Juan Leonardo Sarli

Modelo de Interoperabilidad para Simulación Distribuida de Cadenas de Suministro

Tesis Doctoral

UTN ✳ SANTA FE

CiN REUN
Red de Editoriales de Universidades Nacionales de la Argentina

L U A
Libro Universitario Argentino

edUTecNe

UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL SANTA FE

DOCTORADO EN INGENIERÍA

MENCIÓN EN SISTEMAS DE INFORMACIÓN

Tesis Doctoral

MODELO DE INTEROPERABILIDAD PARA SIMULACIÓN DISTRIBUIDA DE CADENAS DE SUMINISTRO

Ing. Juan Leonardo Sarli

Directora: Dra. María de los Milagros Gutierrez

Co Director: Dr. Horacio Leone

Santa Fe, Argentina Marzo 2019

Sarli, Juan Leonardo

Modelo de interoperabilidad para simulación distribuida de cadenas de suministro / Juan Leonardo Sarli. - 1a ed . - Ciudad Autónoma de Buenos Aires : edUTecNe, 2019.

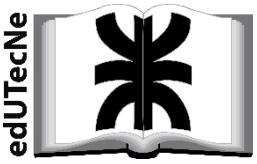
Libro digital, PDF

Archivo Digital: descarga y online

ISBN 978-987-4998-34-7

Universidad Tecnológica Nacional – República Argentina
Rector: Ing. Hector Eduardo **Aiassa**
Vicerrector: Ing. Haroldo **Avetta**
Secretaria Académica: Ing. Liliana Raquel **Cuenca Pletsch**
Secretario de Ciencia, Tecnología y Posgrado: Dr. Horacio Pascual **Leone**

Universidad Tecnológica Nacional – Facultad Regional Santa Fe
Decano: Ing. Rudy Omar **Grether**
Vicedecano: Ing. Eduardo José **Donnet**



edUTecNe – Editorial de la Universidad Tecnológica Nacional
Coordinador General a cargo: Fernando H. **Cejas**
Área de edición y publicación en papel: Carlos **Busqued**
Colección Energías Renovables, Uso Racional de Energía, Ambiente: Dr. Jaime Moragues.

Queda hecho el depósito que marca la Ley Nº 11.723

Sarmiento 440, Piso 6 (C1041AAJ) Buenos Aires, República Argentina
Publicado Argentina – Published in Argentina

ISBN 978-987-4998-34-7

UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL SANTA FE

COMISION DE POSGRADO

Se presenta esta tesis en cumplimiento de los requisitos exigidos por la Universidad Tecnológica Nacional para la obtención del grado académico de Doctor en Ingeniería, mención Sistemas de Información

MODELO DE INTEROPERABILIDAD PARA SIMULACIÓN DISTRIBUIDA DE CADENAS DE SUMINISTRO

por

Ing. Juan Leonardo Sarli

Directora: Dra. María de los Milagros Gutierrez

Co Director: Dr. Horacio Leone

Jurados

Dra. María de los Ángeles Martín

Dr. Germán Montejano

Dr. Carlos María Chezzi

Santa Fe, Argentina Marzo 2019

A Rubén y María Teresa

A Pablo, Maru y Claudia

A Julia

Agradecimientos

A mi codirector de tesis Horacio Leone, quien siempre aporto su consejo y experiencia en los momentos cruciales de este largo camino.

A mi directora de tesis Milagros Gutiérrez, quien hizo innumerables aportes al momento de comenzar a investigar la temática de esta tesis.

A mis compañeros de INGAR, Julia, Gonzalo y Álvaro quienes han colaborado en el trabajo de investigación, compartido sus experiencias, brindado su afecto y apoyo en el ámbito de trabajo.

A todos mis compañeros del comedor diario de INGAR, sin los cuales el día a día hubiera sido mucho más difícil y gracias a los cuales he aprendido mucho tanto personal como laboralmente.

A mi familia y amigos, quienes me alentaron a perseverar a pesar de las dificultades y colaboraron en todo lo que he necesitado.

Por último, al CONICET por haber brindado un soporte económico a través de una beca de doctorado el cual permitió la realización de esta tesis.

Prefacio

La gestión colaborativa de cadena de suministro (CS) con participantes distribuidos geográficamente, sin un participante predominante que coordine la colaboración, con una dinámica cambiante y compleja requiere herramientas sofisticadas para su gestión. El modelado y simulación se ha utilizado extensamente para el estudio de sistemas complejos, constituyéndose en una herramienta fundamental para apoyar el proceso de toma de decisión en la gestión colaborativa de CS. Además, la simulación de cadena de suministro asiste a quienes toman decisiones en el análisis de múltiples escenarios, la selección de soluciones apropiadas, la comprensión de interacciones y la mejora en los indicadores de rendimiento de la CS. A medida que la cadena de suministro se vuelve más grande y compleja, sus respectivos modelos de simulación crecen de forma directamente proporcional a su tamaño tornando, de este modo, muy difícil o imposible su simulación de forma tradicional (una única persona que construye un único modelo de simulación y ejecuta experimentos en una única computadora). Este problema se puede abordar haciendo uso de simulación distribuida (SD) la cual permite construir una simulación a partir de la integración de simuladores (cada uno con su modelo de simulación) distribuidos geográficamente, ejecutados en simultáneo y que intercambian eventos. Este enfoque de simulación, posibilita modelar el comportamiento de la CS como sub modelos ejecutados en distintos nodos de una red de computadoras, los cuales se integran para componer el modelo de simulación total de la cadena de suministro.

Uno de los problemas de este enfoque es la interpretación realizada por un simulador de los eventos enviados por otro simulador. De forma tal que estos eventos provoquen los efectos esperados, por quien diseña el modelo de simulación ejecutado en el otro simulador. Este problema se conoce como interoperabilidad entre simuladores. Para alcanzar la interoperabilidad entre simuladores es necesario determinar qué nivel de interoperabilidad se desea lograr. Los niveles de interoperabilidad se obtienen de forma gradual, dado que para alcanzar un cierto nivel de interoperabilidad es necesario alcanzar el nivel anterior. De acuerdo con (Tolk, Bair, y Diallo 2013), es posible determinar tres grandes niveles: 1) *Sintáctico*, 2) *Semántico* y 3) *Pragmático*. El sintáctico implica que los simuladores pueden

comunicarse e intercambiar eventos. El semántico determina si la comunicación e intercambio de eventos es semánticamente válida, lo que permite detectar si la composición de la simulación es significativa y válida. Para alcanzar este nivel de interoperabilidad, es necesario lograr la comprensión y alineación sin ambigüedades del significado de los eventos por parte de los simuladores, y para ello, es frecuente el uso de modelos conceptuales. El pragmático implica que todos los simuladores de la simulación distribuida utilizan los datos de la misma manera.

High Level Architecture (HLA) es un estándar ampliamente usado por las soluciones propuestas en SD. Este estándar utiliza el concepto de *Federation Object Model* (FOM), que provee una especificación común para el intercambio de datos entre simuladores en un formato estándar. En síntesis, el FOM establece un contrato del modelo de información que es necesario para alcanzar el nivel de interoperabilidad sintáctico, pero no así el semántico entre simuladores. A partir de los lineamientos propuestos por HLA para construir el FOM, es posible asegurar la interoperabilidad sintáctica entre los simuladores intervinientes. Sin embargo, la interoperabilidad semántica está fuera del alcance de este estándar, y es responsabilidad exclusiva de sus miembros garantizarla.

Llevar a cabo una simulación distribuida basada en HLA de CS, en el contexto de gestión colaborativa de la cadena, no es una tarea sencilla. Entre los inconvenientes más relevantes surgen los siguientes problemas: definir el FOM, establecer una interpretación unívoca del mismo, verificar la completitud y consistencia del FOM, y por último pero no menos importante, poseer en cada organización un conocimiento profundo sobre las tecnologías de HLA, para llevar a cabo la SD de cadena de suministro.

Con el objetivo de proveer un método de construcción para el modelo de objetos de federación (FOM) que sea simple de utilizar, que no requiera conocimientos específicos de HLA, que provea un mecanismo para verificar la completitud y consistencia del modelo de objetos, que reduzca el tiempo, esfuerzo y la intervención humana necesarias para su construcción, este trabajo de tesis presenta un marco conceptual basado en una red de ontologías. Este marco da soporte a las tareas de modelado y composición de un modelo de interoperabilidad, para la simulación distribuida de cadenas de suministro, a fin de garantizar la interoperabilidad semántica entre sus miembros. Se utiliza el estándar HLA como herramienta de construcción de una simulación distribuida. La red de ontologías define el vocabulario necesario para la especificación de un modelo de interoperabilidad, el cual

cumple la condición de ser semánticamente interoperable. Las reglas de integridad definidas en la red de ontología, permiten la construcción de modelos completos y consistentes desde el punto de vista sintáctico y semántico. Mientras que las reglas de derivación propuestas permiten unificar los conceptos necesarios de SD basada en HLA.

La red de ontologías *SCFHLA* posee dos dominios fundamentales, los cuales son: CS y federaciones HLA. El dominio de cadena de suministro modela conceptos de SCOR como procesos, flujos, métricas y atributos de rendimiento. El dominio de federaciones modela conceptos de federaciones HLA como federados, federaciones, objetos de usuario e interacciones. Para vincular la información de los conceptos de CS con los conceptos de federaciones HLA, se definieron en la red meta relaciones (relaciones entre conceptos de diferentes ontologías) y reglas de mapeo. La red vincula los individuos de cadena de suministro instanciados, con nuevos individuos de la ontología de federaciones HLA. Para ello, se ejecutan una serie de reglas de mapeo entre ambos dominios. Estas reglas vinculan los conceptos utilizados en la definición del modelo de CS, a simular, con conceptos necesarios de SD basada en HLA. Todas las reglas de la red se encuentran definidas en *Semantic Web Rule Language* (SWRL).

Para transformar la información contenida en la red de ontologías en el modelo de objetos de federación, se utiliza un algoritmo de transformación. El mismo convierte el archivo OWL de la ontología a un archivo con formato XML, el cual se adecua a las definiciones de la plantilla de modelo de objeto OMT (por sus siglas en inglés –*Object Model Template*-) del estándar HLA. El uso de este algoritmo de transformación permite contar con el modelo de objetos de federación en el formato adecuado, para su posterior uso en la simulación distribuida basada en HLA.

Para validar el enfoque propuesto, se desarrolla un caso de estudio donde se muestra el modelado de una cadena de suministro del sector industrial del mueble. Para desarrollar la prueba de concepto, se presenta la construcción del modelo de interoperabilidad en la red de ontologías. Luego, se utiliza el algoritmo de transformación para convertir el modelo de interoperabilidad en un documento con formato XML, el cual pueda ser utilizado como FOM en la SD basada en HLA del caso de estudio modelado. Los resultados parciales de los trabajos de investigación, llevados a cabo en el marco del desarrollo de esta tesis, han sido plasmados y divulgados a través de revistas indexadas, congresos nacionales e

internacionales. A su vez, han sido presentados en diferentes jornadas del ámbito científico. A continuación, se detallan dichas contribuciones:

1. Sarli, Juan L; Leone, Horacio; Gutiérrez, Milagros. (2018). "SCFHLA: Un Modelo de Interoperabilidad Semántica para Simulación Distribuida de Cadenas de Suministro". *RISTI – Revista Ibérica de Sistemas y Tecnologías de Información, (30), 34-50.* ISSN: 1646-9895. https://doi: 10.17013/risti.30.34–50.

2. Sarli, Juan L (2018). "An Interoperability Model for Collaborative Development of Distributed Supply Chain Simulations". En Proceedings of the 2018 Winter Simulation Conference. Pp 4222-4223 Editorial IEEE Press. ISBN: 978-1-5386-6571-8

3. Sarli, Juan L (2017). "Ontology-Based Modelling Framework to Generate Federation Object Model in the Supply Chain Domain". En Proceedings of the 2017 Winter Simulation Conference. Pp 4636-4637. Editorial IEEE Press. DOI: 10.1109/WSC.2017.8248242.

4. Sarli, Juan L.; Gioria, Lucas; Gutierrez, Ma de los Milagros; Leone, Horacio. (2017). "Una Herramienta para el Modelado y Generación del Modelo de Objetos de Federación en Simulaciones HLA". Memorias del 5º CoNaIISI Categoría Docentes-Investigadores tipo Artículo de Investigación. Pp. 572-580. ISSN 2347-0372.

5. Sarli, Juan L.; Gutierrez, Ma de los Milagros; Leone, Horacio. (2016). "Ontology-Based Semantic Model of Supply Chains for Modelling and Simulation in Distributed Environment". En Proceedings of the 2016 Winter Simulation Conference. Pp 1182-1193. Editorial IEEE Press. DOI: 10.1109/WSC.2016.7822175.

6. Sarli, Juan L.; Leone, Horacio; Gutierrez, Ma de los Milagros. (2016). "OpenSCOR: Framework para Análisis de Performance en Simulaciones de Cadenas de Suministro". Anales del 5° Simposio Argentino de Informática Industrial. Pp. 119-130. ISSN 2451-7542.

7. Sarli, Juan L.; Gutierrez, Ma de los Milagros; Leone, Horacio. (2015). "Hacia la Interoperabilidad Semántica en Simulaciones Distribuidas de Cadenas de Suministro". Memorias del 3º CoNaIISI Categoría Docentes-Investigadores tipo Artículo de Investigación. ISBN 978-987-1896-47-9.

8. Sarli, Juan L.; Gutierrez, Ma de los Milagros. (2015). “SCK: Una ontología para evaluar la performance de una cadena de suministro en ambientes de simulación distribuida”. CEUR Workshop Proceedings 1st Argentine Symposium on Ontologies and their Applications. ISSN 1613-0073. Volumen 1449. Pp 31-40.

9. Sarli, Juan L.; Gutierrez, Ma de los Milagros. (2015). “SCK: Una ontología para evaluar la performance de una cadena de suministro en ambientes de simulación distribuida”. Anales del 1° SAOA 1st Argentine Symposium on Ontologies and their Applications. Pp. 31-40. ISSN 2451-7518.

Índice

Lista de Figuras

Lista de Tablas

Nomenclatura

HLA: Representa el estándar de la IEEE denominado *High Level Architecture*, el cual es un estándar de facto entre los desarrolladores de simulaciones distribuidas. Lo desarrolló el ministerio de defensa de los Estados Unidos.

SD: Simulación distribuida. A lo largo de la tesis cuando se refiere a un desarrollo de simulación distribuida que utiliza el estándar *HLA*, se lo denomina como simulación distribuida basada en *HLA*. Además, como sinónimo del término simulación distribuida basada en *HLA* se utiliza el término federación *HLA*.

FOM: Representa el modelo de objetos de federación, y se deriva de su nombre en inglés el cual es *Federation Object Model*. Este documento es un requisito fundamental para el desarrollo de una simulación distribuida basada en el estándar *HLA*, sin este documento no es posible ejecutar la simulación distribuida.

OMT: Representa a la plantilla de modelo de objetos, y se deriva de su nombre en inglés el cual es *Object Model Template*. Es uno de los tres componentes esenciales del estándar *HLA*.

M&S: Referente al campo de modelado y simulación.

CS: Cadena de suministro.

SWRL: Se deriva de su nombre en inglés, el cual es *Semantic Web Rule Language*. Permite inferir nuevo conocimiento a partir de la información que existe en una ontología. Además, expresa las propiedades de los conceptos y las características del dominio.

Capítulo 1 **Introducción**

En este capítulo se introduce el contexto de la problemática que aborda esta tesis. Además, se definen los objetivos a alcanzar, las principales contribuciones y, por último, se presenta la organización de la tesis. Las temáticas principales de este capítulo son: los problemas de interoperabilidad semántica que surgen en una simulación distribuida de cadena de suministro, y la falta de herramientas para el diseño y construcción de modelos de interoperabilidad.

1.1 Contexto

En los últimos años, los avances tecnológicos, la globalización, el internet, la segmentación de mercado, los nuevos mercados, la personalización de productos y la proliferación de nuevas necesidades, han generado que la satisfacción de los clientes sea una tarea cada vez más difícil de lograr por parte de las empresas. Por estas razones, las empresas comprendieron que el éxito de la organización depende de la colaboración y coordinación con sus proveedores, distribuidores, como así también con sus clientes para beneficiarse de la integración global (Arrazola 2007; Mezgár y Rauschecker 2014; Camarinha-Matos et al. 2009). Entre los principales beneficios de la colaboración y coordinación, se pueden destacar: la reducción de costos, un aumento de la satisfacción del cliente, menor nivel de inventarios, mayor nivel de calidad de servicio y mayor flexibilidad para afrontar los cambios en la demanda (Mejía Argueta y Higuita Salazar 2015; Friesen et al. 2012; Chan y Zhang 2011). Estos cambios y necesidades, llevaron a las empresas a buscar nuevas formas de estructura organizacional para satisfacer las demandas, cada vez más exigentes y cambiantes, de sus clientes; así una de las alternativas más utilizadas por las organizaciones es agruparse con los proveedores y distribuidores en una red, para formar de este modo, una cadena de suministro

(CS). Se puede definir una cadena de suministro como, *una de las redes organizacionales más populares, donde sus miembros realizan alianzas para alcanzar metas más importantes de las que obtendrían de forma aislada* (Chung, Yam, y Chan 2004; Hearnshaw y Wilson 2013). Esta forma de organización, permite afrontar los nuevos desafíos que plantea el mundo actual en cuanto a la satisfacción de las necesidades de los clientes. En esta definición de CS, se encuentra implícita la necesidad de colaboración y coordinación en las alianzas realizadas entre los miembros de la red organizacional.

La nueva estructura organizacional de CS impulsó la evolución de la gestión, con lo que surgió el término gestión de la cadena de suministro como una nueva forma integrada de administrar toda la red de organizaciones (Oliveira, Lima, y Montevechi 2016; Álvarez, Díaz, y Larrinaga 2011). Este término se refiere a la necesidad de integrar los flujos de información, dinero y materiales, de manera eficiente y eficaz en la estructura de CS, con el fin de alcanzar metas que las empresas de la red no alcanzarían de forma aislada (Cho et al. 2012; Chavez 2012; Montoya-Torres y Vargas 2011). Si bien la integración de los flujos mencionados permite alcanzar nuevas metas y mejorar el desempeño total de la CS, en la práctica contemporánea dicha integración se limita a algunas configuraciones conocidas, en donde una organización coordina la colaboración con el resto de las empresas de la CS a su medida. En las cadenas de suministro donde no existe una organización dominante, el principal obstáculo es cómo evaluar los beneficios obtenidos y esfuerzos requeridos, además de determinar cómo impactan ambos en las empresas que participan de la colaboración. De este modo, resulta evidente que es necesario disponer de herramientas para analizar distintos escenarios de colaboración, en donde sea posible evaluar el desempeño integral de la CS, así como también, los beneficios y costos que implican para cada organización participante. En este contexto, la simulación emerge como una herramienta fundamental para evaluar las distintas configuraciones o escenarios, con el fin de alcanzar el éxito en la CS.

La simulación permite analizar los procesos y las interacciones de las organizaciones que pertenecen a la CS, en donde el análisis puede ser sobre la configuración actual o sobre distintas alternativas. De este modo, los resultados de la simulación permiten a los analistas de negocios o modeladores de cadena de suministro, obtener un análisis más certero de que configuración es la más adecuada para la CS. Como ventajas de la simulación se pueden

destacar las siguientes (Lustig et al. 2010; Hennies et al. 2014; Rahimi, Møller, y Hvam 2016; Aversano, Grasso, y Tortorella 2016; Ali, Petersen, y de França 2015):

- Análisis de escenarios (denominado en la literatura como *what if analysis*), se configuran escenarios alternativos en donde los procesos e interacciones de la CS reaccionan a una serie de condiciones o eventos, como por ejemplo: aumento de la demanda, disminución del nivel de inventario de una materia prima o producto, ausencia de un recurso (sea recurso humano, maquinaria o insumo de producción) determinado.
- Evaluación de rendimiento o utilización, se analizan indicadores de utilización de recursos, tiempo de espera entre tareas, tiempo ocioso de los recursos, tiempo de entrega de las órdenes, porcentajes de ordenes entregadas de forma correcta.
- Detección de oportunidades de mejora o cambio, se analizan las configuraciones de CS para evaluar si es necesario modificar los procesos e interacciones.

Para hacer uso de la información cuantitativa obtenida como resultado de una simulación, es necesario construir modelos de simulación de los procesos y las interacciones involucradas en la cadena de suministro a analizar. Entonces, se plantea la siguiente pregunta ¿Qué modelo de procesos e interacciones de CS representarán los modelos de simulación? Para responder a esta pregunta, en el año 1996, diversas organizaciones crearon el *Supply Chain Council* (SCC) que definió el modelo de referencia de operaciones de cadena de suministro SCOR (por sus siglas en inglés -*Supply Chain Operations Reference*-) (Supply Chain Council 2012). La versión 11 del modelo posee un conjunto de métricas, procesos, buenas prácticas, herramientas de diagnóstico y aplicaciones especiales que son ampliamente utilizadas por las distintas organizaciones que conforman una CS. El modelo SCOR muestra cuales son los conceptos relevantes para modelar una CS, como así también presenta los distintos atributos de rendimiento y métricas para realizar una evaluación de la cadena, tanto a nivel global como a nivel operativo. Tanto las métricas como los procesos definidos se encuentran en una organización jerárquica, es decir, en diferentes niveles. Las relaciones entre los distintos niveles son diagnósticas, por ejemplo: las métricas de nivel 2 sirven como diagnóstico para las de nivel 1. Esto significa qué observando el rendimiento de las métricas de nivel 2, se puede explicar las diferencias de performance, o las posibles mejoras, para las

métricas de nivel 1. Este tipo de análisis de rendimiento de la CS se conoce como descomposición de métricas. De esta forma, es posible comparar el rendimiento entre distintas cadenas de suministro, analizar las causas de deficiencias en los procesos, lograr mayores beneficios y aumentar la satisfacción de las necesidades de los clientes.

