply Chain. Journal of the Korea Society for Simulation. 21, 55–68 (2012).
10. Andres, B., Sanchis, R., Poler, R.: Modelado y simulación de la cadena de suministro con AnyLogic®. Modelling in Science Education and Learning. 9, 57–72 (2016).
11. Ingalls, R.G.: Introduction to supply chain simulation. In: Proceedings of the 2014 Winter Simulation Conference. pp. 36–50. IEEE Press (2014).
12. SCOR. The SCC (2012).
13. Simio, accessed April 29. http://www.simio.com/index.php.
14. Multimethod Simulation Software and Solutions, accessed April 29. http://www.anylogic.com/.
15. Flexsim Simulation Software, accessed April 28. https://www.flexsim.com/es/.
16. Implexa, accessed April 28. http://www.implexa.net/.
17. Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press (2000).
18. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Springer Science & Business Media (2009).
19. Bergero, F., Kofman, E.: PowerDEVS: a tool for hybrid system modeling and real-time simulation. SIMULATION. 87, 113–132 (2011).