A medida que las cadenas de suministro crecen en tamaño (mayor número de organizaciones participantes) y en complejidad (involucran más procesos e interacciones, así como la lógica interna de los procesos es más compleja), sus respectivos modelos de simulación crecen de forma directamente proporcional a estas dimensiones. Además, cuando la cadena de suministro posee organizaciones distribuidas geográficamente, en donde no existe una empresa dominante que coordine la colaboración con las demás organizaciones, donde se producen cambios en los integrantes de la CS y, en consecuencia, existe desconfianza para compartir información sobre la lógica interna de los procesos intervinientes en la colaboración; entonces, en este contexto, la simulación de forma tradicional se torna muy difícil o imposible, entendiendo por simulación de forma tradicional a *una única persona que construye un único modelo de simulación y, ejecuta experimentos en una única computadora* (Gogi, Tako, y Robinson 2016; Proudlove et al. 2017; Tako y Kotiadis 2015). Por estos motivos, la simulación distribuida (SD) emerge como la más adecuada para simular las cadenas de suministro con estas características. El término SD puede ser definido como *la ejecución distribuida de programas de simulación a través de múltiples procesadores*, y se refiere a la ejecución de modelos de simulación por computadora en plataformas paralelas o distribuidas (Nouman, Anagnostou, y Taylor 2013; D'Angelo 2011). La SD permite construir un modelo complejo de simulación a partir de integrar simuladores (con sus respectivos modelos de simulación) distribuidos en distintas ubicaciones, ejecutados en paralelo y que intercambian información. Este enfoque de simulación, permite modelar el comportamiento de las cadenas de suministro como sub modelos, que encapsulan el conocimiento y funcionamiento de cada organización, ejecutados en distintas ubicaciones geográficas de una red de computadoras, y que se integran para componer el modelo de simulación de la CS en su totalidad (Robinson 2002; R.M. Fujimoto 2000; Richard M. Fujimoto, Malik, y Park 2010).

Si bien la simulación distribuida trae aparejadas numerosas ventajas para la simulación de cadenas de suministro, existen varios inconvenientes que aun se están investigando para lograr una mayor difusión y uso de este tipo de simulación.

Uno de los problemas se debe a que los paquetes de simulación comerciales CSP (por sus siglas en inglés -C*ommercial off the shell Simulation Packages*-), no presentan un soporte directo para SD. Esto se debe a que no soportan la interoperabilidad entre modelos de simulación de distintos CSP, no permiten el acceso a la lista de eventos, ni la inclusión de eventos externos, entre otros. Este problema ha sido tan relevante que la *Simulation Interoperability Standards Organization* y el *CSP Interoperability Product Development Group* trabajaron en conjunto en la definición del modelo denominado *Interoperability Reference Model*. Este modelo provee un marco de referencia para especificar los requerimientos de interoperabilidad a nivel de modelado, con el fin de que los desarrolladores de CSP identifiquen que es lo necesario para lograr la interoperabilidad con otros CSP, en una simulación distribuida (S. J. E. Taylor et al. 2012).

Otro de los problemas, se refiere a la complejidad inherente a realizar una SD de CS. Llevar a cabo la simulación distribuida de cadenas de suministro, implica la convergencia de conceptos de tres grandes áreas, las cuales son: gestión de operaciones OM (por sus siglas en inglés -*Operations Management*-), investigación de operaciones OR (por sus siglas en inglés -*Operations Research*-) y computación aplicada AC (por sus siglas en inglés -*Applied Computing*-). La Figura 1.1 presenta la convergencia entre estas disciplinas, en donde se observa que el área de gestión de operaciones se muestra en color azul, la investigación de operaciones en color rojo y la computación aplicada en amarillo. Los conceptos vinculados a OM son aquellos relacionados a la gestión de CS, como ser: gestión de logística e inventario, cadena de valor y localizaciones de las unidades de negocio, entre otros. Estos conceptos se utilizan para formular ó definir el problema a simular. Los conceptos vinculados a OR son aquellos relacionados a la técnica de modelado y simulación (M&S) a utilizar, como ser: simulación basada en agentes, de eventos discretos, de Monte Carlo y dinámica de sistemas. Dichas técnicas se utilizan para implementar el comportamiento del modelo a simular. Los conceptos vinculados a AC son aquellos relacionados a computación distribuida ó paralela,

como ser: simulación distribuida, simulación basada en la nube o *grid*. Estos conceptos se utilizan para ejecutar los modelos o para llevar a cabo los experimentos de la simulación.

La intersección entre la OM y la AC, presentada en color verde en la Figura 1.1, ha conllevado el desarrollo de modelos de dominio para cadenas de suministro. La relación de conceptos de OR y AC, presentada en color naranja en la Figura 1.1, ha estimulado el estudio de estándares de interoperabilidad. Por último, la convergencia entre la OM y la OR, presentada en color violeta en la Figura 1.1, ha incentivado el estudio de técnicas de M&S en las cadenas de suministro. Por estos motivos, en el trabajo de (Mustafee, Katsaliaki, y Taylor 2014) se propone que la SD de CS sea considerada un nuevo tema de investigación, que resulte de la intersección o convergencia de las disciplinas de OM, OR y AC (presentada en color negro en la Figura 1.1). La complejidad de este tipo de simulaciones en el dominio de cadenas de suministro es tan grande, que se ha propuesto una metodología para desarrollar simulaciones distribuidas en el ámbito de cadenas de suministro (Anagnostou y Taylor 2017).

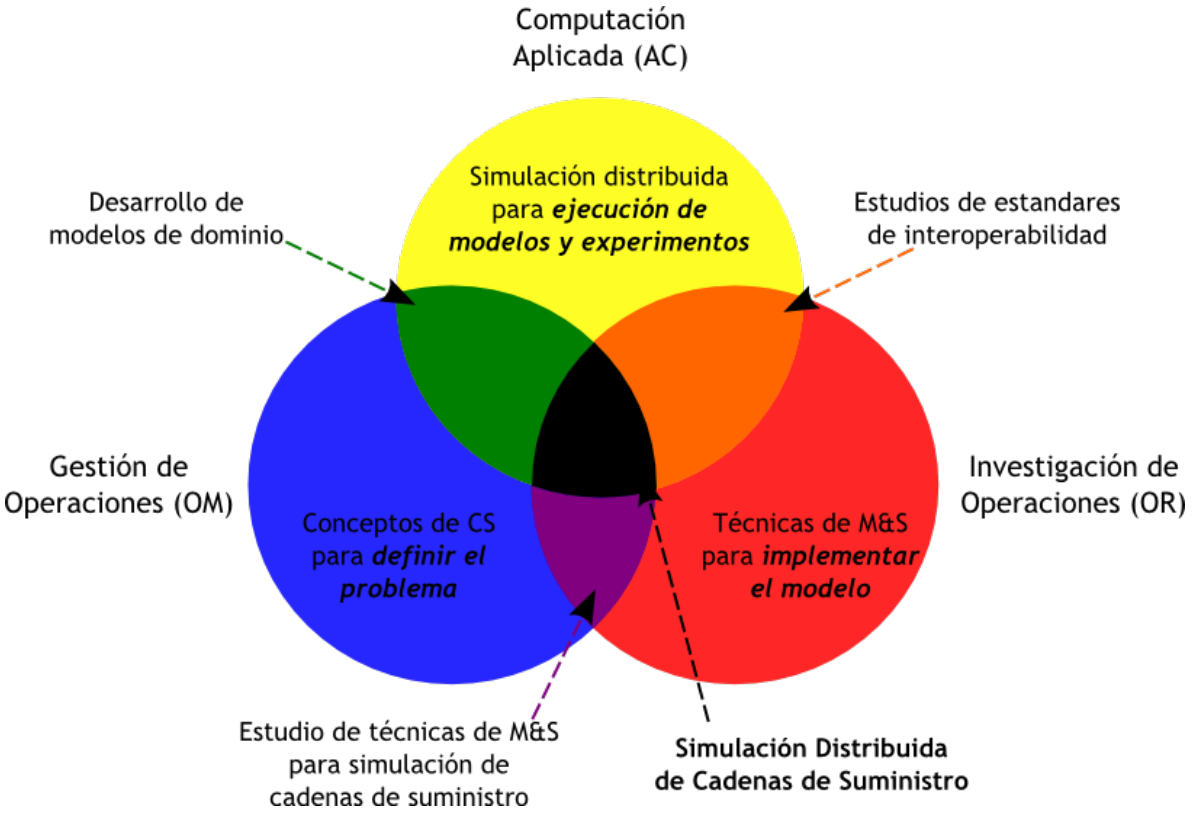


Figura 1.1 – La SD de CS como convergencia de tres grandes disciplinas
(Adaptado de (Mustafee, Katsaliaki, y Taylor 2014))

Otro de los problemas se refiere a la integración de los simuladores y modelos, para componer el sistema de simulación distribuida total (o de mayor nivel). Diversas investigaciones han señalado que esta temática aún continúa siendo un gran desafío, dado que las recomendaciones para lograr la integración de simuladores, no han alcanzado un grado satisfactorio (R. Fujimoto 2015; S. J. E. Taylor et al. 2015; Rycerz et al. 2015; Sekiguchi et al. 2012). Para componer el sistema de SD de mayor nivel, es necesario asegurar que los simuladores participantes son: *integrables* (se ejecutan en infraestructuras que son integrables), *interoperables* (el sistema de simulación distribuida soporta protocolos de interoperabilidad consistentes) y *componibles* (los modelos conceptuales subyacentes no son contradictorios) (S. J. E. Taylor et al. 2013). Como se puede observar, cada una de estas condiciones está asociada al nivel de interoperabilidad que se logre entre los simuladores participantes de una SD.

En el marco de esta tesis, se trabaja sobre el concepto de interoperabilidad entre simuladores y, se lo define como *la interpretación realizada por un simulador de los eventos enviados por otro simulador, de forma que estos eventos provoquen los efectos esperados por quien diseña el modelo de simulación ejecutado en el otro simulador*. Cabe destacar que existen distintos niveles de interoperabilidad y para alcanzar un cierto nivel, es necesario alcanzar el nivel anterior. Según (Tolk, Bair, y Diallo 2013) es posible determinar tres grandes niveles: 1) *Sintáctico*, 2) *Semántico* y 3) *Pragmático*. El nivel sintáctico implica que los simuladores pueden comunicarse e intercambiar eventos a través de una infraestructura de comunicación. El nivel semántico determina si la comunicación e intercambio de eventos es semánticamente válida, lo que permite detectar si la composición de la simulación es significativa y válida (Tolk, Diallo, y Padilla 2012; Petty, Weisel, y Mielke 2005; 2004). Dicho de otro modo, este nivel hace referencia a si la interpretación por parte de los simuladores, de los datos o eventos intercambiados, es comprendida por todos sin ambigüedades y de la misma forma en la composición de la SD. En el caso que se alcance este nivel de interoperabilidad, se puede decir que la composición de la simulación distribuida es significativa y válida (Mezgár y Rauschecker 2014). El nivel pragmático implica que todos los simuladores de la simulación distribuida utilizan los datos de la misma manera. Además, este nivel pretende que cada simulador posea un modelo conceptual para

hacer explicitas las restricciones definidas, hipótesis supuestas y simplificaciones realizadas en el modelo de simulación que están ejecutando, a fin de determinar que los modelos conceptuales subyacentes no son contradictorios. En la Figura 1.2 se presentan los niveles de interoperabilidad y cuál es la condición de interoperabilidad que se obtiene en cada nivel.

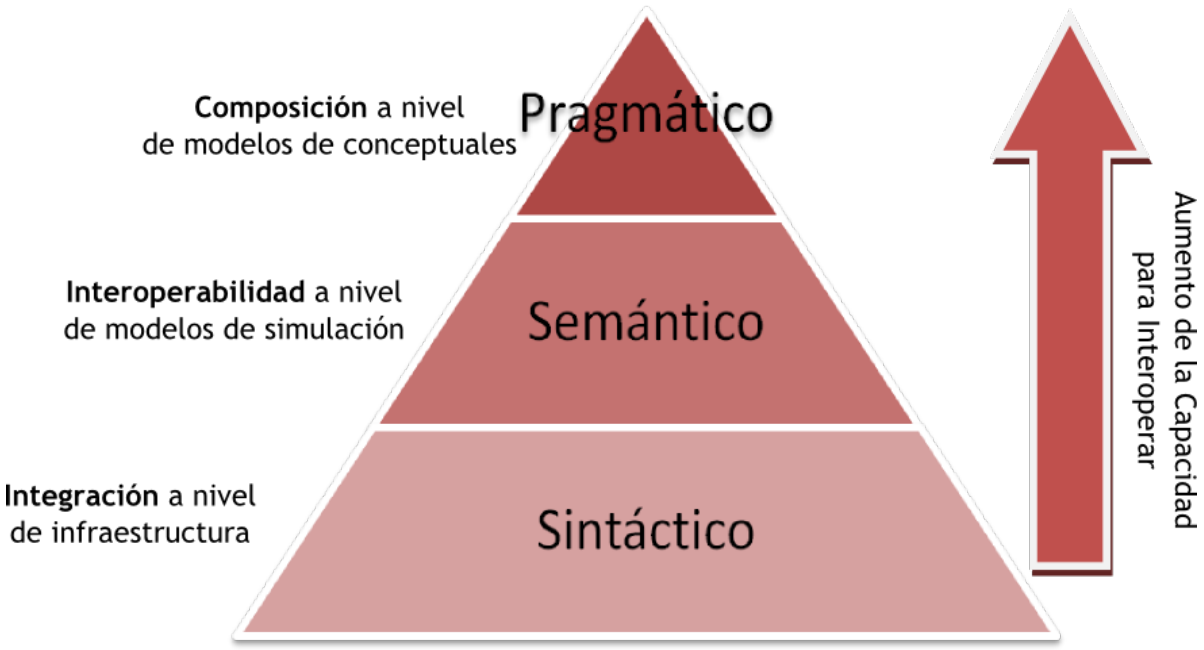


Figura 1.2 – Niveles de interoperabilidad entre simuladores

El estándar del *Institute of Electrical and Electronics Engineers* (IEEE) *High Level Architecture* (HLA) 1516-2010 (IEEE 2010a), denominado HLA *Evolved*, es ampliamente utilizado para solucionar los problemas de interoperabilidad a nivel sintáctico en simulaciones distribuidas. Esta nueva versión de HLA ha incorporado soporte para comunicación mediante servicios web, información modular para los modelos de objetos y compatibilidad para la vinculación dinámica de distintas implementaciones del *Runtime Infrastructure* (RTI). Con estas nuevas características se han solucionado algunos problemas de las versiones anteriores. Este estándar utiliza el concepto de modelo de objetos de federación FOM (por sus siglas en inglés -*Federation Object Model*-), que provee una especificación común para el intercambio de datos entre simuladores en un formato estándar. En síntesis, el FOM establece un contrato del modelo de información que es necesario, pero no es suficiente, para alcanzar la interoperabilidad semántica entre simuladores. A partir de

los lineamientos propuestos por HLA para construir el modelo de objetos de federación, es posible asegurar la interoperabilidad sintáctica entre los simuladores intervinientes. Sin embargo, la interoperabilidad semántica está ajena a este estándar y es responsabilidad exclusiva de sus miembros garantizarla. El FOM provee las etiquetas necesarias para interpretar correctamente interacciones y objetos, aunque los valores que tomen estas etiquetas deben ser semánticamente acordados para que la simulación tenga resultados válidos (Tolk, Bair, y Diallo 2013; Tolk, Diallo, y Padilla 2012). Este acuerdo sobre los valores se considera una solución para el tratamiento del significado de la información, mediante la construcción de recursos individuales semánticamente consistentes (Mezgár y Rauschecker 2014; Piedra et al. 2015; Piedra y Suárez 2018).

A la hora de desarrollar una simulación distribuida de cadena de suministro, una de las tareas que más esfuerzo conlleva es la construcción del modelo de objetos de federación (Jian, Yang, y ZiYang 2017; Jain et al. 2015; Zhu y Liu 2014). Para definir el FOM, una alternativa es que la información requerida, sea definida por una única persona u organización para cada escenario de CS a simular en la SD basada en HLA. En este caso, no existen problemas de interpretación de los datos, ya que toda la información se interpreta como establece dicha persona u organización. Otra alternativa surge cuando la cadena de suministro no presenta una organización dominante, y entonces, la construcción del FOM se realiza de forma colaborativa y manual. En este caso las organizaciones ó personas, acuerdan de forma previa al desarrollo del modelo de objetos de federación, la información que va a contener el mismo y su significado. Para ambos casos, las herramientas disponibles son editores que garantizan la consistencia sintáctica del FOM, es decir, solo verifican que la información contenida en el mismo se encuentra en el formato correcto y con las etiquetas definidas en el lenguaje *eXtensible Markup Language* (XML) (W3C 2015). Por estos motivos, la construcción del modelo de objetos de federación de forma manual tiene las siguientes desventajas: es propensa a errores, el modelo obtenido no resulta sencillo de comprender (aún por los expertos), y no existe un mecanismo que garantice la completitud y consistencia del modelo (Sun et al. 2012). En este contexto, se define completitud como *que el modelo posea todas las etiquetas requeridas*, y consistencia como *que la información contenida en el modelo de objetos sea la misma que la definida en el modelo de CS*.

El objetivo de la interoperabilidad semántica es lograr que sistemas de información autónomos, comprendan el significado de la información generada y compartida por otros sistemas. Por este motivo, para que la SD basada en HLA de una CS colaborativa (en donde no existe una organización dominante) pueda alcanzar este nivel de interoperabilidad, es necesario que las organizaciones acuerden cómo construir el FOM. De modo tal que, el modelo de objetos de federación represente el conocimiento de los datos a intercambiar entre los simuladores, de una forma estandarizada y sin ambigüedades. En este punto es donde surge la posibilidad de emplear ontologías, a efectos de expresar el conocimiento común que utilizan los diversos simuladores. Las ontologías son usadas para organizar la representación del conocimiento en un determinado dominio, y razonar en base a él. Una ontología se puede concebir como, una definición explicita de una conceptualización compartida de cierto dominio (Gómez-Pérez, Fernández-López, y Corcho 2004).

Además en este contexto, la simulación se utiliza como una herramienta de evaluación del rendimiento de la CS, y como soporte para la toma de decisiones. Quienes utilizan la simulación como herramienta de evaluación, son los analistas de negocio y los modeladores de CS. Los mismos poseen un amplio conocimiento de los conceptos asociados a cadenas de suministro, pero no así sobre las herramientas para implementar una SD basada en HLA. Por lo tanto, disponer de herramientas que brinden términos familiares a estos usuarios, y permitan encapsular los conocimientos específicos sobre HLA para realizar una SD, resulta de gran interés. En conclusión por lo expresado en los párrafos anteriores, queda clara la necesidad que existe de herramientas de software para diseñar y construir colaborativamente el modelo de objetos de federación, en una SD basada en HLA de cadenas de suministro.

1.2 Objetivos

Para la toma de decisiones en la gestión de cadena de suministro, en un contexto colaborativo sin una empresa dominante, es necesario analizar distintos escenarios en la simulación distribuida; a efectos de evaluar cual es la mejor configuración para el objetivo propuesto. Realizar esta evaluación, por parte de los analistas de negocio, implica contar con un amplio conocimiento en tres grandes disciplinas.

A partir de esta breve descripción del contexto de las cadenas de suministro, se puede vislumbrar la complejidad del objeto de estudio de esta tesis, y la necesidad de contar con herramientas eficaces que ayuden en la gestión colaborativa de la misma. Por lo tanto, resulta fundamental el uso de un conjunto de herramientas de software a efectos de brindar soporte a las etapas de diseño, implementación, composición, validación y puesta en ejecución de la simulación distribuida, basada en HLA, de cadenas de suministro. En tal sentido, y en consonancia con lo expuesto en la sección anterior, la presente tesis se plantea como **objetivo general** el diseño y desarrollo de un marco conceptual, el cual permita el modelado y composición de un modelo de interoperabilidad que de soporte a la ejecución de la simulación distribuida de cadenas de suministro, a fin de garantizar la interoperabilidad semántica entre los miembros de la simulación.

Para lograr el objetivo general propuesto, el mismo se ha desglosado en los siguientes **objetivos específicos**:

- Determinar cuáles son los requerimientos, desde el punto de vista técnico, para ejecutar una simulación distribuida. Para definir estos requerimientos se tendrá en cuenta los aspectos de infraestructura, protocolos y formato de intercambio de información entre los simuladores participantes de la simulación.
- Identificar los conceptos empleados al especificar qué políticas se utilizan para la gestión de la cadena de suministro, en un contexto colaborativo sin una organización dominante. Los conceptos a modelar se vinculan con los procesos centrales, las interacciones entre los mismos, los atributos de rendimiento medibles, y el conjunto de métricas, directas e indirectas, asociadas para la evaluación diagnóstica del rendimiento de la cadena de suministro en una simulación.
- Desarrollar una red de ontologías que contenga el vocabulario asociado con las necesidades de ejecución de una simulación distribuida, y de gestión de la cadena de suministro. Este vocabulario se utiliza para el modelado y composición del modelo de interoperabilidad, que cumple la condición de ser semánticamente interoperable.
- Definir un mecanismo para verificar que el modelo de interoperabilidad es completo y consistente, desde el punto de vista sintáctico y semántico.

- Diseñar un algoritmo de transformación que permita la construcción del modelo de objetos de federación, en el formato adecuado, para su utilización en la simulación distribuida basada en HLA, del escenario de cadena de suministro modelado.
- Identificar y analizar los requerimientos de las herramientas de soporte de toma de decisión, para la gestión y evaluación de cadenas de suministro. Evaluar las ventajas y desventajas del uso del enfoque propuesto, por parte de los analistas de negocio y modeladores de cadena de suministro.
- Implementar un prototipo de la red de ontologías y el algoritmo de transformación, a efectos de desarrollar un caso práctico para validar el enfoque propuesto.

1.3 Principales Contribuciones

El principal aporte de esta tesis es la definición de un marco conceptual basado en una red de ontologías, el cual soporta las tareas de modelado y composición del modelo de interoperabilidad. La red de ontologías propuesta define el vocabulario necesario para la construcción del modelo de interoperabilidad, que cumple con la condición de garantizar la interoperabilidad a nivel semántico, entre los participantes de la simulación distribuida basada en HLA, de cadenas de suministro. Las ventajas de utilizar el marco conceptual se pueden resumir en:

- La disminución del conocimiento específico requerido, para la construcción del modelo de objetos de federación, en cuanto al estándar HLA. El marco conceptual permite obtener un modelo de interoperabilidad, el cual se utiliza en la construcción del FOM (en una etapa posterior). El modelo de interoperabilidad se construye a partir del modelo de estructura y comportamiento de la CS a simular.
- La especificación de un mecanismo para verificar la completitud y consistencia del modelo de interoperabilidad. Por medio de las reglas definidas, es posible verificar que la información del modelo de interoperabilidad, se corresponda con la información del modelo de cadena de suministro especificado. Además mediante el uso de un algoritmo de transformación, la información del modelo de interoperabilidad se convierte en un documento con el formato adecuado y con todas

las etiquetas requeridas, a efectos de ser utilizado como FOM en la SD basada en HLA del escenario modelado.

- La reducción del tiempo y esfuerzo necesarios para el diseño y construcción del modelo de objetos de federación. Las herramientas existentes son muy rudimentarias, y requieren de un amplio conocimiento sobre el estándar HLA para utilizarse de forma correcta. Sin este conocimiento y sin el consenso sobre la definición del FOM, el uso de las herramientas actuales puede conllevar meses de esfuerzo y tiempo, invertido en una tarea manual.
- La reducción de la intervención humana necesaria para la construcción del modelo de objetos. El mayor esfuerzo en cuanto a tareas manuales se produce en las tareas de modelado, mientras que las demás actividades requieren mínima intervención manual.
- La simplicidad en el uso. El marco conceptual presenta conceptos familiares para los analistas de negocios o quienes modelan una CS. De esta forma, les resulta transparente utilizar el marco conceptual para obtener el modelo de interoperabilidad.
- La definición de un vocabulario común para definir el modelo de cadena de suministro a simular.
- La vinculación, por medio de reglas, entre conceptos utilizados en la definición del modelo de cadena de suministro a simular, y conceptos necesarios de simulación distribuida.
- La construcción del modelo de objetos de federación, de forma automática, a partir de la información contenida en la red de ontologías.

1.4 Organización de la Tesis

En el capítulo 2 se describen los conceptos fundamentales asociados con el desarrollo de esta tesis. Entre ellos se hace una breve descripción sobre cadenas de suministro y los tipos de colaboración, haciendo hincapié en el modelo SCOR. Dicho modelo, se utiliza como base para modelar los conceptos de CS. Luego se presenta la definición de simulación distribuida, y se describe en detalle el estándar HLA, como herramienta para la construcción y ejecución de una SD. Además, se revén sintéticamente las técnicas de M&S para los

modelos de simulación. Por último se presentan los conceptos de ontologías y redes de ontologías, así como también la metodología de desarrollo de ontologías a utilizar, y las aplicaciones de ontologías para la interoperabilidad entre sistemas.

En el capítulo 3 se presentan las problemáticas asociadas a la SD basada en HLA de CS, a partir de las cuáles se especifican los requerimientos para el modelo conceptual de la red de ontologías denominada *SCFHLA*. Además se presenta el diseño de las ontologías que componen la red, a partir de los dominios de cadenas de suministro y federaciones HLA; luego se detalla cómo se integran las ontologías de dominio a la red de ontologías. Tanto en cada uno de los dominios como en la red de ontologías, se define un conjunto de reglas para: garantizar la validez de los conceptos instanciados y relacionados, inferir nueva información y, por último, mapear instancias del dominio de cadenas de suministro al dominio de federaciones HLA. A su vez, se detalla cómo se lleva a cabo la implementación de la red de ontologías. Finalmente, se realiza una evaluación de cada una de las ontologías de dominio, como así también de la red de ontologías, con el fin de detectar problemas de modelado, diseño, y cumplir con el objetivo de la red de ontologías.

En el capítulo 4 se presentan las problemáticas asociadas a la construcción de un modelo de objetos de federación, a partir de las cuáles se definen los requerimientos para una plataforma basada en el modelo conceptual desarrollado en el capítulo 3. Luego se presenta el diseño de la arquitectura de la plataforma, y la definición de sus componentes. Entre los componentes de la plataforma, se destaca el algoritmo de transformación. Este componente utiliza la información del modelo de interoperabilidad de la red de ontologías, para la construcción de un modelo de objetos de federación.

En el capítulo 5 se realiza una prueba de concepto por medio de un caso de estudio, donde se muestra el modelado de una cadena de suministro, del sector de la industria del mueble. Para llevar a cabo la prueba de concepto, se presenta la construcción del modelo de interoperabilidad en la red de ontologías. Luego, se utiliza el algoritmo de transformación, para convertir el modelo de interoperabilidad en un documento con formato XML, el cual pueda ser utilizado como FOM en la SD basada en HLA, del escenario modelado.

Finalmente, en el capítulo 6 se presentan las conclusiones y trabajos futuros de la tesis.

Capítulo 2 **Conceptos Fundamentales**

En este capítulo se describen los conceptos fundamentales asociados con el desarrollo de esta tesis. Entre ellos, se hace una breve descripción sobre cadenas de suministro y los tipos de colaboración, haciendo hincapié en el modelo de referencia SCOR, que se toma como base para modelar los conceptos de CS. Luego se presenta la definición de simulación distribuida, y se detalla el estándar HLA, el cual es una herramienta para la construcción y ejecución de una simulación distribuida. Además, se presentan sintéticamente las técnicas de M&S para los modelos de simulación. Por último, se presentan los conceptos de ontologías y redes de ontologías, así como también la metodología de desarrollo de ontologías a utilizar, y las aplicaciones de ontologías para la interoperabilidad entre sistemas.

2.1 Cadenas de Suministro

En la actualidad las necesidades de los consumidores son cada vez más variadas y cambiantes. Con el fin de adaptarse a las demandas globales, las organizaciones deben modificar sus estructuras para ser ágiles y mantenerse competitivas. Una forma de lograr estos cambios en la estructura organizacional, es mediante la conformación de una cadena de suministro. Una CS puede definirse como *una red de participantes (proveedores, fábricas, distribuidores, entre otros) que, por medio de actividades y planificaciones coordinadas, desarrolla productos al convertir las materias primas en productos terminados* (Yoo, Cho, y Yücesan 2010; Chandra y Grabis 2007). Esta red de organizaciones se ha popularizado ampliamente, y representa una composición de varias unidades autónomas (personas y organizaciones), distribuidas geográficamente, heterogéneas en cuanto a cultura, objetivos y operación, en donde se generan vínculos de colaboración entre sus miembros (Camarinha-Matos et al. 2009). De esta forma, cada uno de los participantes puede concentrarse en desempeñar un rol específico, para que la CS abarque todas las etapas de la cadena de valor.

Mediante esta nueva estructura, las organizaciones participantes pueden cumplir con objetivos de mayor envergadura, que los que obtendrían de forma individual.

Una CS es un tipo particular de red inter organizacional, donde participan distintas empresas o unidades organizacionales de la misma empresa, que establece vínculos entre sus participantes con el fin de ofrecer productos y servicios en mercados específicos. A su vez, una cadena de suministro, puede ser vista como: *una red de empresas autónomas o semiautónomas responsables de los procesos y actividades que van desde, la obtención de materia prima hasta la entrega del producto final al consumidor, pasando por la producción, fabricación, y distribución de los productos, asociados con una o más familias de productos relacionados.* Esta red de organizaciones, implica una compleja interacción de recursos, funciones y medios de distribución, para producir y entregar productos o servicios a los consumidores (Chandra y Tumanyan 2007; Long y Zhang 2014). En estas definiciones de CS, se encuentra implícita la necesidad de colaboración y coordinación en las alianzas realizadas, entre los miembros de la red organizacional. Las empresas que conforman una cadena de suministro, comprendieron que el éxito de la organización depende de la colaboración y coordinación con sus proveedores, distribuidores, como así también con sus clientes, para beneficiarse de la integración global (Ramanathan 2014; Raweewan y Ferrell 2018). Como se puede observar, la gestión de la CS ya no solo involucra a una organización, sino que transciende los límites organizacionales y alcanza a empresas autónomas, con distintos objetivos y políticas.

La nueva estructura organizacional de CS impulsó la evolución en la gestión, lo que dio lugar al término *gestión de la cadena de suministro* que, de acuerdo con (Oliveira, Lima, y Montevechi 2016; Álvarez, Díaz, y Larrinaga 2011), refiere a una nueva forma integrada de administrar una red de organizaciones. Desde una perspectiva general, este término se puede definir como: *un conjunto de estrategias utilizadas para integrar en forma eficiente proveedores, fabricantes, depósitos y centros de venta al consumidor, de modo tal que, los productos sean producidos y distribuidos en las cantidades correctas, en el lugar correcto y en el momento correcto, al menor costo, y con el nivel de servicio requerido* (Simchi-Levi, Kaminsky, y Simchi-Levi 1999; Dyer 2000). La gestión de una cadena de suministro es una actividad compleja, ya que requiere una alta interacción entre sus miembros, y el balance

entre conveniencias e intereses opuestos de las distintas empresas, o unidades organizacionales de la misma empresa. Las interacciones entre las organizaciones que conforman una CS, se regulan por medio de acuerdos, en donde se determina un intervalo de tiempo durante el cual se realiza la colaboración. De esta forma, los objetivos de una cadena de suministro, los miembros de la red, y la toma de decisiones suelen modificarse en el tiempo. Al existir esta dinámica de cambio, las organizaciones que participan en una cadena de suministro se centran en sus ventajas distintivas, y colaboran con otras empresas en las áreas en donde no poseen una diferencia competitiva. De esta forma, las organizaciones pretenden lograr una sinergia al integrar una CS. Por lo tanto, se desprende que el principal interés de la gestión de CS es: organizar los flujos de información, dinero, y materiales, para alcanzar metas que las empresas de la red no alcanzarían de forma aislada (Sabouhi, Pishvaee, y Jabalameli 2018; Cho et al. 2012).

En las interacciones entre los participantes de una cadena de suministro, se intercambian distintos recursos (materiales, financieros y de información). El flujo de materiales o productos fluye principalmente, desde la obtención de materias primas hacia la elaboración de productos elaborados (con mayor valor agregado), que son entregados a los clientes. En la jerga de gestión de CS, se denomina que este flujo fluye *aguas abajo* en la cadena. Sin embargo, también se debe considerar que los productos se tornan obsoletos, se dañan o requieren reparación, y puede ser necesario que retornen al lugar donde se los produjo o donde se los repara, recicla, o se les da un destino final en el caso que ya no sirven. En estos casos, el flujo de materiales ocurre en sentido inverso al principal.

El mayor flujo de recursos financieros en una cadena de suministro, ocurre como retribución de los productos o servicios recibidos por las empresas, por lo cual posee un sentido inverso al del movimiento principal de materiales. En la jerga de gestión de CS, se dice que este flujo fluye *aguas arriba* en la cadena.

Con respecto a los principales flujos de información, en una CS convencional (en la cual los clientes se abastecen en base a puntos de stock o mediante órdenes de pedidos) los mismos están ligados a procesos de consultas (precios, tiempos y condiciones de entrega), generación y gestión de órdenes de pedido, entre otros. La comunicación entre las empresas,

se limita a los contactos estrictamente necesarios para la buena marcha de sus respectivos negocios, por lo que el flujo de información ocurre aguas arriba en la cadena.

Sin embargo, hoy en día las empresas necesitan trabajar de forma colaborativa y coordinada para lograr mejores desempeños. En este nuevo contexto, es necesario mantener una comunicación fluida y confiable, entre las distintas organizaciones de la cadena. Este hecho representa un gran desafío, ya que para lograrlo se requiere una interpretación consistente de la información intercambiada. Esto no resulta sencillo debido a la heterogeneidad de los participantes de una CS, quienes poseen distintas culturas organizacionales, experiencias, conocimientos, formas de trabajo, tecnologías y lenguajes.

La estructura de una CS se puede definir como: *el conjunto de miembros que la conforman y los vínculos establecidos entre ellos*. En sus inicios, la estructura de una CS era concebida mediante una analogía con la estructura de una cadena física (en donde existe una sucesión unidimensional de eslabones conformados por proveedores o clientes individuales). En la actualidad la estructura de una CS difiere de esta primera concepción, ya que en la mayoría de los casos, involucra redes donde cada organización está vinculada con múltiples empresas. Este tipo de estructuras, brinda la posibilidad de que cada organización (entendida como un nodo de la CS) pueda vincularse con organizaciones externas, dando lugar al surgimiento de una estructura de árbol (Birasnav y Bienstock 2019; Yunus y Tadisina 2016; Flynn, Huo, y Zhao 2010). Los vínculos entre los distintos nodos de una CS, están relacionados con los flujos de recursos analizados con anterioridad.

En la Figura 2.1 se presenta una conceptualización de la estructura de cadena de suministro, compuesta por distintos tipos de organizaciones (proveedores de materias primas, proveedores, fábricas, centros de distribución, centros de venta y consumidores finales), y los flujos de recursos que se intercambian. En esta conceptualización, los tipos de organizaciones participantes se representan mediante cuadriláteros de distintos colores, los flujos de recursos se presentan como arcos dirigidos, que reflejan el flujo aguas arriba o aguas abajo de la cadena, y las distintas interacciones entre las organizaciones de la cadena de suministro se exhiben mediante arcos bidireccionales.

Una organización puede formar parte de más de una CS, y colaborar en ellas de forma simultánea. En este contexto, existen tres posibles combinaciones en donde una organización, se puede ver involucrada como componente de una cadena de suministro:

- En el caso A una organización del tipo proveedor, participa en una CS con cada cliente o grupo de clientes.
- En el caso B una organización del tipo cliente, conforma una CS con cada proveedor o grupo de proveedores.
- En el caso C, tanto los proveedores como los clientes, se encuentran en más de una CS con otros clientes y proveedores respectivamente.

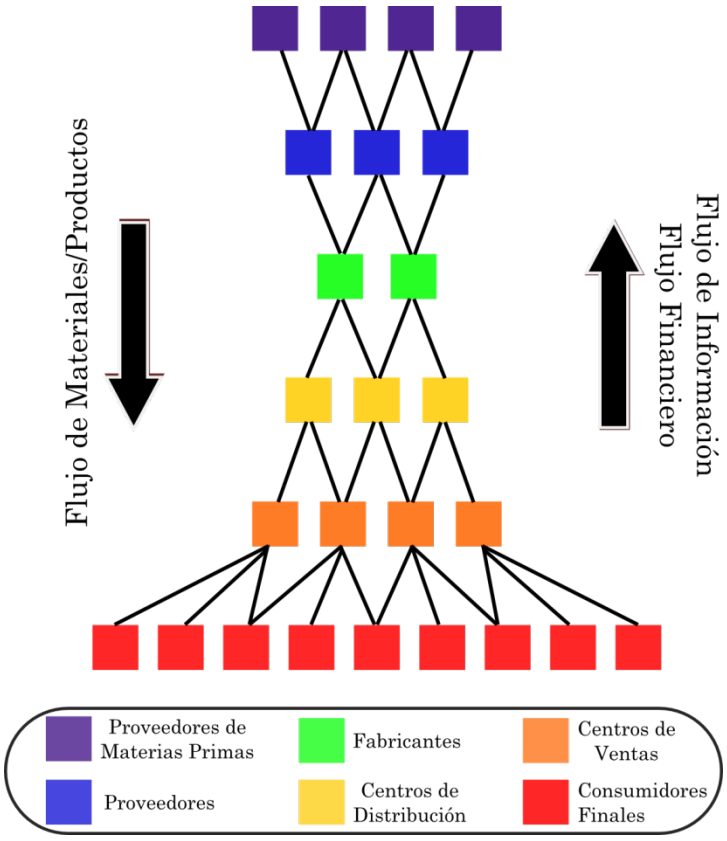


Figura 2.1 – Estructura de Cadena de Suministro

La Figura 2.2 esquematiza los casos A, B y C descriptos. Puede observarse en la misma que los participantes de la cadena se representan con cuadriláteros blancos, las interacciones entre las organizaciones se presentan mediante arcos bidireccionales, y las distintas cadenas de suministro se exhiben por medio de áreas de color.

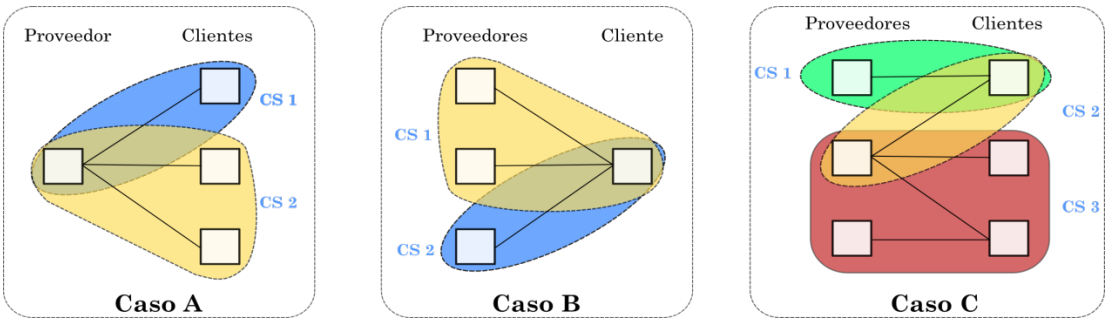


Figura 2.2 – Casos de Colaboración en Cadenas de Suministro

Como se puede observar, los escenarios de colaboración pueden ser muy variados ya que los objetivos de una cadena de suministro, los miembros de la red y los flujos de recursos suelen modificarse en el tiempo. De este modo, resulta evidente la necesidad de disponer de herramientas para analizar distintos escenarios de colaboración, en donde sea posible evaluar el desempeño integral de la CS, como así también, los beneficios y costos implicados para cada organización participante. En este contexto, la simulación emerge como una herramienta fundamental para evaluar las distintas configuraciones o escenarios, con el fin de alcanzar el éxito en la cadena de suministro (Bocciarelli et al. 2017; Tako y Kotiadis 2015).

Para hacer uso de la información cuantitativa obtenida como resultado de una simulación, es necesario construir modelos de simulación de los procesos y las interacciones, involucradas en la cadena de suministro a analizar. En otras palabras, es necesario definir un modelo que represente la estructura de la CS, así como también el flujo de recursos. Entonces, con el objetivo de especificar un modelo de CS para su posterior simulación, en la siguiente sección se analiza el modelo de referencia SCOR, el cual se utiliza en el desarrollo de esta tesis para cumplir con el objetivo mencionado.

2.1.1 Modelo de Referencia de Operaciones de Cadena de Suministro

El Supply Chain Council es un consorcio mundial sin fines de lucro, que desarrolló el modelo SCOR para brindar herramientas metodológicas, diagnósticas, y de evaluación comparativa, con el fin de realizar grandes y rápidas mejoras en los procesos de CS. Uno de los objetivos de este modelo, es permitir la evaluación de las actividades y la comparación del rendimiento de las cadenas de suministro. El modelo SCOR reúne la opinión consensuada de

todas las organizaciones integrantes del SCC, en cuanto a la gestión de la cadena de suministro. Este modelo proporciona un marco único de referencia, que vincula procesos comerciales, métricas, mejores prácticas, y tecnología, en una estructura unificada para respaldar la comunicación entre los socios de la cadena. De este modo, se pretende mejorar la efectividad tanto de la gestión, como de las actividades relacionadas a la mejora de la CS.

El SCC se creó en el año 1996 e inicialmente incluía 69 empresas profesionales, las cuales se reunían en un consorcio informal. Posteriormente, las empresas del SCC eligieron formar una asociación comercial independiente sin fines de lucro. La suscripción al SCC está abierta a todas las compañías y organizaciones, interesadas en hacer uso y avanzar en el estado del arte, en los sistemas, y las prácticas de gestión de la CS. La mayoría de los miembros del SCC son empresas profesionales, que representan una amplia gama de industrias, como son los: fabricantes, distribuidores y minoristas. Además de las organizaciones industriales, existen otros tipos de miembros, como son los: proveedores de tecnología e implementadores, las instituciones académicas, y las organizaciones gubernamentales. Estos miembros participan en las actividades del SCC para el desarrollo y mantenimiento del modelo. Las revisiones del modelo se realizan cuando los miembros del SCC, determinan que se deben hacer cambios para facilitar el uso del modelo en la práctica.

El SCC está interesado en diseminar, de la forma más amplia posible, el modelo SCOR. El uso generalizado del modelo, da como resultado: mejores relaciones cliente-proveedor, sistemas de software que utilizan medidas y términos comunes, para brindar un mejor soporte a los miembros; además de poseer la capacidad de reconocer y adoptar rápidamente, las mejores prácticas sin importar dónde se originen.

Para el desarrollo de esta tesis se utiliza la versión 11.0 del modelo SCOR (Supply Chain Council 2012), que es la decimotercera revisión desde la introducción del modelo en 1996. A fines del año 2017 se anunció el lanzamiento de la versión 12.0 del modelo SCOR (Supply Chain Council 2017), la cual incluye nuevas características asociadas a los tópicos emergentes de: *block chain*, cambios en las mejores prácticas para alinearse con la estrategia digital, actualización de las habilidades requeridas, estándares de sustentabilidad, y

generación de los flujos de trabajo de los procesos. En las siguientes secciones, se describe en detalle los distintos aspectos del modelo SCOR en su versión 11.0.

2.1.1.1 Alcance

El modelo SCOR se ha desarrollado para describir las actividades comerciales, asociadas con todas las fases de satisfacción de la demanda de un cliente. El modelo en sí contiene varias secciones, y se encuentra organizado en torno a los seis procesos de gestión principales: *Plan* (Planear), *Source* (Abastecer), *Make* (Producir), *Deliver* (Entregar), *Return* (Devolver) y *Enable* (Gestionar). Estos procesos se centran en el flujo de materiales e información entre las organizaciones de la CS, los mismos se presentan en la Figura 2.3.

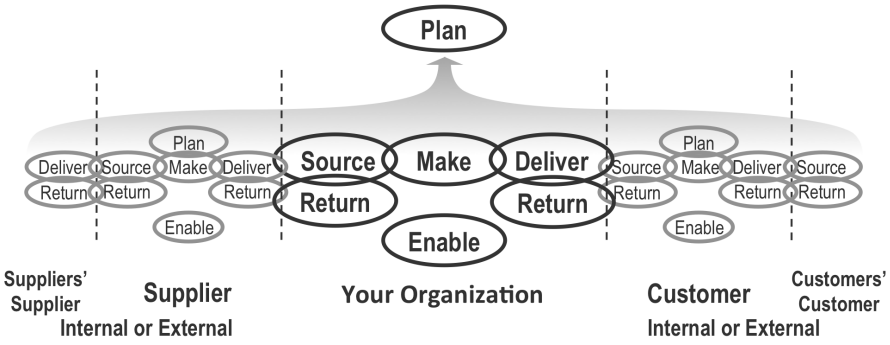


Figura 2.3 – Organización horizontal de los procesos de gestión en SCOR

(Obtenida de (Supply Chain Council 2012))

Al describir las cadenas de suministro usando estos bloques de construcción de procesos, el modelo se puede emplear para describir cadenas de suministro desde muy simples hasta muy complejas, a partir de un conjunto común de definiciones. Como resultado, las industrias dispares pueden vincularse para describir la profundidad y amplitud, de prácticamente cualquier CS. El modelo de referencia ha sido capaz de proporcionar y describir con éxito, una base para la mejora de cadenas de suministro tanto para proyectos globales, como para proyectos específicos de un sitio.

Dentro de los alcances del modelo SCOR, se encuentran: todas las interacciones con los clientes (entrada de pedidos a través de la factura pagada), todas las transacciones de material

físico (proveedor del proveedor hasta el cliente del cliente, incluyendo los equipos, suministros, repuestos, productos a granel, software, entre otros), y todas las interacciones del mercado (desde la comprensión de demanda agregada al cumplimiento de cada orden). El modelo de referencia, no intenta describir cada proceso o actividad comercial en detalle. Específicamente, se encuentran fuera del alcance del modelo SCOR los procesos de: ventas y marketing (generación de demanda), desarrollo de productos, investigación y desarrollo, y algunos elementos de soporte al cliente posterior a la entrega. Cabe señalar que, el alcance del modelo ha cambiado desde su versión previa y, además, se prevé que cambie según los requerimientos de los miembros del SCC. Con la introducción del proceso devolver (*Return*), el modelo SCOR se extendió al área de atención al cliente posterior a la entrega (aunque no incluye todas las actividades de esa área).

Como se muestra en la Tabla 2.1, el modelo de referencia se encuentra diseñado para soportar el análisis de la cadena de suministro, en múltiples niveles. El SCC se ha enfocado en los tres niveles principales de procesos, los cuales son independientes del dominio de la industria. El modelo SCOR no intenta prescribir, cómo una organización en particular debe llevar a cabo sus actividades o adaptar su sistema/flujo de información. Toda organización que implemente mejoras en la cadena de suministro por medio del uso del modelo SCOR, deberá extender el modelo, al menos al Nivel 4. Para realizar esta extensión al modelo, es necesario utilizar procesos, sistemas y prácticas específicos de la industria, de la organización y/o de la ubicación.

Tabla 2.1 – Organización jerárquica de los procesos en SCOR

	Nivel #	Nombre	Descripción
Dentro del alcance de SCOR	1	*Alcance*	Define los tipos de procesos, alcance y contenido de la CS
	2	*Configuración*	Describe las categorías de procesos y la estrategia de operaciones
	3	*Pasos*	Establece los elementos de proceso y la configuración de los procesos individuales
Fuera del alcance	4	*Actividades*	Especifica las actividades desarrolladas dentro de la CS

Es importante observar que este modelo describe procesos, no funciones. En otras palabras, el modelo se centra en la actividad involucrada, no en la persona o el elemento organizativo que realiza la actividad.

2.1.1.2 Estructura

El modelo SCOR es un modelo de referencia y, como todo modelo de referencia, tiene como propósito describir la arquitectura de su proceso, de una manera significativa y clara para los socios clave a nivel comercial. La arquitectura, en este contexto, significa la forma en que los procesos interactúan, funcionan, se configuran y los requisitos (habilidades) en el personal que opera el proceso.

El modelo de referencia SCOR consta de 5 secciones principales:

- **Rendimiento**: Métricas estándar para describir el rendimiento de los procesos y definir objetivos estratégicos.
- **Procesos**: Descripciones estándar de procesos de gestión y relaciones de procesos.
- **Prácticas**: Prácticas de gestión que producen un mejor rendimiento de procesos.

- **Personas**: Definiciones estándar de las habilidades requeridas para realizar los procesos de cadena de suministro.

- **Aplicaciones Especiales**: Se utiliza para sugerir adiciones al modelo SCOR que aún no se han probado exhaustivamente, pero que el SCC considera que serían beneficiosas para los usuarios de SCOR.

A continuación se describen los apartados de evaluación del rendimiento y procesos, los cuales son los más relevantes para el trabajo que se desarrolla, en el marco de esta tesis.

2.1.1.3 Evaluación del Rendimiento

La sección de rendimiento de SCOR consta de dos tipos de elementos: atributos y métricas de rendimiento. Un atributo de rendimiento es una agrupación de métricas, utilizadas para expresar una estrategia. Un atributo en sí mismo no se puede medir; solo se utiliza para establecer una dirección estratégica. Algunos ejemplos de las estrategias comerciales que se pueden aplicar a la cadena de suministro son: "desempeño superior para la confiabilidad de la cadena de suministro" o "rendimiento avanzado para la agilidad". Las métricas del modelo SCOR miden la capacidad de una CS, para lograr estos atributos estratégicos. Por lo tanto, un rendimiento superior para la confiabilidad se puede expresar en un objetivo como: "cumplir de forma perfecta con el 80% de las ordenes entregadas". En donde, el atributo de rendimiento es la Confiabilidad (*Reliability*), la métrica de rendimiento es Cumplimiento Perfecto de Orden (*Perfect Order Fulfillment*) y el valor porcentual se calcula en base a técnicas de benchmarking

Atributos de Rendimiento

Cada atributo tiene asociada una o más métricas de nivel uno, las cuales se denominan *indicadores clave de rendimiento* KPI (por sus siglas en inglés *–Key Performance Indicator-*) para el atributo. El modelo SCOR reconoce cinco atributos de rendimiento, los cuales son:

1. Confiabilidad RL (por sus siglas en inglés *–Reliability-*): Este atributo aborda la capacidad de realizar tareas según sea necesario, y se centra en la previsibilidad del resultado de un proceso. Las métricas típicas para el atributo de confiabilidad incluyen: a tiempo, la

cantidad correcta, la calidad correcta, entre otras. La confiabilidad es un atributo centrado en el cliente, cuya métrica asociada de nivel uno es el cumplimiento perfecto de orden.

2. Capacidad de Respuesta RS (por sus siglas en inglés -*Responsiveness*-): Este atributo describe la velocidad a la que se realizan las tareas y, además, mide la velocidad de la CS para entregar productos a los clientes. Las métricas típicas para el atributo capacidad de respuesta, son aquellas que miden ciclos de tiempo. La capacidad de respuesta es un atributo centrado en el cliente, cuyo indicador clave de rendimiento es el ciclo de tiempo para el cumplimiento de pedidos.

3. Agilidad AG (por sus siglas en inglés –*Agility*-): Este atributo describe la habilidad de responder a las influencias externas; en cuanto a la capacidad y velocidad del cambio. Las influencias externas incluyen: aumentos o disminuciones no previsibles de la demanda, proveedores o socios que cierran sus negocios, desastres naturales, actos de (ciber) terrorismo, disponibilidad de recursos financieros (la economía), problemas laborales, entre otros. La agilidad es un atributo centrado en el cliente, cuyas métricas de nivel uno son flexibilidad, adaptabilidad y valor total del riesgo.

4. Costo CO (por sus siglas en inglés –*Cost*-): Este atributo describe el costo de operar los procesos de CS. Los costos típicos incluyen: mano de obra, material, transporte, y distribución. El costo es un atributo centrado en la estructura interna de la CS, cuyo indicador clave de rendimiento es el costo total a pagar.

5. Eficiencia en la Gestión de Activos AM (por sus siglas en inglés –*Assets Management Efficiency*-): El atributo eficiencia en la gestión de activos, también denominado *activos*, describe la capacidad de utilizar activos de manera eficiente. Las estrategias de gestión de activos en las cadenas de suministro, incluyen la reducción de inventario, y producción frente a subcontratación. Las métricas típicas para el atributo de activos incluyen: días de suministro y utilización de la capacidad, entre otros. Los activos son un atributo centrado en la estructura interna de la CS, cuyos indicadores clave de rendimiento son: ciclo de tiempo efectivo a efectivo, retorno sobre el capital de trabajo, y rendimiento de los activos de cadena de suministro fijos.

En la Figura 2.4 se presenta el vínculo existente entre los atributos de rendimiento y las métricas de nivel uno, además se especifica en qué se centra cada atributo.

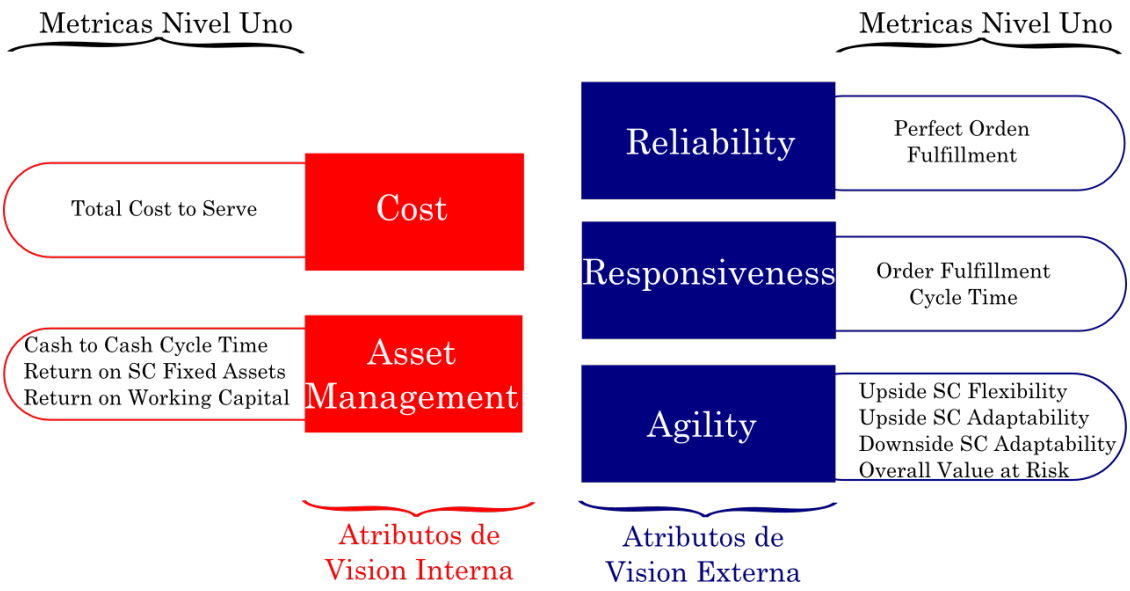


Figura 2.4 – Vínculo entre atributos y métricas de rendimiento en SCOR

Los atributos centrados en el cliente otorgan una visión externa a la CS, sobre la satisfacción de los clientes finales. En cambio, los atributos centrados en la estructura de la CS presentan una visión interna de la cadena, sobre los procesos inter organizacionales. Cada atributo de rendimiento tiene una o más métricas estratégicas de nivel 1, mediante las cuales una organización puede medir su nivel de éxito, para lograr el posicionamiento deseado dentro del mercado competitivo. Cabe destacar que todas las métricas del modelo SCOR, se agrupan dentro de alguno de los atributos de rendimiento.

Métricas

Una métrica es una forma estándar de medición, que se utiliza para medir el rendimiento de una cadena de suministro o proceso. Las métricas de SCOR son métricas utilizadas para un análisis diagnóstico. El modelo SCOR reconoce tres niveles jerárquicos de métricas:

- Las métricas de nivel 1, son diagnósticos para el estado general de la CS. Estas métricas también, se conocen como métricas estratégicas o KPI. Las métricas de benchmarking de nivel 1, ayudan a establecer objetivos concretos para respaldar las direcciones estratégicas.

- Las métricas de nivel 2, sirven como diagnósticos para las métricas de nivel 1. La relación de diagnóstico, ayuda a identificar la o las causas que producen una brecha de rendimiento, para una métrica de nivel 1.
- Las métricas de nivel 3, sirven como diagnósticas para las métricas de nivel 2.

El análisis del rendimiento de las métricas del nivel 1 al 3, se conoce como descomposición de métricas o diagnóstico del rendimiento. La descomposición de las métricas, es un primer paso para identificar los procesos que requieren mayor investigación, dado que los procesos se encuentran vinculados a las métricas.

Para identificar con mayor facilidad a las métricas, en la versión 9.0 del modelo SCOR se introdujo la codificación de las mismas. De esta forma, se asegura a las organizaciones que pueden adoptar las métricas del modelo, sin la necesidad de renombrar sus métricas existentes. La codificación de las métricas sigue el formato definido en la Ecuación 2.1:

$$Codificación\ Metrica = [atributo].[nivel\ métrica].[id\ único] \quad \text{(Ecuación 2.1)}$$

En donde:

- **Atributo** = Es el identificador del atributo de rendimiento al que pertenece la métrica. Sus posibles valores son: RL (para confiabilidad), RS (para capacidad de respuesta), AG (para agilidad), CO (para costo) y AM (para activos).
- **Nivel métrica** = Este identificador representa el nivel asociado de la métrica. Sus posibles valores son: 1, 2 ó 3.
- **Id único** = El valor de este identificador, se utiliza para identificar unívocamente a una métrica en un nivel dado, para un atributo de rendimiento específico. Este campo comienza la secuencia en uno, y no marca precedencia entre las métricas, por lo que se utiliza solo a efectos de distinguir las mismas.

Cada código de una métrica debe contener un valor para cada uno de los tres campos (atributo, nivel métrico e id único), que componen la codificación de la métrica. En el Anexo A, se presenta un detalle completo de la descomposición de todas las métricas contenidas, en la versión 11 del modelo SCOR.

El SCC recomienda que las tablas de puntuación de la cadena de suministro, contengan al menos una métrica por cada atributo de rendimiento, para garantizar la toma de decisiones equilibrada y la gobernanza.

2.1.1.4 Procesos de Negocio

Un proceso es una actividad única, realizada para cumplir con los resultados esperados. Los procesos de SCOR proveen un conjunto de descripciones predefinidas, para las actividades que se requieren ejecutar en la mayoría de las organizaciones de una CS, con el objetivo principal de cumplir con los pedidos de los clientes. El modelo SCOR posee una única representación, para cada uno de los seis principales procesos de gestión de la CS (procesos de nivel 1), los cuales son:

- ***Plan*** **(Planear)**: Se enfoca en administrar y programar el suministro, la gestión del inventario, y la planificación de la demanda, entre otras cosas. En este proceso, se deben equilibrar los recursos disponibles con los requerimientos necesarios.

- ***Source*** **(Abastecer)**: Se centra en la infraestructura de abastecimiento y adquisición de materiales. Este proceso involucra las actividades de: adquisición de bienes y servicios para satisfacer la demanda de materias primas, componentes y otros servicios necesarios para la producción, y las actividades involucradas con la recepción de los materiales.

- ***Make*** **(Producir)**: Es un proceso de negocio interno de la organización, que cubre todas las áreas productivas y operacionales de la misma. Además contiene las actividades de producción y empaque, las cuales transforman los insumos en productos finales listos para ser despachados.

- ***Deliver*** **(Entregar)**: Contiene todas las actividades asociadas a la entrega de los productos elaborados, a los clientes finales.

- ***Return*** **(Devolver)**: Este proceso gestiona el retorno de insumos o materias primas a los proveedores, así como la recepción de los productos devueltos por los clientes. Es un proceso basado en la logística inversa.

- ***Enable*** **(Gestionar)**: Es el proceso de negocio enfocado en gestionar las reglas del negocio, el rendimiento de la CS, la recolección de datos e información, los recursos humanos, los activos, los contratos, los escenarios de configuración de CS, las finanzas y el riesgo.

Esta estructura de negocios, es la estructura horizontal de los procesos. Además de la misma, los procesos poseen una estructura vertical a través de niveles jerárquicos. Los distintos niveles, se describen a continuación.

- *Nivel de Alcance: Tipos de Procesos*

En este nivel se identifican todos los vínculos de suministro en los que participa la empresa, tanto con proveedores como con clientes. Además, se define la estrategia de gestión de la CS, la cual se planifica a largo plazo puesto que implica decisiones estructurales. Como es una estrategia de largo plazo, se consideran elementos de incertidumbre y variaciones. En este nivel se modelan los activos, los tipos de productos y su volumen, los requerimientos tecnológicos, las limitaciones, y todos los objetivos de rendimiento a alcanzar

- *Nivel de Configuración: Categorías de Procesos*

En este nivel se planifica la estrategia de operaciones de la cadena de suministro. A su vez, se define la configuración de la planificación y la ejecución del flujo de materiales, en donde los procesos de negocio se configuran de acuerdo con la estrategia de la CS. Además se definen las capacidades de la cadena, y se utilizan herramientas de pronóstico para estimar el posicionamiento de la misma en su entorno comercial. Este posicionamiento se utiliza con el propósito de maximizar las ganancias potenciales, durante el período de tiempo a analizar. Las limitaciones del mercado, las limitaciones y restricciones de los productos finales, son consideradas para configurar las categorías de procesos de negocio internos, y procesos de negocio colaborativos de los que participará la organización.

- *Nivel de Pasos: Elementos de Procesos*

Este nivel permite a las empresas configurar en detalle los procesos de negocio individuales, los parámetros de rendimiento, las mejores prácticas, las capacidades tecnológicas necesarias para ejecutar el proceso, las entradas y salidas, así como también las

habilidades necesarias de los recursos humanos para realizar las actividades del proceso. En este nivel se definen un conjunto de políticas operativas para el corto plazo.

Al igual que las métricas, los procesos poseen una codificación para su identificación de forma fácil y sencilla. En el caso de los procesos, la codificación depende del nivel asociado de los mismos. De este modo, los procesos de nivel uno respetan la codificación definida en la Ecuación 2.2, mientras que los procesos de nivel dos se identifican de acuerdo con la codificación definida en la Ecuación 2.3 y, por último, los procesos de nivel tres se representan por medio de la codificación definida en la Ecuación 2.4.

$$Codificación\ Nivel\ 1 = s[id\ proceso] \quad \text{(Ecuación 2.2)}$$

$$Codificación\ Nivel\ 2 = s[id\ proceso][id\ categoría] \quad \text{(Ecuación 2.3)}$$

$$Codificación\ Nivel\ 3 = s[id\ proceso][id\ categoría].[id\ único] \quad \text{(Ecuación 2.4)}$$

En donde:

- **s** = Representa que el proceso es un proceso de SCOR.
- **Id proceso** = Es el identificador del tipo de proceso. Sus posibles valores son: P (para planear), S (para abastecer), M (para producir), D (para entregar), R (para devolver) y E (para gestionar).
- **Id categoría** = Es un identificador numérico de la categoría del proceso. El identificador comienza la secuencia en uno y no marca precedencia entre los procesos, sólo se utiliza solo a efectos de identificar unívocamente los procesos.
- **Id único** = Es un número que identifica a un proceso en un nivel dado. El identificador comienza en la secuencia uno y no marca precedencia entre los procesos, sólo se utiliza solo a efectos de identificar unívocamente los procesos.

Las únicas excepciones a esta codificación son los procesos del tipo *devolver*. Para el nivel dos de estos procesos, se establece una división entre las devoluciones a los proveedores (en donde el id proceso = SR) y las recepciones de productos devueltos por los clientes (en donde el id proceso = DR). A su vez los procesos de nivel tres del tipo devolver, también siguen esta codificación.

2.1.1.5 Ventajas del Modelo

Las principales ventajas que presenta el modelo SCOR se pueden resumir en:

- Es un modelo que presenta una metodología para la identificación, diseño, análisis, y evaluación de las operaciones de la cadena de suministro.
- Actualmente no existe otra metodología similar, que abarque la operación de la CS desde el proveedor del proveedor hasta el cliente del cliente de la organización.
- Por medio del uso de esta metodología, se crea conciencia en la empresa respecto a la importancia de la mejora de la cadena de suministro.
- El modelo SCOR se utiliza como lenguaje común de comunicación, entre las organizaciones participantes de la CS.
- Define un formato estándar, el que facilita el flujo de información.
- Presenta una organización sistemática de los procesos de negocio, que se deben llevar a cabo, para una buena gestión de la cadena de suministro.

En conclusión, el modelo SCOR ofrece una buena aproximación que cubre todos los aspectos necesarios para mejorar la gestión de la cadena de suministro. Este modelo proporciona un conjunto de herramientas para, de forma rápida, modelar, comprender y evaluar la cadena, a efectos de identificar oportunidades de mejora para la misma.

2.2 Simulación Paralela y Distribuida

La simulación distribuida es un campo que tiene sus raíces en la computación paralela y distribuida. La misma ha contribuido al éxito en la simulación de grandes sistemas en distintos ámbitos, como son: la defensa, el diseño de sistemas de computadora, los ambientes de ciudades inteligentes, y las cadenas de suministro. Según lo menciona (Richard M. Fujimoto 2016; 1990), la SD emerge de dos comunidades. Por un lado, en los ´70 la comunidad de simulación paralela de eventos discretos PDES (por sus siglas en inglés –

Parallel Discrete Event Simulation-) investigaba cómo incrementar la velocidad en la ejecución de una simulación, utilizando múltiples procesadores en sistemas de computadoras de alto rendimiento. La motivación para realizar esta investigación surgía dada la necesidad de ejecutar grandes sistemas de simulación, con tiempos de ejecución excesivos o que no podían ejecutarse dados los recursos de computadora (memoria, procesamiento, y almacenamiento principalmente) existentes en ese momento. Por otro lado, en los ’80, principalmente debido a esfuerzos del sector de defensa para fomentar la reutilización de simulaciones, la comunidad de simulación distribuida utilizó técnicas de PDES para interconectar simulaciones en una sola gran simulación, ejecutada sobre computadoras distribuidas geográficamente, y conectadas mediante una red de computadoras. En la actualidad, los investigadores de estas áreas llaman informalmente al campo simulación paralela y distribuida PADS (por sus siglas en inglés –*Parallel and Distributed Simulation*-). Sin embargo, dado que inicialmente la simulación paralela se ejecutaba en máquinas con múltiples procesadores, pero con el tiempo se comenzaron a ejecutar en distintas máquinas distribuidas geográficamente, para la comunidad de simulación el campo de PADS se denomina simplemente como SD. Según (R. Fujimoto 2015; D’Angelo y Marzolla 2014) las diferencias entre simulación paralela y SD se hacen cada vez menos claras, dado el surgimiento de los clúster de estaciones de trabajos, el concepto de *grid* y *cloud computing*.

Lo cierto es que generalmente no se hace una diferencia entre simulación paralela y distribuida, y estos términos se pueden encontrar usados indistintamente. Por este motivo, a lo largo de esta tesis se emplea el término simulación distribuida como: *la ejecución simultánea de simulaciones, que se encuentran en máquinas distribuidas geográficamente, y admiten eventos concurrentes* (S. J. E. Taylor 2019). Cabe destacar que, en el marco de esta definición de SD el término simulaciones hace referencia a: *varios simuladores, cada uno con un modelo de simulación que representa el comportamiento a simular*. Por lo tanto, el término simulación se refiere a: *un simulador con un modelo de simulación que representa el comportamiento a simular*.

La SD utiliza las técnicas de computación paralela y distribuida, además de múltiples computadoras, con los siguientes objetivos principales: para incrementar la velocidad en la ejecución de un programa de simulación, y/o para interconectar simulaciones en una sola

simulación a efectos de fomentar el reúso de simulaciones (R.M. Fujimoto 2000; S. J. E. Taylor 2019). Con respecto a este último objetivo, en donde se plantea el reúso de simulaciones, algunos ejemplos de sus aplicaciones son: entrenamientos militares donde las simulaciones de tanques de guerra, aviones, infantería, generadores de fuertes, y otra variedad de simulaciones, integran una sola gran simulación. Dicha simulación se torna en un entorno virtual distribuido, donde las personas pueden entrenar y es posible plantear situaciones hipotéticas (McIntyre, Smith, y Goode 2013; Zhang y Zhang 2013). Además de estos ejemplos de aplicación, la SD comienza a utilizarse con más frecuencia en el área de cadenas de suministro, de fábricas de manufactura, en ámbitos tanto sociales como naturales, y para predecir o analizar catástrofes (Long 2014; Zehe et al. 2015).

Si bien la SD tiene sus ventajas con relación a la simulación local, existen varios problemas que deben ser tenidos en cuenta a la hora de hacer uso de la misma. Entre ellos podemos mencionar dos grandes problemas: uno está relacionado a la sincronización de los simuladores, y el otro se refiere a la integración de los mismos para componer una sola simulación total o de mayor nivel. El primer problema es también conocido como problema de administración del tiempo, y se refiere a la necesidad de que la simulación distribuida produzca exactamente los mismos resultados, que si es ejecutada localmente. El segundo problema, es la interpretación realizada por un simulador de los eventos enviados por otro simulador. De forma tal que, estos eventos provoquen los efectos esperados por quien diseña el modelo de simulación ejecutado en el otro simulador. Este problema es conocido como de interoperabilidad entre simuladores.

Con relación al manejo del tiempo, se propusieron diferentes esquemas de administración del mismo los cuales se conocen como: esquema conservador, esquema optimista y esquema de tiempo real. Con relación a la interoperabilidad, la temática es abordada por el estándar de la IEEE denominado HLA (IEEE 2010a), el cual es ampliamente usado por la mayoría de las soluciones propuestas de simulación distribuida. Este estándar tiene su origen en el estándar DMSO HLA 1.3 (DOD 1995), que luego fue evaluado y estandarizado por el organismo de estandarización IEEE, dando origen a una nueva versión conocida como el estándar 1516-2010 HLA. Este último estándar, provee de interoperabilidad sintáctica a los simuladores, pero carece de interoperabilidad semántica.

El desarrollo de esta tesis se enfoca en la problemática de interoperabilidad, y específicamente en el nivel semántico de la misma. Por este motivo, en la próxima sección se presenta una introducción a la última versión del estándar 1516-2010 HLA de la IEEE.

2.2.1 Estándar IEEE 1516-2010 High Level Architecture Evolved

HLA es un *framework* de simulación distribuida propuesto por el departamento de defensa de los Estados Unidos DMSO (por sus siglas en inglés -*Defense Modelling and Simulation Office*-), el cual definió una arquitectura para el modelado y simulación de sistemas complejos. Con el paso de los años HLA fue adoptado por la IEEE para su estandarización, dando origen al estándar IEEE 1516-2000 en el año 2000, y luego evolucionando al estándar IEEE 1516-2010 en el año 2010. La arquitectura de HLA es un estándar, que soporta el reúso de las capacidades disponibles en los diferentes simuladores. Ademas, posibilita el desarrollo de sistemas de simulación complejos de manera cooperativa y distribuida, soportando el desarrollo de una simulación basada en componentes, donde los componentes se denominan federados.

La arquitectura de HLA provee un *framework* dentro del cual los desarrolladores de simuladores, pueden estructurar y describir sus aplicaciones de simulación. En este sentido, HLA propone integrar sistemas de simulación diferentes, cada uno con un objetivo determinado, en un sistema mayor que represente el comportamiento del mismo. Además, se pretende que los componentes se utilicen sin necesidad de reescribirlos, o comenzar a crear desde cero el modelo de mayor nivel y su simulador. En HLA, un federado *es un simulador que cumple con las especificaciones de HLA y es integrado al framework*. Luego, los federados se unen en federaciones. El término federación, se utiliza para hacer referencia al *simulador del sistema complejo, conformado por los simuladores de menor nivel, llamados federados, los cuales integran la federación*. La definición de HLA abarca tres componentes esenciales, los cuales son: reglas, especificación de la interfaz y patrón del modelo de objetos.

Las reglas son un conjunto de ítems que definen las responsabilidades, y relaciones entre los componentes de una federación HLA. Las mismas deben ser seguidas por los

federados y las federaciones, que quieran ajustarse al estándar. Seguir estas reglas asegura una interacción correcta entre los simuladores de una federación.

La especificación de la interfaz, define la interfaz funcional entre los federados HLA y la infraestructura de tiempo de ejecución RTI (por sus siglas en inglés -*Runtime Infrastructure*-) de HLA. En la misma se detallan los servicios que debe prestar el RTI, e identifican los servicios que debe proveer cada federado. El RTI ofrece una librería de programación y una interfaz de programación de aplicaciones API (por sus siglas en inglés –*Application Program Interface*-), las cuales son compatibles con la especificación de la interfaz. El RTI puede ser visto como el sistema operativo distribuido, que provee comunicación y coordinación entre los federados.

El patrón del modelo de objetos OMT (por sus siglas en inglés –*Object Model Template*-), presenta un mecanismo para especificar el modelo de datos. Este modelo contiene la información producida y consumida por los elementos de la SD. De un modo más formal, el OMT describe un método común para declarar la información, que será producida y comunicada, por cada simulador participante en la simulación distribuida. Una SD basada en HLA, utiliza como formato común para definir los modelos de datos, los siguientes modelos: el FOM, el modelo de objetos de simulación SOM (por sus siglas en inglés -*Simulation Object Model*-) y el modelo de objetos de gestión MOM (por sus siglas en inglés -*Management Object Model*-). El FOM describe los objetos, los atributos y las interacciones de la federación completa; en tanto que, el SOM describe los objetos, atributos e interacciones utilizados por un federado, en todas las federaciones en las que participa. Por otro lado, el MOM provee facilidades para acceder a la información operativa del RTI, en tiempo de ejecución. El mismo se debe definir utilizando objetos, interacciones y constructores de la arquitectura de comunicación de HLA.

En la Figura 2.5 se esquematizan los componentes descriptos anteriormente, de la arquitectura de HLA.

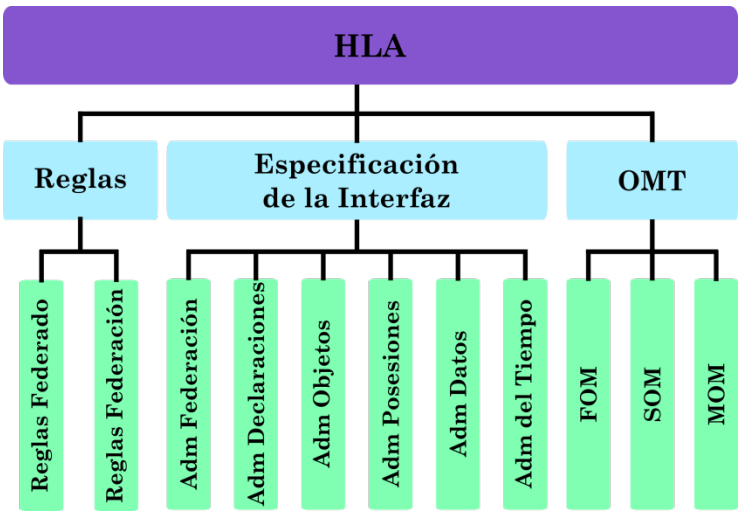


Figura 2.5 – Componentes de la arquitectura HLA

Cada federado tiene su modelo de objetos SOM, que describe los datos que un federado puede producir o consumir. Cada federación tiene un modelo de objetos propio denominado FOM, el cual especifica las partes comunes de los SOM utilizados en la federación, y que se corresponden con los federados integrantes de la federación. El RTI consiste básicamente en una implementación de la especificación de la interfaz, compuesta por un conjunto de servicios. Además, es considerado como el software necesario y fundamental para ejecutar la federación. El RTI provee las siguientes funcionalidades: comenzar y terminar la ejecución de la simulación, transferir datos entre los simuladores intervinientes, controlar la cantidad de datos que son transferidos, y finalmente coordinar el paso del tiempo de simulación entre los simuladores HLA. Este software se encuentra fuera del alcance del estandar HLA, y es suministrado por proveedores de software específicos.

El estándar 1516-2010 se encuentra dividido en dos partes:

- 1516.1-2010 IEEE standard for modeling and simulation high level architecture
  - *Federate Interface Specification*
- 1516.2-2010 IEEE standard for modeling and simulation high level architecture
  - *Object Model Template Specification*

El documento denominado *Federate Interface Specification*, describe la interfaz entre el federado y el servicio de software subyacente que soporta la comunicación entre los

federados, en un dominio de simulación distribuido. La especificación de la interfaz se encuentra dividida en las siguientes secciones: *federation management*, *declaration management*, *object management*, *ownership management*, *time management*, *data distribution management*, *support services*, *management object model* y *programming language mappings*. Cada uno de estos componentes describe detalladamente los servicios que un federado puede pedir al RTI, además de los servicios que el federado debe proveer al RTI (denominados *callback services*).

El documento denominado *Object Model Template Specification*, describe la sintaxis y el formato para representar la información de los modelos de objetos de HLA. Entre la información que incluye este documento, se destaca: objetos, atributos, interacciones, parámetros, dimensiones, representaciones de tiempo, tipos de sincronización, de transporte, de datos y frecuencias de actualización. En este documento se especifica, qué contenido deben poseer los documentos FOM, SOM y MOM. El FOM tiene como objetivo proveer una especificación común, para el intercambio de datos entre federados, en un formato estándar. En síntesis, el FOM establece un contrato del modelo de información que es necesario (pero no suficiente), para alcanzar la interoperabilidad entre los federados que integran la federación. El SOM especifica los tipos de información, que un federado individual podría proveer a la federación HLA, y la información que este podría recibir desde otro federado de la federación. El MOM provee facilidades para acceder a información operativa del RTI en tiempo de ejecución. Los federados usan estas facilidades, para tener un conocimiento detallado del funcionamiento de la ejecución de la federación. Además, por medio de estos servicios se puede controlar la ejecución del RTI, como también de la federación y de los federados individuales. El MOM se define utilizando objetos, interacciones y constructores de HLA predefinidos. Todas las clases de objetos, clases de interacciones, atributos y parámetros se definen en el archivo FDD (FOM *Document Data*).

La Figura 2.6 muestra una representación gráfica de una federación y los federados que participan en ella. En esta federación, denominada *Federación A*, intervienen tres federados llamados: *Federado A*, *Federado B* y *Federado C*. Estos federados comparten el mismo *FOM* y se comunican unos con otros a través del RTI. El componente *FedExec1*, corresponde a la ejecución de la federación, y es creado cuando se invoca el servicio *Create Federation*

Execution sobre el RTI. Este servicio puede ser invocado por cualquiera de los federados o aún por algún componente externo a la federación. Durante la ejecución del RTI varias federaciones pueden estar activas, y un federado puede estar en una o varias federaciones. Para ello, cada uno de los federados y las federaciones deben identificarse con un identificador único.

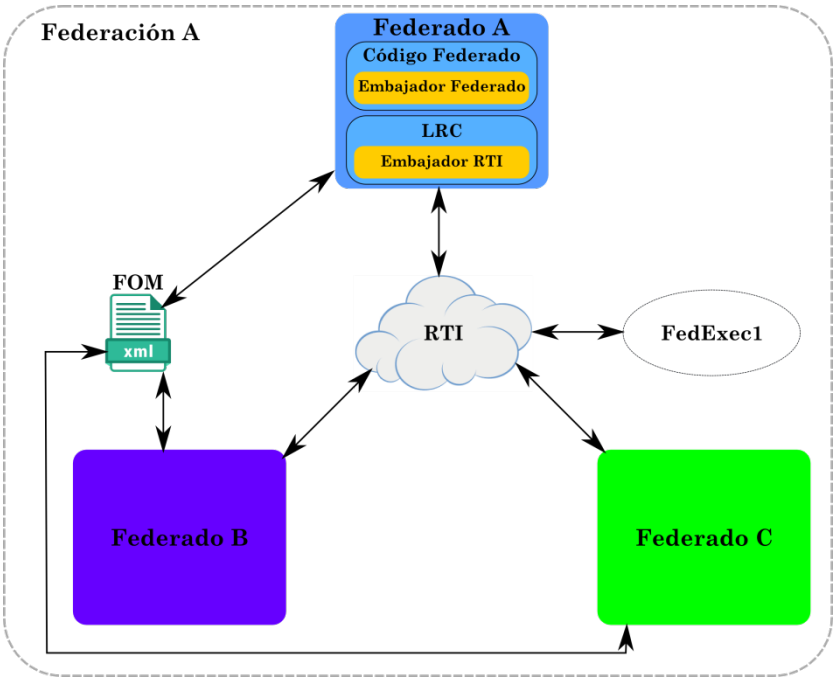


Figura 2.6 – Esquema de una federación de HLA

De acuerdo a la definición dada por HLA, un federado es una aplicación de software que puede ser o está actualmente acoplada, con otras aplicaciones de software bajo un FDD y un RTI. De este modo, como se puede observar en la Figura 2.6, la estructura interna de un federado está compuesta por dos elementos esenciales: el componente local de tiempo de ejecución LRC (por sus siglas en inglés -*Local Runtime Component*-) y el Código Federado (código que implementa la funcionalidad del modelo de simulación).

El componente LRC posee a su vez un subcomponente denominado *Embajador RTI*, que se encarga de la comunicación entre el RTI y el federado. Este componente debe invocar los servicios sobre el RTI, correspondientes a aquellas funciones definidas en el documento *Federate Interface Specification*, como pueden ser, por ejemplo: unir al federado a una federación, que un federado abandone una federación, suscribir al federado a una interacción,

suscribir al federado a un objeto, publicar interacciones, publicar objetos, actualizar atributos, solicitar permiso para avanzar el tiempo, entre los más comunes.

El componente Código Federado representa el modelo de la simulación llevada a cabo. Este componente posee un subcomponente denominado *Embajador Federado*, el cual implementa los *callback services* que serán invocadas por el RTI. Por ejemplo: cuando el RTI envía una interacción al federado, estará invocando funciones sobre el *Embajador Federado* (en este caso particular se trata del servicio *Receive Interaction* definido en la especificación), el cual debe tratar de una manera particular dicho requerimiento. Algunos ejemplos de la invocación de servicios son: permitir al federado avanzar el tiempo de simulación (*Time Advance Grant*), enviar una interacción (*Receive Interaction*), informar sobre la actualización en los atributos de un objeto (*Provide Attribute Value Update*), informar de que atributos es dueño el federado (*Inform Attribute Ownership*), entre otros.

De esta forma, todas las interacciones desde el RTI hacia el federado son manejadas por el *Embajador Federado*, mientras que todas las interacciones desde el federado hacia el RTI son controladas por el *Embajador RTI*.

Como se dijo anteriormente, el RTI es un componente de software que está fuera del alcance de la especificación HLA, pero que debe estar de acuerdo con la misma. El RTI provee los servicios de software necesarios, para soportar la ejecución de una simulación distribuida basada en HLA, debido a que:

- Brinda interoperabilidad y portabilidad.
- Permite separar simulación de comunicación.
- Facilita la construcción y destrucción de federaciones.
- Soporta la declaración y administración entre federados.
- Asiste en la administración del tiempo del federado.
- Provee comunicación eficiente a grupos lógicos de federados.

El RTI es un componente que permite a los federados de una federación interactuar entre sí. Pueden encontrarse distintas implementaciones de RTI, que cumplen con las especificaciones de HLA y que se encuentran certificadas por la IEEE. Entre ellas existen

varias versiones comerciales como es el software *PITCH RTI* (“Pitch Technologies” 2017) o herramientas de software libre como es *PoRTIco* (Pokorny, Fraser, y Burns 2016). Particularmente para el desarrollo de esta tesis, se selecciona el RTI llamado *PoRTIco* dado que es de uso libre, y de código abierto. Además, el mismo cuenta con la certificación de la IEEE, posee implementaciones en C++ y Java, y cumple con las especificaciones tanto del estándar HLA actual como de la versión anterior, de forma completa. Fue desarrollado por Tim Pokorny, Michael Fraser y Lance Burns. La elección de este software se hizo teniendo en cuenta, principalmente, que es una herramienta de software libre y código abierto. Además, las versiones comerciales de los RTI son costosas y no se tiene acceso al código fuente, el cual es necesario para analizar y entender el funcionamiento de las federaciones HLA. La característica de ser de código abierto, brinda la posibilidad de modificarlo para contribuir a su mejora y beneficiar a la comunidad de usuarios.

A continuación en las siguientes secciones, se realiza una breve descripción de cada uno de los principales aspectos, sobre la especificación de la interfaz de HLA.

2.2.1.1 Comunicación entre los Federados

Los federados en una federación no se comunican directamente unos con otros, sino que lo hacen a través del RTI. Existen dos formas en que los federados pueden comunicarse: (1) a través del envío de interacciones y (2) a través de la actualización de valores de atributos de objetos. Tanto las interacciones intercambiadas como los atributos actualizados, deben encontrarse definidos en el FOM. Como se indicó anteriormente, el FOM representa un contrato entre los federados, el cual establece qué datos van a ser compartidos durante la ejecución de la federación que integran. Cuando los federados se unen a la federación, se asume que éstos conocen y aceptan dicho contrato.

El primer tipo de comunicación, el envío de interacciones, es una comunicación que se produce entre dos federados (de la misma federación) a través del RTI. En este caso la comunicación no es persistente, esto quiere decir que una vez recibida la interacción, la misma es tratada por el federado receptor y luego descartada. Las interacciones pueden contener parámetros para el envío de información.

El segundo tipo de comunicación, se produce cuando se comparten atributos de objetos. Esta comunicación se considera una comunicación, entre objetos cuyos dueños son distintos federados. En este caso la comunicación es persistente, dado que el valor del atributo permanece, aún después que el federado haya procesado la información solicitada.

Los mecanismos para publicar y suscribir a interacciones y atributos, están descriptos en detalle en la sección *declaration management*, de la especificación de la interfaz. En cualquiera de los dos casos, se utiliza el mecanismo denominado publicar/suscribir, también conocido como *publish/subscribe* en inglés. Un federado puede subscribirse a las interacciones y a los atributos que le sean de interés, y publicar las interacciones y los atributos que serán generados como consecuencia de su actividad. Luego, cuando un federado publica una interacción, el RTI se encarga de enviarla a todos aquellos federados que se encuentren subscriptos a la misma. De igual manera sucede con los atributos: un federado que es dueño de un objeto, modifica un atributo de dicho objeto, y como consecuencia, el RTI informa a todos aquellos interesados de la actualización producida. Se debe notar que sólo el dueño del objeto, puede en cualquier momento modificar un atributo del objeto. Existen mecanismos que le permiten a un federado, solicitar al RTI la tenencia de un objeto, así como también liberarlo para que otro federado pueda obtener la tenencia. Estos mecanismos se encuentran definidos en la sección *ownership management*, de la especificación de la interfaz.

2.2.1.2 Administración del Tiempo

El tiempo de simulación en HLA, es representado como puntos sobre el eje del tiempo global de la federación. En un instante de la ejecución, un federado estará sobre un punto de este eje, referido como el tiempo del federado.

En HLA, como en otros simuladores, las anomalías se eliminan asignando una marca temporal o *time stamp* a cada evento, y se asegura que los eventos son entregados al federado en forma ordenada, de acuerdo a su marca temporal. Además, el servicio de administración del tiempo asegura a un federado, no recibir eventos correspondientes a su pasado. En otras palabras, los eventos que arriban al federado deben poseer marcas temporales mayores al

tiempo local de ese federado, y nunca menor a este tiempo. Un objetivo clave de HLA es lograr la interoperabilidad y el reúso de simuladores. Para soportar la interoperabilidad entre los simuladores, el servicio de administración del tiempo permite que cada federado, utilice el mecanismo de administración del tiempo interno que le resulte más conveniente. De este modo se logra combinar en la ejecución de una federación, los mecanismos de administración del tiempo. Un aspecto importante para la alcanzar la interoperabilidad, es la transparencia en la administración del tiempo, lo cual requiere que el mecanismo para administrar el tiempo interno de un federado no pueda ser visible por el resto de los federados.

De acuerdo a cómo cada federado administra el tiempo, es posible presentar la siguiente clasificación:

- Dirigido por Eventos (*Event Driven*): El federado procesa tanto los eventos locales, como los eventos externos generados por otros federados, y los ordena de acuerdo a sus marcas temporales. El tiempo del federado, típicamente avanza al tiempo de la marca temporal del evento procesado.
- Pasos del Tiempo (*Time Stepped*): El federado avanza el tiempo de simulación, en una cantidad fija llamada paso (*step*). El simulador no avanza a la próxima etapa, si no se realizaron todas las actividades de la etapa actual.
- Simulación de Eventos Discretos Paralelos (*Parallel Discrete Event Simulation*): Los federados que se ejecutan sobre sistemas de múltiples procesadores, deben ser sincronizados internamente usando protocolos optimistas o conservadores.
- Dirigido por el Tiempo del Reloj Real (*Wall Clock Time Driven*): Este mecanismo se utiliza para aquellos simuladores, donde su tiempo de simulación es derivado del tiempo del reloj real. El tiempo que demanda la simulación, debe coincidir con el tiempo que transcurre, cuando ese proceso se ejecuta en la realidad. Los federados que usan este mecanismo, no requieren necesariamente que los eventos sean procesados, en un orden de marca temporal estricto.

La administración del tiempo en una federación, debe ser realizada tanto por el federado, como por el RTI. Por un lado, cada federado se encarga de procesar los eventos internos y los eventos producidos por otros federados, los cuales ordena de acuerdo a la

marca temporal de cada uno de ellos. Por otro lado, el RTI se encarga de que cada federado no reciba eventos de su pasado. Esta tarea es llevada a cabo, por el servicio de administración del tiempo en HLA, a través de:

- Un servicio de entrega de mensajes en un orden de marca temporal TSO (por sus siglas en inglés -*Time Stamp Order*-).
- Un protocolo donde el federado explícitamente solicita el avance de su tiempo interno, y el RTI provee un permiso para realizar el avance, cuando puede garantizar que no arribarán eventos con tiempos menores.

La Figura 2.7 presenta los servicios para el manejo del tiempo provistos por el RTI a los federados. En la figura, se observa que el RTI mantiene dos listas: una cola FIFO (por sus siglas en inglés -*First In First Out*-) donde los eventos se encuentran ordenados por el orden de llegada al RTI, y otra cola TSO donde los eventos se ordenan por marca temporal. Estos eventos se entregan a los federados, de acuerdo al tipo de servicio que requieren para la administración interna de su tiempo. En una misma federación, pueden existir federados que reciben los eventos de la cola FIFO, y otros federados que reciben los eventos de la cola TSO.

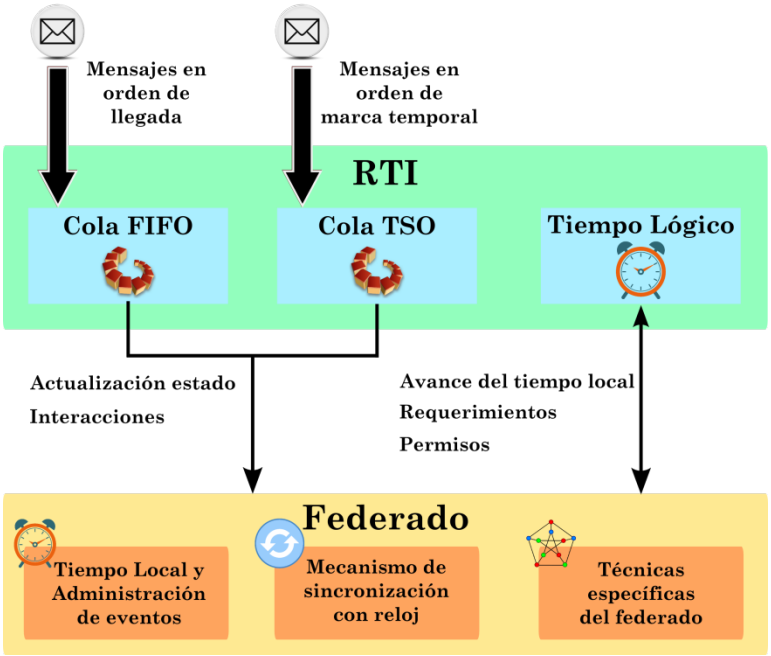


Figura 2.7 – Manejo del tiempo en HLA

Los mensajes que son recibidos por el RTI, pueden ordenarse de dos maneras: (1) por el orden de llegada RO (por sus siglas en inglés -*Receive Order*-), en cuyo caso el primer mensaje en llegar es el primero en ser enviado al destino, o (2) por orden de marca temporal TSO, en cuyo caso los mensajes van a ser entregados a sus destinos, cuando el RTI pueda determinar que no existe un evento con marca temporal menor, que puede ser recibido en un futuro inmediato para ese destino. Para lograr esta propiedad, el RTI debe calcular un límite inferior de las marcas temporales de los mensajes futuros.

La marca temporal es asignada al mensaje por el federado que lo genera, y el mensaje es entregado al receptor en orden no-decreciente de marca temporal. Dado que el RTI no puede entregar mensajes TSO inmediatamente, esta estrategia genera mayor latencia en los mensajes que la RO. Los federados pueden recibir cualquier tipo de mensajes, tanto TSO como RO. En el momento de asociarse a una federación, el federado informa al RTI qué tipo de servicio de entrega de mensajes va a requerir.

Los mensajes con la misma marca temporal, correspondiente a eventos simultáneos, son entregados al federado en un orden arbitrario; el federado que recibe estos mensajes, puede almacenarlos temporalmente en un buffer y ordenarlo. Puede suceder que el federado incluya un campo en la marca temporal del mensaje (cuando este mensaje es generado), a efectos de ordenar los eventos simultáneos. En este caso debe existir en la especificación de la federación, el formato y el significado del campo de marca temporal, así como los valores y duración del tiempo lógico. Entonces la federación puede, por ejemplo, especificar la representación del tiempo, usando un tipo de dato abstracto. La especificación del formato de la marca temporal, debe incluirse en el FOM. Algunas veces la federación debe indicar también, ciertas operaciones tales como, la comparación de los valores de marca temporal.

El RTI debe garantizar a los federados, no recibir eventos con marca temporal menor a su tiempo lógico actual. Para cumplir con esto, se establece un contrato entre el federado y el RTI, sobre cómo avanzar el tiempo lógico. Así, cada vez que un federado debe avanzar su tiempo lógico, dado que procesó todos los eventos en el tiempo actual de simulación, es necesario solicitar permiso al RTI. El federado no puede avanzar su tiempo lógico, hasta no recibir este permiso. Este protocolo es central en el manejo del tiempo en HLA.

El ciclo de administración del tiempo, consiste de tres etapas: (1) el federado envía al administrador del tiempo su requerimiento de avance del tiempo (a través de la invocación del servicio *Time Advance Request*). (2) El RTI entrega mensajes (tal vez cero) al federado, invocando servicios definidos en el federado, como puede ser *Reflect Attribute Value* para entregar nuevos valores de atributos de objetos o *Receive Interaction* para entregar eventos de interacción. (3) El ciclo se completa cuando el RTI invoca el proceso definido en el federado, denominado *Time Advance Grant*, que autoriza al federado a avanzar su tiempo lógico.

En la especificación de la interfaz de HLA, existen dos mecanismos que el federado puede utilizar, para avanzar su tiempo lógico: (1) TAR (por sus siglas en inglés -*Time Advance Request*-) y (2) NMR (por sus siglas en inglés -*Next Message Request*-). El mecanismo TAR es mejor para federados que usan internamente un mecanismo de tiempo en etapas, mientras que NMR es la primitiva preferida para federados dirigidos por eventos.

2.3 Técnicas de Modelado y Simulación

El término simulación, se puede definir como: *el proceso de modelar y desarrollar un modelo de computadora de un sistema o proceso dinámico*. Es decir, es un modelo que muestra el comportamiento del proceso o sistema, sobre el tiempo y el espacio (S. J. Taylor et al. 2013; Lustig et al. 2010). Además es posible realizar experimentos sobre este modelo, con el propósito de entender el comportamiento del sistema, evaluar estrategias alternativas de operación, y detectar oportunidades de mejora.

La simulación es un proceso mediante el cual se representa, reproduce, o imita, el comportamiento observable de un proceso real, a lo largo del tiempo y el espacio. Esta herramienta, permite describir y analizar el comportamiento del sistema real, detectar comportamientos emergentes, y utilizar las salidas de información para apoyar el diseño y mejora del sistema (Mittal, Diallo, y Tolk 2018). En general, la simulación se utiliza en apoyo a la toma de decisiones, ya sea para el diseño de sistemas antes de ser construidos, o probando políticas de funcionamiento con anterioridad a su uso sobre el sistema real. Como herramienta, la simulación ayuda a identificar los problemas relevantes, a evaluar cuantitativamente las soluciones alternativas, y a detectar oportunidades de mejora.

Por lo general, la simulación es apropiada cuando resulta muy difícil o imposible desarrollar un modelo analítico, cuando el sistema tiene una o más variables aleatorias relacionadas, cuando la dinámica del sistema es extremadamente compleja y produce comportamientos emergentes, o cuando el objetivo es observar el comportamiento del sistema sobre un período de tiempo.

Los avances en las metodologías de simulación, así como la gran disponibilidad actual de herramientas existentes en el mercado, han provocado que la técnica de simulación sea una de las herramientas más ampliamente usada, tanto en el análisis de sistemas como en el apoyo a la toma de decisiones (Powers, Sanchez, y Lucas 2012; Liu et al. 2012; Anagnostou y Taylor 2017; Rainey y Tolk 2015). Entre las ventajas que presenta el uso de la simulación, se pueden destacar las siguientes:

- En los modelos totalmente analíticos, la dinámica entre los componentes de un sistema es muy difícil de analizar. En tanto que en un modelo de simulación, la dinámica de interacción es más simple de comprender y visualizar.
- Es posible realizar un análisis de escenarios sin intervenir sobre el sistema real, lo que permite observar la respuesta del sistema ante diferentes configuraciones de parámetros y variables.
- Mediante la prueba del sistema en distintos escenarios, es posible comprender el comportamiento del mismo y, de esta forma, detectar los factores que determinan dicho comportamiento.
- Permite analizar cuáles son las relaciones que existen entre los factores del sistema, y como estas relaciones afectan al comportamiento.
- Es posible evaluar el rendimiento del sistema de acuerdo a ciertos indicadores, lo que permite detectar oportunidades de mejora.

A pesar de las ventajas mencionadas, también se pueden presentar obstáculos al momento de diseñar los modelos de simulación, tales como: dificultad para modelar y entender el problema; imposibilidad de obtener soluciones óptimas (Ingalls 2014); tiempo de desarrollo excesivo; elevado costo de desarrollo y ejecución; necesidad de habilidades

específicas para el desarrollo y la ejecución de los modelos; resultados difíciles de interpretar y la falta de un método para verificar su validez (S. J. E. Taylor et al. 2013).

Existen distintos tipos de modelos de simulación, de acuerdo a los cuales la simulación se puede clasificar en: estática o dinámica, determinística o estocástica, continua o discreta, sistemas combinados y basados en agentes. A continuación, se realiza una breve descripción de los tipos de modelos de simulación.

Modelos de Simulación Estáticos: Representan sistemas en estado estacionario, o bien sistemas en los que el paso del tiempo no es importante. Un caso de este tipo de simulación es la simulación Montecarlo.

Modelos de Simulación Dinámicos: Representan sistemas en los que el tiempo es una variable de interés. Un ejemplo típico son los sistemas de gestión de inventario.

Modelos de Simulación Determinísticos: En estos modelos la salida queda determinada una vez que se especifican los datos de entrada y las relaciones entre los mismos. Estos modelos no contienen ningún componente probabilístico.

Modelos de Simulación Estocásticos: Estos modelos producen una salida que es de carácter aleatorio en sí misma, y se utiliza únicamente para estimar las características reales del sistema. Para este caso, se toma en cuenta algún componente aleatorio de entrada.

Modelos de Simulación Continuos: Para estos modelos las variables de interés o de estado, pueden modificarse en cada instante de tiempo. Los modelos de crecimiento poblacional, son en general, un caso típico de este tipo de modelos de simulación.

Modelos de Simulación Discretos: Para estos modelos las variables de interés o de estado, pueden modificarse solo en aquellos momentos específicos que son determinados, por la ocurrencia de un evento. Por ejemplo, en un sistema de venta de productos online, cuando se produce el arribo de un pedido de compra para un producto.

Modelos de Simulación Híbridos: En estos modelos se integran tanto variables discretas como continúas. Para modelar tal tipo de sistemas, el modelo debe representar tanto los componentes discretos como los continuos y, además, las interacciones que pueden ocurrir

entre ellos. Existen tres tipos de interacciones entre las variables de estado, en este tipo de simulaciones: (1) Un evento discreto puede causar un cambio discreto, en el valor de una variable de estado continua; (2) Un evento discreto puede causar que la relación que gobierna una variable de estado conti

nua, cambie en un instante de tiempo en particular; (3) Una variable de estado continua puede causar que un evento discreto, sea disparado o planificado. Este tipo de modelos se utiliza normalmente en la planificación de lotes de producción, de los procesos de elaboración de productos químicos.

Modelos de Simulación Basados en Agentes: En estos modelos existen individuos autónomos dentro de un entorno (denominados agentes), los que en base a ciertos eventos recibidos del ambiente, toman una decisión, y modifican su contexto o a otros agentes.

Como se puede observar del resumen presentado, la simulación se puede realizar por medio de distinto tipo de modelos. Puntualmente para el caso de una CS, existe una gran cantidad de trabajos en la literatura, que resaltan los beneficios del uso de la simulación de eventos discretos en este dominio específico (Proudlove et al. 2017; Gogi, Tako, y Robinson 2016; Oliveira, Lima, y Montevechi 2016; Furian et al. 2015; Cigolini et al. 2014; Long y Zhang 2014; Mustafee, Katsaliaki, y Taylor 2014; Moon, Lee, y Cho 2012; Gutiérrez y Leone 2012; Yoo, Cho, y Yücesan 2010).

2.4 Desarrollo de Ontologías

En los tiempos actuales el desarrollo de una ontología se ha vuelto una actividad compleja, que requiere la consideración de un grupo de actividades y aspectos, los cuales deben ser definidos y evaluados. En este sentido, la construcción de una ontología ha tomado muchos conceptos de la ingeniería de software, a efectos de desarrollar sus modelos. Uno de los aspectos a considerar, es el nivel de usabilidad y re usabilidad que tendrá la ontología. Las ontologías más reusables son muy generales, es decir, en ellas solo se distinguen los tipos básicos de entidades del dominio, y se captura el conocimiento de manera independiente a cualquier problema a resolver. En el otro extremo, se encuentran las ontologías de aplicación,

las cuales son utilizadas para resolver únicamente problemas específicos. En estos casos, el vocabulario es tan específico para la solución del problema en cuestión, que las ontologías no resultan apropiadas para otras situaciones o problemas similares. En consecuencia, el problema es que cuanto más reusable es una ontología, se vuelve menos usable, y viceversa.

Todas las actividades relacionadas con el proceso de desarrollo de una ontología, su ciclo de vida, los métodos y metodologías para su construcción, así como las herramientas y los lenguajes que soportan este proceso de desarrollo, se agrupan en una disciplina que se denomina Ingeniería Ontológica (Gómez-Pérez, Fernández-López, y Corcho 2010). A partir de los años 90, se ha incrementado el desarrollo de esta disciplina. Entre las tendencias del área, se pueden destacar: la creación de ontologías de alto nivel, que permiten la clasificación de conceptos fundamentales como eventos, tiempo, espacio; la implementación de sistemas basados en ontologías, que permiten inferir conocimiento a partir de ciertos hechos verdaderos; la definición de ontologías para delimitar el conocimiento y el comportamiento, dentro de un dominio específico; la definición de lenguajes para representar el conocimiento; el uso de ontologías para la búsqueda de componentes, por medio de la clasificación en categorías definidas a través de la inferencia (Roussey et al. 2011).

En cuanto al desarrollo de lenguajes específicos para la implementación de ontologías, se puede decir que comenzó en los años 90 con la creación de lenguajes basados en Inteligencia Artificial, y que a partir de la popularización de Internet y la Web Semántica, continuó con la construcción de lenguajes que explotan las características de dichas tecnologías. Algunos ejemplos de lenguajes basados en inteligencia artificial, son KIF, CycL, LOOM, OCML y FLogic, los cuales se basan en lógica de primer orden, *frames* y lógica descriptiva. Dentro de los lenguajes más prominentes que utilizan la tecnología Web, se encuentran: SHOE, XOL, OIL, DAML+OIL y OWL (Negri et al. 2016). La sintaxis de estos lenguajes está basada en HTML (por sus siglas en inglés -*HyperText Markup Language*-) (W3C 2017) y XML. Dentro de este último grupo, se destacan tres lenguajes, los cuales establecen los fundamentos para la Web Semántica: RDF (por sus siglas en inglés -*Resource Description Framework*-) (W3C 2014a), RDF-Schema (W3C 2014b) y OWL (por sus siglas en inglés -*Web Ontology Language*-) (W3C 2012).

Luego de la breve descripción sobre la disciplina de ingeniería ontológica, en las siguientes secciones se desarrollan los conceptos, que se utilizan a lo largo de esta tesis. En particular, se ahonda sobre la definición de ontologías y redes de ontologías; la metodología de desarrollo que se utiliza para la construcción de la red de ontologías, desarrollada en el marco de tesis; y, por último, una breve revisión sobre la utilidad de las ontologías en la interoperabilidad entre sistemas, en particular en el área de modelado y simulación.

2.4.1 Ontologías y Redes de Ontologías

Una ontología es *una especificación formal y explícita de una conceptualización compartida de un dominio* (Hofmann, Palii, y Mihelcic 2011). Las ontologías son apropiadas para definir explícitamente, qué se entiende que existe en el dominio (Staab y Studer 2009), y cómo cada componente se relaciona con los demás componentes (Zeigler y Hammonds 2007). Las ontologías organizan la representación del conocimiento, y capturan la información sobre objetos en un dominio de conocimiento (Gómez-Pérez, Fernández-López, y Corcho 2010; Tolaba, Caliusco, y Galli 2014).

De acuerdo con (Maedche 2002), una ontología se define como: *una 6-tupla de conceptos, relaciones, jerarquías, una función que relaciona conceptos de manera no taxonómica, un conjunto de axiomas y un conjunto de reglas*. Formalmente esta definición puede expresarse, como se observa en la ecuación 2.5, de la siguiente manera:

$$O = \{C, R, H, rel, A, DR\} \qquad \text{(Ecuación 2.5)}$$

Donde:

C = Son los conceptos que representan las clases de objetos.

R = Son las relaciones entre los conceptos.

Como condición de O, se establece que C y R son dos conjuntos disjuntos.

H = Representa una jerarquía de conceptos o taxonomía, en donde $H \subseteq C \times C$ y $H(C_1, C_2)$ indica que C_1 es un subconcepto de C_2.

rel = $R \rightarrow C \times C$ es una función que relaciona los conceptos de manera no taxonómica.

A = Es un conjunto de axiomas que expresan las propiedades del modelo.

DR = Es un conjunto de reglas de derivación expresadas como cláusula de HORN.

Las ontologías se utilizan para **organizar la representación del conocimiento en un determinado dominio y razonar en base a él**.

Según (Gómez-Pérez, Fernández-López, y Corcho 2004), una ontología se define como: *una definición explicita de una conceptualización compartida de cierto dominio*. Desde una perspectiva pragmática, una ontología puede ser definida como, *un artefacto de representación basado en cuatro clases de componentes de modelado: conceptos, roles, restricciones e individuos* (Gómez-Pérez, Fernández-López, y Corcho 2010). Los conceptos representan clases de objetos, es decir, conceptos del mundo real que se categorizan como clases de objetos. Los roles describen las relaciones binarias entre los conceptos, por lo tanto, proporcionan además una descripción de las propiedades de los conceptos. Las restricciones son empleadas para expresar propiedades de los roles, como puede ser la cardinalidad. Los individuos representan instancias de los conceptos definidos, como son los objetos puntuales.

Además de estos componentes básicos, es posible usar axiomas y reglas para inferir nuevo conocimiento, a partir del conocimiento que se encuentra en la ontología. Los axiomas son sentencias lógicas siempre verdaderas, que expresan las propiedades de los conceptos modelados en la ontología. Las reglas son sentencias lógicas, al igual que los axiomas, que expresan las características del dominio como, por ejemplo, las reglas de negocio.

Estos últimos dos componentes, las reglas y los axiomas, son los que diferencian a una ontología de una taxonomía. En definitiva, una taxonomía solo clasifica términos jerárquicamente, usando la relación padre-hijo (también denominada en la literatura como generalización, es-un o tipo-de). Como se puede observar, una taxonomía deja de lado cualquier otro tipo de relaciones y, además, no permite definir atributos a los términos modelados (Guarino, Oberle, y Staab 2009). El lenguaje OWL es el estándar de facto utilizado para implementar una ontología, y no siempre resulta suficiente para realizar las inferencias necesarias, dado que no permite expresar reglas lógicas como cláusulas de HORN

(Negri et al. 2016; Zhao, Dong, y Peng 2009). Entonces, es necesario combinar OWL con otro formalismo de representación de reglas. Uno de los enfoques más utilizados para realizar esta combinación, es el lenguaje SWRL (por sus siglas en inglés -*Semantic Web Rule Language*-) (W3C 2004), que provee la habilidad de expresar reglas lógicas como cláusulas de HORN, en términos de los conceptos de OWL (O'Connor et al. 2005).

Para extraer información de las ontologías implementadas en OWL, es necesario un lenguaje de consultas. Uno de los lenguajes más poderosos es SPARQL (por sus siglas en inglés –*Simple Protocol And Rdf Query Language*-) (W3C 2013), que se trata de un lenguaje estandarizado para la consulta sobre grafos RDF. Entre las capacidades de este lenguaje, se pueden destacar las siguientes: soporte para agregaciones, sub consultas, negaciones, crear valores por medio de expresiones, consultar información requerida y opcional de los grafos RDF.

De acuerdo con (Gómez-Pérez, Fernández-López, y Corcho 2004), se destaca como característica distintiva el carácter consensuado, del conocimiento reflejado en una ontología, así como la capacidad de reusar, y compartir dicho conocimiento. Además, dichos autores realizan una distinción entre ontologías livianas y pesadas, incluyendo entre las primeras aquéllas que son básicamente una taxonomía; y entre las últimas, las que modelan el dominio con todos los componentes, y limitan su semántica a través de axiomas y restricciones. De acuerdo a esta interpretación, el espectro de las ontologías incluye un amplio rango de modelos, con expresividad y complejidad incremental.

De esta forma, pueden considerarse ontologías a los modelos que representan conocimiento que es compartido, consensuado o aceptado por una comunidad, y que está representado explícitamente. Además, es deseable que dicho conocimiento sea especificado formalmente, para que pueda ser interpretado sin ambigüedades, y pueda ser tratado computacionalmente.

Independientemente de la riqueza de la estructura interna de las ontologías, también se puede realizar una distinción de las mismas, considerando el área temática que éstas conceptualizan. A continuación se presenta la clasificación de los tipos de ontologías, propuesta por (Gómez-Pérez, Fernández-López, y Corcho 2004):

- Ontologías de Representación de Conocimiento: Capturan las primitivas usadas para formalizar el conocimiento, en un determinado paradigma de representación del conocimiento. Estas ontologías proveen definiciones formales, de las primitivas usadas en los lenguajes de representación del conocimiento como, por ejemplo, clases, atributos, relaciones y axiomas.
- Ontologías Generales o Comunes: Se usan para representar conocimiento común, reusable en distintos dominios. Estas ontologías incluyen conceptos relacionados con: cosas, eventos, tiempo, espacio, funciones, comportamiento, entre otras.
- Ontologías de Alto Nivel o de Nivel Superior: Describen conceptos muy generales y proveen nociones genéricas, con las cuales deberían vincularse los términos fundamentales de otras ontologías. El principal problema de este enfoque es que, existen diversas ontologías de alto nivel, y entre ellas difieren en la clasificación de los conceptos fundamentales.
- Ontologías de Dominio: Son reusables en un dominio en particular (ingeniería, medicina, cadenas de suministro, entre otras). Estas ontologías proveen conceptos y relaciones dentro de un dominio, los cuales son relativos a las actividades que se llevan a cabo en ese dominio particular, así como, respecto a las teorías y principios elementales que son aceptados en el mismo.
- Ontologías de Tareas: Describen el vocabulario relacionado con una tarea o actividad determinada (como puede ser realizar diagnósticos, realizar una reparación o comprar un artículo). Proveen un vocabulario sistemático de términos, usados para resolver problemas asociados con tareas genéricas, que pertenecen o no al mismo dominio.
- Ontologías de Tareas-Dominio: Son ontologías de tareas reusables dentro de un dominio en particular, pero no entre distintos dominios. Estas ontologías son independientes de la aplicación para la cual se utiliza la ontología.
- Ontologías de Métodos: Proveen definiciones de los conceptos y relaciones relevantes, los que se aplican para especificar un proceso de razonamiento, de forma tal que se concrete una tarea en particular.
- Ontologías de Aplicación: Contienen todas las definiciones necesarias para modelar el conocimiento requerido, de una aplicación informática específica. Generalmente

extienden y especializan, tanto el vocabulario de una ontología de dominio, como de una ontología de tareas.

Comprendido el concepto de una ontología y sus distintos tipos, a continuación, se presenta el concepto de red de ontologías.

Una red de ontologías es: *un conjunto de ontologías vinculadas a través de relaciones de mapeo, modularización, versión y dependencia* (Allocca, D'Aquin, y Motta 2009). En esta red, las meta-relaciones entre cada una de las ontologías son especificadas explícitamente (Diaz, Motz, y Rohrer 2011). Existen algunos modelos que cubren los aspectos sintácticos y semánticos de trabajar con las relaciones entre las ontologías, en una red de ontologías. En la ontología DOOR (por sus siglas en inglés -*Descriptive Ontology of Ontology Relations*-), las relaciones generales entre ontologías, como pueden ser: incluido-en, equivalente-A, similar-A y versionado, se encuentran definidas utilizando primitivas ontológicas y reglas (Allocca, D'Aquin, y Motta 2009).

Desde el punto de vista de integración de modelos, cada ontología de la red conceptualiza un dominio independiente, y tiene un rol particular. La principal ventaja de emplear una red de ontologías es la posibilidad de desarrollar un dominio dado, de una forma modular y colaborativa, donde cada ontología es lo suficientemente pequeña, para qué sea fácil de comprender y mantener. De este modo, los diseñadores pueden estar trabajando sobre las ontologías en forma concurrente.

2.4.2 Metodología de Desarrollo de Ontologías

A lo largo de esta tesis se desarrolla una red de ontologías, en donde cada ontología de la red es una ontología de dominio. Cada ontología se formula como un modelo aplicable a un ámbito definido, en base a un punto de vista específico, y por ello se considera una ontología de dominio (Roussey et al. 2011). Para llevar a cabo el desarrollo de esta red de ontologías, se realiza un adaptación de la metodología METHONTOLOGY (Breitman, Casanova, y Truszkowski 2007), en combinación con lo propuesto por la metodología NEON (Suárez-Figueroa, Gómez-Pérez, y Fernández-López 2012). Por lo tanto, la adaptación realizada queda conformada de cinco etapas, las cuales son:

- Determinar el Dominio: Como dominios se establecen, por un lado las federaciones HLA y por el otro las cadenas de suministro. Se definen dos dominios dado que se trabaja con una red de ontologías, la cual presenta los conceptos relacionados a simulaciones distribuidas basadas en HLA, para la evaluación del rendimiento de cadenas de suministro. El objetivo de la red de ontologías es dar un marco de referencia, para la construcción del modelo de objetos de federación, en la simulación distribuida basada en HLA de cadenas de suministro. El FOM resultante cumple con la condición de ser semánticamente interoperable, entre los federados participantes de la federación de CS. Por este motivo, los modelos semánticos que conforman la red de ontologías, se desarrollan en base a: 1) los conceptos del modelo SCOR para el dominio de cadenas de suministro y 2) los conceptos del estándar HLA para el dominio de federaciones.
- Determinar el Alcance: La red de ontologías tiene como alcance, los conceptos asociados a la evaluación del rendimiento de una CS (como son objetivos, atributos de rendimiento, métricas y procesos), los participantes y los roles que adoptan, las relaciones que poseen los participantes, los conceptos de conformación de una federación, los conceptos asociados a la gestión de los servicios del RTI, y los conceptos de conformación de un modelo de objetos. A fin de obtener información sobre la composición del modelo de objetos de federación, asociado a la SD basada en HLA de cadena de suministro, se definen preguntas de competencia. Una pregunta de competencia, se puede definir como: *una pregunta que la ontología debe ser capaz de responder* (Grüninger y Fox 1995). Estas preguntas sirven como prueba de control de calidad, ya que permiten determinar si la ontología resultante contiene suficiente información, como para responder a todas las preguntas planteadas de la forma esperada.
- Enumeración de Términos Importantes y Definición de Conceptos: En base a las descripciones de los dominios de cadenas de suministro y de federaciones HLA, se identifican cada uno de los conceptos relevantes, y se los transforma en un concepto de la red de ontologías propuesta.
- Definición de Propiedades de Conceptos: Se especifican atributos y relaciones, asociadas a cada uno de los conceptos identificados en la etapa previa. Las

dependencias entre los conceptos, se incorporan a los modelos bajo la forma de relaciones, utilizando un nombre descriptivo a fin de indicar, la forma en la cual se vinculan los elementos (por ejemplo, *hasSubprocess*).

- Definición de Aspectos de Propiedades: Se detallan para cada una de las propiedades identificadas en la etapa previa, las características o aspectos a considerar en su definición (como ser tipo de dato, dominio y rango, entre otras).

En cuanto a la representación gráfica que se utiliza para describir los elementos intervinientes en cada uno de los modelos, el formato elegido es similar al propuesto por (Gómez-Pérez, Fernández-López, y Corcho 2010), en donde:

- Los nodos grises representan conceptos.
- Los nodos blancos representan relaciones entre conceptos.
- El sentido de las flechas indica la dirección de las relaciones.
- La relación *isA* indica subsunción completa.

Cada uno de los modelos semánticos de la red incorpora un conjunto de reglas SWRL, a fin de contribuir a su correcta instanciación. Este lenguaje se utiliza para definir reglas, bajo el formato de implicancia lógica (permitiendo derivar nueva información, a partir de un conjunto de instancias ya definidas). En el caso del diseño de la red de ontologías, este lenguaje se utiliza para formular reglas que permiten mantener la consistencia, en relación a los vínculos requeridos como parte de la definición de los conceptos y, además, para el mapeo de conceptos entre el dominio de cadenas de suministro y federaciones HLA.

Es importante destacar que los modelos semánticos propuestos, son especificados en el idioma inglés con el fin de mantener la compatibilidad directa, con el material de base que se utiliza para la red de ontologías.

2.4.3 Ontologías para la Interoperabilidad de Sistemas

Como se mencionó anteriormente, el objetivo de la interoperabilidad semántica es lograr que sistemas de información autónomos, comprendan el significado de la información generada, y compartida por otros sistemas. Con este propósito, distintas investigaciones han

intentado producir simulaciones, a partir de la composición de modelos de simulación. Por ejemplo, en (Barnett y Miller 2000) se presenta un trabajo en donde se definen componentes de simulación basados en el modelo SCOR, y se los utiliza para construir una simulación distribuida basada en HLA, para la simulación de una cadena de suministro. En este trabajo, se muestra un primer intento por componer simulaciones a partir de otros componentes de simulación. Una de las áreas que más ha impulsado la investigación en la construcción de una simulación, a partir de la composición de otros modelos de simulación, es la de defensa militar. Por ejemplo, en (Lutz 2014; Mojtahed et al. 2011) se propone integrar aplicaciones específicas del dominio de entrenamientos militares, y además se presenta una metodología de trabajo parar lograr esta integración de componentes, en el caso de análisis particular.

Con el paso del tiempo, la idea de componer simulaciones se comenzó a expandir a otros dominios. Por ejemplo, en (Gregory Zacharewicz, Chen, y Vallespir 2008) se propone integrar los procesos de negocio de empresas del rubro aeroespacial. Además en esta propuesta, se introduce la idea de trabajar con un enfoque de integración dirigido por modelos. En las investigaciones realizadas por (Snively, Leslie, y Gaughan 2013; Rycerz et al. 2015), se ha propuesto una arquitectura para la composición de modelos de simulación, la cual permite realizar la ejecución de la simulación compuesta, en un entorno basado en la web. Además en este trabajo, los autores presentan algunos mecanismos para la validación de los modelos compuestos generados.

En cuanto al uso de ontologías en la comunidad de modelado y simulación (M&S), diversos autores han resaltado su importancia en este campo (Hofmann, Palii, y Mihelcic 2011; Bell et al. 2007; S. J. E. Taylor et al. 2010; Dragoicea et al. 2012; Silver, Hassan, y Miller 2007; Karhela, Villberg, y Niemistö 2012). Estos trabajos se centran principalmente, en utilizar a las ontologías para la búsqueda y reúso de componentes de simulación. De este modo, las ontologías son vistas como un medio adecuado, para la representación del comportamiento que poseen los modelos de simulación. Al poder representar ese conocimiento, sin ser dependientes del lenguaje de implementación del modelo de simulación, surge como un aspecto promisorio para el área de M&S el potencial de utilizar ontologías para buscar y reutilizar los componentes de simulación.

Además del uso como buscador de componentes, se han realizado investigaciones en las cuáles se destaca la necesidad de desarrollar modelos conceptuales, a efectos de lograr la interoperabilidad entre los componentes de simulación (Hofmann, Palii, y Mihelcic 2011; Tolk, Diallo, y Padilla 2012). Este hecho se debe a las características únicas que presenta el área de M&S, en donde cada modelo de simulación representa una parte de la realidad que es de interés para quien lo construye (Turnitsa, Padilla, y Tolk 2010). En otras palabras, no se parte de un conjunto de requerimientos o de una pregunta de investigación, para realizar el modelo de simulación. Este conocimiento se encuentra implícito en la perspectiva de quien modela y construye el componente, por lo que el uso de ontologías en modelado y simulación es de gran importancia, para capturar dicha perspectiva de forma explícita, sin ambigüedades, y de forma que sea comprensible por una máquina. Por estas razones, los autores de estos trabajos estipulan que el futuro del M&S no está en las soluciones de plataforma, sino en los componentes de simulación y los servicios.

Según (Hofmann, Palii, y Mihelcic 2011) las ontologías como especificaciones son prescriptivas, es decir, definen una semántica formal para el procesamiento automático de la información. Los trabajos de (Zhu y Liu 2014; Jain et al. 2015) plantean como motivación que uno de los objetivos de HLA es el reúso de componentes y que, con las herramientas actuales, resulta muy difícil cumplir con ese objetivo. Por este motivo, presentan un enfoque para componer simulaciones, en la búsqueda de disminuir el tiempo y esfuerzo necesario que conlleva hacerlo manualmente. Además, con este enfoque aportan herramientas para el reúso de componentes. El foco de estos trabajos se encuentra en la definición de reglas de transformación, para mapear los conceptos de distinto tipo de ontologías en el FOM. En (Silver, Hassan, y Miller 2007) se presenta un método para usar el conocimiento de las ontologías y, de este modo, facilitar el desarrollo de modelos de simulación. Se parte de ontologías de dominio, luego se realiza una transformación a un documento XML para, a partir de este último, producir un modelo de simulación ejecutable. El objetivo principal de realizar transformaciones automáticas, desde modelos conceptuales a modelos de simulación ejecutables, es proveer herramientas sólidas para los usuarios finales. De este modo, los usuarios ingresan la estructura de logística e interacciones de la CS de forma precisa, y sin requerir demasiado conocimiento de las técnicas de simulación (Cope et al. 2007).

En este sentido estos artículos sientan un precedente, sobre la factibilidad y aplicabilidad del uso de ontologías para la generación del modelo de objetos de federación, el cual es un documento de vital importancia para la ejecución de la SD.

En (G. Zacharewicz, Chen, y Vallespir 2009) se hace uso de las ontologías para el intercambio de información entre agentes, en una simulación distribuida basada HLA. En esta investigación, los autores se centran en el uso de ontologías no persistentes, es decir, que no se almacenan y solo se mantienen en memoria. Para cada una de las interacciones que existen entre los agentes, se crea una ontología nueva que representa la interacción a ser llevada a cabo, junto con los parámetros de la misma. Esta ontología de corta duración, se basa en los conceptos de HLA para definir las interacciones. La información de estas ontologías se utiliza para construir parcialmente, las interacciones del modelo de objetos de federación. En este trabajo se destaca que aún es necesaria la intervención humana, para configurar algunos parámetros de las interacciones definidas. Esta investigación fue un primer enfoque, desarrollado para demostrar que es posible construir de forma parcial (o total en nuevas investigaciones) el modelo de objetos de federación.

En (Jain et al. 2015) se presenta la construcción del FOM, a partir de las características que posee cada federado integrante de una federación. En este trabajo los autores plantean reutilizar federados existentes. De este modo utilizando el lenguaje SysML (Delligatti 2013), se describen las características que cada uno va a consumir o producir en la federación. Luego se definen relaciones entre las características, a fin de determinar el nivel de semejanza entre las mismas. Finalmente se utiliza un algoritmo, para determinar cuáles son las características más parecidas, y éstas conforman el FOM. A partir del FOM, se utiliza una herramienta para generar el código del embajador RTI. Luego el código generado, debe ser integrado de forma manual al código del federado. Esta integración implica la modificación de las rutinas para: determinar cuándo un atributo debe ser enviado (*update*) y recibido (*reflected*), e invocar los servicios de inicialización y de administración del tiempo. Si bien este enfoque construye el FOM, cuenta con un alto grado de intervención humana, no presenta mecanismos para garantizar la integridad, como así tampoco para verificar los diagramas modelados en SysML.

En (Zhu y Liu 2014) se propone la generación del FOM a través de una ontología de dominio, que conceptualiza componentes de simulación de los federados, con la información que los mismos pueden brindar o necesitar. Para construir la federación, se fusionan las ontologías de dominio de los distintos federados integrantes de la federación, en una ontología colaborativa. De este modo, se obtiene la información relativa a la federación. Esta fusión se hace empleando una función de semejanza, qué determina que tan parecidos son un par de conceptos, de las distintas ontologías de dominio. Esta función es un aporte realizado por los autores del trabajo. Una vez obtenida la ontología colaborativa, se generan ontologías de tareas para determinar quién publica y suscribe una interacción, como así también, los valores para los parámetros, atributos y propiedades de cada interacción u objeto. A partir de esta ontología, se puede generar automáticamente el FOM con sus respectivos valores. Tanto las ontologías de dominio como la de tareas, son creadas por los usuarios. Dicha tarea requiere el conocimiento necesario, tanto de simulación como de ontologías.

En (Sun et al. 2012) se presenta un método de construcción de un modelo de interoperabilidad basado en ontologías, que permite el reúso de subsistemas en varios contextos colaborativos de desarrollo. Se enfoca principalmente en el desarrollo colaborativo de productos. Se utiliza una ontología general (denominada meta ontología) para representar conceptos de HLA, asociados al dominio de desarrollo colaborativo de productos y a sus reglas de negocio. Para cada escenario a simular, dicha ontología se instancia con los datos del SOM de cada federado, por medio de un proceso automático (basado en un autómata de estado finito) definido por los autores. Finalmente se chequea la consistencia de la ontología, para verificar que no existan redundancias, sinónimos y posiciones incorrectas en las categorías. Esta ontología se denomina aplicable al escenario a simular, y es utilizada para lograr la interoperabilidad entre los federados. Este enfoque apuesta a generar ontologías aplicables, para luego fusionarlas en una gran ontología colaborativa, que abarque la mayor cantidad de escenarios de desarrollo colaborativo de productos. Como desventaja, se puede remarcar que: no se presenta cómo utilizar la información de las ontologías aplicables en la construcción del FOM, a pesar de remarcarse que para el reúso de federados existentes, se deben modificar las interfaces para que cumplen con lo definido en el FOM. Esta tarea disminuye la eficiencia del reúso, e implica una alta demanda de tiempo y esfuerzo.

2.5 Conclusiones

En este capítulo se presentan los conceptos fundamentales, para la comprensión de la temática de este trabajo de tesis. Entre ellos se analizan los conceptos de las tres grandes áreas (presentadas en la Figura 1.1), que involucran a la simulación distribuida de cadenas de suministro, los cuales son:

- Por el área de gestión de operaciones: Qué es una cadena de suministro, cuáles son sus particularidades, y el modelado de una cadena de suministro a partir del modelo de referencia SCOR.
- Por el área de computación aplicada: Qué es la simulación paralela y distribuida, así como también sus problemas asociados, como son la administración del tiempo y la interoperabilidad entre los componentes de simulación. Principalmente se hizo hincapié en el problema de interoperabilidad entre simuladores, el que es de especial interés en esta tesis. Se presentó el estándar IEEE 1516-2010 HLA, que es el más utilizado para realizar simulaciones distribuidas, y se describieron sus principales aspectos en cuanto a su organización y funcionamiento.
- Por el área de investigación de operaciones: Qué significa el término simulación, cuáles son sus ventajas sobre otras técnicas, cuáles son los tipos de modelos de simulación, y las técnicas que se utilizan para implementar un modelo de simulación.

Además de los conceptos de estas áreas, en este capítulo se introducen conceptos asociados al desarrollo de ontologías. Entre ellos, se destacan: qué es una ontología y red de ontologías, qué es la ingeniería ontológica, y una metodología de trabajo para el desarrollo de ontologías. Por último, se presenta una breve revisión sobre el uso de las ontologías en la interoperabilidad de sistemas.

Por lo tanto, además de los conceptos de las tres grandes áreas de la SD de CS, se añaden como fundamentales los conceptos del área de ingeniería ontológica, para poder analizar la temática de esta tesis.

Capítulo 3 Red de Ontologías para Especificar un Modelo de Interoperabilidad en la Simulación Distribuida basada en HLA de Cadenas de Suministro

En este capítulo se presentan las problemáticas asociadas a la SD basada en HLA de CS, a partir de las cuáles se especifican los requerimientos para el modelo conceptual de la red de ontologías denominada SCFHLA. Además, se presenta el diseño de las ontologías que componen la red, a partir de los dominios de cadenas de suministro y federaciones HLA; luego se detalla cómo se integran las ontologías de dominio a la red de ontologías. Tanto en cada uno de los dominios como en la red de ontologías, se define un conjunto de reglas para: garantizar la validez de los conceptos instanciados y relacionados, inferir nueva información y, por último, mapear instancias del dominio de cadenas de suministro al dominio de federaciones HLA. A su vez, se detalla cómo se lleva a cabo la implementación de la red de ontologías. Finalmente, se realiza una evaluación de cada una de las ontologías de dominio, como así también de la red de ontologías, con el fin de detectar problemas de modelado, diseño, y cumplir con el objetivo de la red de ontologías.

3.1 Problemáticas Asociadas a la SD basada en HLA de CS

En el contexto de cadenas de suministro colaborativas, en donde la coordinación de la colaboración es descentralizada, la SD basada en HLA emerge como una herramienta fundamental, para evaluar tanto el comportamiento de distintas configuraciones, como también la misma configuración en diferentes situaciones (tal como se estableció en la sección 2.1 Cadenas de Suministro). Además los resultados obtenidos del proceso de SD, brindan información que se utiliza como soporte para la toma de decisiones (tal como se estableció en la sección 1.1 Contexto). Para evaluar el rendimiento, es necesario modelar la

estructura (es decir, especificar los participantes de la CS y los vínculos o interacciones entre ellos). De este modo los vínculos entre las organizaciones, se interpretan como las interacciones entre las entidades de negocio que conforman la CS, como consecuencia de una relación de colaboración.

Modelar la estructura de una CS no es una tarea sencilla, menos aun cuando se realiza de forma colaborativa, entre múltiples participantes. Entre los problemas que surgen al realizar esta tarea, se destacan:

- *Especificar un vocabulario común, y definir la semántica asociada a cada concepto del vocabulario*. Por ejemplo: cada miembro de la cadena utiliza términos diferentes para describir conceptos similares, o utiliza términos idénticos para referir a conceptos diferentes. Esta situación introduce confusión y errores de uso del modelo.
- *Definir la información asociada a cada interacción, de los procesos interorganizacionales*. Por ejemplo: si una fábrica solicita suministros a su proveedor, entonces esta interacción debe contener información asociada al identificador del ítem solicitado, la cantidad requerida del mismo, y una fecha de entrega de la mercadería.
- *Acordar un objetivo común para la cadena de suministro*. Resulta esencial acordar un objetivo, dado que cada participante de la cadena necesita alinear su lógica de negocios interna, para alcanzar dicho objetivo común.
- *Decidir la dirección estratégica de la CS*, la cual se puede centrar en atributos externos (como por ejemplo la satisfacción del cliente) o en atributos internos (como por ejemplo la reducción de los costos).
- *Identificar y seleccionar el atributo de rendimiento junto con la métrica adecuada, para satisfacer el objetivo común de la CS.*

Además de la estructura de la cadena de suministro, otro de los modelos requeridos para llevar a cabo la SD basada en HLA, es el modelo de objetos de federación. La construcción de este modelo, en el contexto de colaboración entre los participantes de una

CS, implica que las organizaciones deben acordar cómo construir el FOM, para que el mismo represente el conocimiento de los datos a intercambiar (entre los federados), de forma consensuada y sin ambigüedad. Otras cuestiones a acordar entre los participantes, para la construcción del modelo de objetos de federación, son las siguientes:

- Identificar los federados involucrados en la federación.
- Seleccionar el federado que inicia la federación.
- Describir la información definida por el usuario (que es propia del escenario a simular en la federación).
- Especificar la información de gestión que utiliza la federación.
- Vincular los federados con los participantes, para definir sus responsabilidades.

Si los participantes logran acordar el vocabulario a utilizar para construir el modelo de objetos de federación, entonces pueden alcanzar la interoperabilidad semántica. En este punto es donde surge la posibilidad de utilizar ontologías, para lograr un acuerdo que exprese el conocimiento común, de los diversos federados de la SD basada en HLA. De esta manera, resulta natural utilizar una red de ontologías que conceptualice, tanto el dominio de CS como el de federaciones HLA.

Los usuarios de la simulación distribuida son: los analistas de negocio y los modeladores de CS. Dichos usuarios, son quienes utilizan los resultados de la simulación, para la toma de decisiones. Estos usuarios poseen un amplio conocimiento de los conceptos asociados a cadenas de suministro, pero no así sobre las herramientas necesarias para implementar una SD basada en HLA. Por lo tanto, contar con herramientas basadas en términos accesibles (conocidos por los usuarios), que permitan encapsular los conocimientos específicos sobre HLA para realizar una SD, resulta de gran interés. La base para el desarrollo de este tipo de herramientas, es la red de ontologías.

3.2 Análisis de Requerimientos de la Red de Ontologías

3.2.1 Especificación de Requerimientos

En base a las problemáticas descriptas con anterioridad, se definen como requerimientos de la red de ontologías, las siguientes propiedades:

- *Proporcionar un conjunto de conceptos claramente definidos, que se puedan utilizar fácilmente para modelar escenarios de colaboración, entre los participantes de una cadena de suministro*. Resulta fundamental representar la estructura de la CS, por medio de las organizaciones participantes, y las interacciones entre los procesos de negocio involucrados. Para cumplir con esta necesidad, se toma como referencia el modelo SCOR, el cual es ampliamente conocido y aceptado por los analistas de negocio del dominio. A partir de los conceptos del modelo SCOR, se propone representar la estructura de una CS en un contexto de colaboración, sin una empresa dominante.
- *Identificar el objetivo de una cadena de suministro, y determinar si la evaluación del mismo, corresponde a una perspectiva interna o externa*. Para determinar el objetivo y el atributo de evaluación del mismo, la red de ontologías debe incluir conceptos que permiten el modelado y establecen el vínculo, entre ambos aspectos.
- *Determinar una forma de medición clara, que sea aceptada por los participantes, permita la evaluación del rendimiento dentro del ámbito de la cadena de suministro, y su comparación con otras cadenas de suministro*. Este requerimiento refleja la necesidad de determinar qué se quiere medir, donde medirlo, cómo asociar estas mediciones para obtener un diagnóstico del rendimiento de la CS, y que el rendimiento sea comparable con otras cadenas de suministro. Esta característica de comparación del rendimiento, contribuye a evaluar cadenas de suministro de la misma industria (como por ejemplo la automotriz), o de diferentes industrias (como por ejemplo una CS

farmacéutica con una CS alimenticia). Para cumplir este requerimiento, se propone utilizar un conjunto de conceptos asociados a las métricas del modelo SCOR y a los procesos de negocio vinculados. En estos últimos, se releva la información requerida para el cálculo de las métricas. El uso de este conjunto de métricas, permite una evaluación diagnóstica del rendimiento dentro de la cadena, y su comparación con otras cadenas de suministro.

- *Proporcionar un conjunto de conceptos claramente definidos, que se puedan utilizar para representar la información asociada, a una simulación distribuida basada en HLA*. Para cumplir con este requerimiento se toma como referencia el estándar 1516-2010 HLA, el cual es ampliamente conocido y aceptado en el ámbito de simulación distribuida. A partir de sus conceptos, se debe representar la información asociada a la composición de la federación, el modelo de objetos de federación, y la gestión de los servicios del RTI.
- *Disminuir los conocimientos específicos sobre el estándar HLA, los cuales son necesarios para la construcción del modelo de interoperabilidad*. Para cumplir con este requerimiento se propone definir en la red de ontologías, reglas de mapeo entre los conceptos utilizados en la definición de la estructura de CS, y los conceptos de SD basada en HLA. De este modo, los usuarios se encargan de modelar la estructura de cadena de suministro, mientras que la red de ontologías (por medio de las reglas de mapeo), infiere la información necesaria para la construcción del modelo de interoperabilidad. A través del uso de la red de ontologías junto con las reglas de mapeo, se pretende disminuir el conocimiento necesario sobre el estándar HLA, para la construcción del modelo de interoperabilidad.
- *Definir un mecanismo para verificar, que la información del modelo de interoperabilidad es consistente*. A efectos de cumplir con esta necesidad, la red de ontologías debe contar con un conjunto de reglas de integridad, que guíen a los modeladores en el modelado de la estructura de CS. Además de este tipo de reglas, la red de ontologías debe permitir la vinculación de: los conceptos utilizados en la definición de la estructura de cadena de suministro,

con los conceptos necesarios de SD que conforman el modelo de interoperabilidad

3.2.2 Objetivos y Alcance

Con el objetivo de satisfacer los requerimientos definidos en la sección previa, se diseña e implementa una red de ontologías denominada ***SCFHLA*** (por sus siglas en inglés – *Supply Chain Federation High Level Architecture*-).

Las razones fundamentales por las cuales se opta por construir una red de ontologías, en lugar de utilizar una única ontología, son las siguientes:

- Permitir el desarrollo modular de cada ontología de dominio.
- Facilitar la gestión y mantenimiento de cada ontología.
- Soportar el desarrollo colaborativo y concurrente de la red de ontologías.
- Mantener cada ontología tan simple como sea posible, a efectos de facilitar su comprensión y entendimiento.

De esta manera, la red de ontologías provee:

- Un vocabulario común para definir la estructura de CS.
- Reglas de integridad que definen restricciones entre los conceptos, lo cual permite organizar los mismos y vincular conceptos de manera valida.
- Reglas de mapeo entre los conceptos definidos en las distintas ontologías.
- Reglas de derivación para unificar los conceptos necesarios de SD basada en HLA, a efectos de simular el modelo de CS.

Luego, el objetivo de la red de ontologías *SCFHLA* es el siguiente: proporcionar un marco conceptual, que permita la especificación del modelo de interoperabilidad, para la SD basada en HLA de cadenas de suministro. Por lo que, el alcance de la red de ontologías debe contemplar los conceptos que permitan evaluar el rendimiento de la estructura de CS, en un ámbito de simulación distribuida basada en HLA.

En este sentido, la red de ontologías propuesta define el vocabulario asociado, con la conformación de una federación HLA y el modelado de la estructura de CS. Por medio del vocabulario, la red de ontologías permite modelar y relacionar la información necesaria, del modelo de interoperabilidad para SD basada en HLA, de cadenas de suministro. Al utilizar esta información para la construcción del FOM, se garantiza que el modelo de objetos de federación obtenido, cumple con la condición de ser semánticamente interoperable.

La estructura de diseño elegida (red de ontologías), se justifica en la existencia de dos dominios claramente diferenciados: por un lado el dominio de cadenas de suministro, y por otro lado el dominio de federaciones HLA. Cada uno de estos dominios, se desarrolla por medio de una ontología de dominio con nombre: ***SCK*** (por sus siglas en inglés –*Supply Chain Knowledge*-) para el dominio de cadenas de suministro, y ***HLAFed*** (por sus siglas en inglés –*High Level Architecture Federation*-) para el dominio de federaciones HLA. Finalmente, la red de ontologías queda conformada por: la integración de ambas ontologías de dominio en una red, y la definición de las metarelaciones *performs* y *representA*.

En relación a la evaluación del rendimiento de la estructura de CS, se establece como alcance requerido de la red, los conceptos de: cadena de suministro, participante de CS, relación entre participantes, objetivo de CS, atributos de evaluación del rendimiento, métricas asociadas a estos atributos, y procesos de negocio que recopilan la información necesaria para el cálculo de las métricas. Todos estos conceptos se han tomado del modelo SCOR, aunque los conceptos de cadena de suministro, participante, relación entre participantes, y objetivo, no se encuentran definidos como tales, en el modelo de referencia. Para el caso de los conceptos que no se encuentran definidos, se hacen menciones de su significado y uso, mediante distintos nombres a lo largo de la especificación del modelo SCOR. Es por esto que, a partir de un análisis pormenorizado, se ha detectado la relevancia de dichos conceptos, para la evaluación del rendimiento de la estructura de una CS[1].

[1]El resto del contenido del modelo SCOR (aplicaciones especiales, prácticas asociadas a cada uno de los procesos y, por último, habilidades requeridas por las personas para realizar un proceso) queda fuera del alcance de la red propuesta, debido a que no es necesario para el objetivo definido de la red.

En relación a la SD basada en HLA, se establece como alcance requerido de la red, los conceptos de: federación, federado, modelo de objetos tanto de la federación, como del federado; clases de objetos e interacciones, atributos, parámetros y propiedades definidas por el usuario para un escenario específico; clases de objetos e interacciones de gestión del RTI. Tanto la información del escenario como la de gestión del RTI, conforman el modelo de objetos. Todos estos conceptos se han tomado del estándar HLA, tanto de la especificación de la interfaz del federado, como del modelo de objetos. Además, se han incluido los conceptos de administrador y objetivo, para representar qué federado es el encargado de iniciar la federación y cuál es el objetivo de la federación, respectivamente. La Tabla 3.1 resume el alcance de la red de ontologías SCFHLA, a partir de los conceptos que abarca.

Tabla 3.1 – Alcance SCFHLA

Dominio	Conceptos
Cadenas de Suministro	Cadena de Suministro – Objetivo – Atributo de Rendimiento – Métrica – Proceso de Negocio –Participante – Relación
Federaciones HLA	Federación – Federado – Modelo de Objeto – FOM – SOM – Objetivo – Administrador – Clase de Objeto – Clase de Interacción – Atributo – Parámetro – Clase de Objeto de Gestión – Clase de Interacción de Gestión

3.2.3 Preguntas de Competencia

Para obtener información acerca del modelo de objetos de federación, se plantean preguntas de competencia. Estas preguntas establecen un control para determinar si la red de ontologías, es capaz de responder a las necesidades de información de los usuarios. La Tabla 3.2 presenta una clasificación, de las preguntas de competencia formuladas para la red de ontologías, de acuerdo a la temática sobre la que se desea obtener información.

Tabla 3.2 – Preguntas de competencia para la red SCFHLA

Temática	ID	Pregunta de Competencia
Estructura de Cadenas de Suministro	1	¿Cuáles son los procesos asociados a la métrica *m*?
	2	¿Qué atributo de rendimiento es evaluado por la métrica *m*?
	3	¿Cuáles son las métricas asociadas al proceso *bp*?
	4	¿Cuáles son las variables que componen la fórmula *x*?
	5	¿Qué procesos conforman la cadena de suministro *sc*?
	6	¿Cuál es el objetivo de la cadena de suministro *sc*?
Federaciones HLA	7	¿Qué federados forman parte de la federación *fed*?
	8	¿Cuáles son las clases de objetos que conforman el FOM *f*?
	9	¿Cuáles son las clases de interacción que conforman el FOM *f*?
	10	¿Qué federado es el encargado de iniciar la federación *fed*?
	11	¿Cuáles son los parámetros de la clase de interacción *ic*?
	12	¿Cuáles son los atributos de la clase de objeto *obj*?
Federaciones HLA de Cadena de Suministro	13	¿Qué procesos ejecuta el federado *y*?
	14	¿Qué métricas están asociadas al federado *y*?
	15	¿Qué procesos se representan en la federación *fed*?
	16	¿Qué objetivo tiene la cadena de suministro asociada a la federación *fed*?
	17	¿Cuáles son las clases de interacción que contiene la federación *fed*?
	18	¿Qué métricas se utilizan para evaluar a la federación *fed*?

3.2.4 Grupos de Usuarios y Escenarios de Uso

En cuanto a quienes van a hacer uso de la red de ontologías, se identifican tres grupos posibles de usuarios.

El primer y principal grupo de usuarios, es la comisión directiva de una cadena de suministro o los analistas de negocio. Este grupo puede utilizar la red para dos tipos de análisis: por un lado para evaluar el rendimiento de una estructura de CS, a partir de distintos atributos de rendimiento y métricas; mientras que, por otro lado, para experimentar con distintas estructuras de CS, bajo los mismos atributos de rendimiento y métricas.

El segundo grupo de usuarios, lo constituyen los desarrolladores de aplicaciones basadas en ontologías. Este grupo utiliza la red de ontologías para distintos fines, como por ejemplo: buscar información acerca de los procesos y/o métricas, u obtener información sobre el modelo de objetos de federación de la CS.

El tercer grupo de usuarios, lo conforman los expertos en HLA. Este grupo se encarga de aportar mejoras, en cuanto al diseño de la ontología de dominio HLAFed. Además este grupo verifica que no existan inconsistencias, en la información necesaria para la construcción del modelo de objetos de federación. Dicha información, se infiere a partir de la estructura de CS.

Dentro de los posibles usos de la red de ontologías, se definieron tres escenarios posibles: modelar estructura de cadena de suministro, obtener información sobre la estructura de cadena de suministro, y por último, verificar la información asociada al modelo de objetos de federación.

En el primer escenario se utiliza la red de ontologías, para modelar la estructura (todos los participantes y los vínculos entre ellos) y la forma de evaluar el rendimiento de CS (definir objetivo, atributo de rendimiento, métricas y procesos de negocios). Este escenario se encuentra destinado al primer y segundo grupo de usuarios, es decir, a los analistas de negocios y a los desarrolladores de aplicaciones basadas en ontologías.

El segundo escenario refleja cómo realizar consultas, para obtener información sobre la estructura de cadena de suministro, o el modelo de objetos de federación. En este caso, los tres grupos de usuarios pueden utilizar el escenario, a fin de obtener la información requerida.

Finalmente, el tercer escenario permite corroborar que la información asociada al modelo de objetos de federación, es consistente con la estructura de CS que se define en el Escenario 1. Además, este escenario permite validar que la información asociada al modelo de objetos de federación, respeta lo establecido por el estándar HLA. Este escenario se encuentra destinado al tercer grupo de usuarios, ya que es el único grupo que puede verificar si la información inferida (asociada al FOM), es consistente con la estructura de CS definida.

Las Tablas 3.3, 3.4, y 3.5 especifican los escenarios propuestos. Para tal fin se utiliza la plantilla propuesta por (Brusa, Caliusco, y Chiotti 2006), la cual se basa en los esquemas previstos para la descripción de casos de uso, de UML (OMG 2013).

Tabla 3.3 – Escenario 1: Modelar estructura de cadena de suministro

<table>
<tr><td colspan="2">Nombre: Modelar estructura de cadena de suministro</td><td>Nro: EM 001</td></tr>
<tr><td colspan="2">Objetivo: Modelar la estructura y la forma de evaluar el rendimiento de cadena de suministro</td><td>Versión: 1.0</td></tr>
<tr><td colspan="3">Actores:
Analistas de negocio.
Desarrolladores de aplicaciones basadas en ontologías.
Precondiciones:
La red de ontologías se encuentra abierta en la herramienta Protégé, y lista para su uso.
Evento Disparador:
Creación del modelo de una cadena de suministro.
Postcondiciones:
Éxito: Se ha modelado la estructura y forma de evaluar el rendimiento de CS.
Fracaso: No se ha podido crear la estructura y la forma de evaluación del rendimiento de CS.
Requerimientos Especiales: -</td></tr>
<tr><th colspan="2">Flujo Normal</th><th>Flujo Excepcional</th></tr>
<tr><td colspan="2">El actor identifica la cadena de suministro.</td><td>El proceso falla, produce un error y el escenario concluye.</td></tr>
<tr><td colspan="2">Se determinan cuáles son los participantes, sus procesos, y los vínculos entre ellos, para definir la estructura de CS.</td><td>La creación de procesos y relaciones falla, produce un error y el escenario concluye.</td></tr>
<tr><td colspan="2">Se define el objetivo que desea alcanzar la CS, y el atributo de rendimiento para evaluar el objetivo.</td><td>La definición del objetivo o el atributo de rendimiento fallan, producen un error y el escenario concluye.</td></tr>
<tr><td colspan="2">Se selecciona que métricas van a evaluar el atributo de rendimiento, definido en el paso previo.</td><td>La elección de métricas falla, produce un error y el escenario concluye.</td></tr>
<tr><td colspan="2">Se instancia la estructura de CS con la información de los puntos previos.</td><td>La instanciación del modelo falla, produce un error y el escenario concluye.</td></tr>
<tr><td colspan="2">El escenario motivador concluye.</td><td></td></tr>
</table>

Tabla 3.4 – Escenario 2: Obtener información de estructura de CS o la federación HLA

<table>
<tr><td colspan="2">Nombre: Obtener información de estructura de CS o la federación HLA</td><td>Nro: EM 002</td></tr>
<tr><td colspan="2">Objetivo: Consultar la red de ontologías para obtener la información requerida</td><td>Versión: 1.0</td></tr>
<tr><td colspan="3">Actores:
Analistas de negocio.
Desarrolladores de aplicaciones basadas en ontologías.
Expertos en HLA.
Precondiciones:
La red de ontologías se encuentra abierta en la herramienta Protégé, y lista para su uso.
La red de ontologías está poblada (contiene instancias o individuos de los conceptos).
Evento Disparador:
Consulta sobre la información de la estructura de CS, o del modelo de objetos de federación.
Postcondiciones:
Éxito: Se ha obtenido una respuesta a la consulta realizada.
Fracaso: No se ha obtenido una respuesta a la consulta realizada.
Requerimientos Especiales: -</td></tr>
<tr><th>Flujo Normal</th><th colspan="2">Flujo Excepcional</th></tr>
<tr><td>El actor ingresa la información necesaria para realizar la consulta a la red de ontologías.</td><td colspan="2">El ingreso de información falla, produce un error y el escenario concluye.</td></tr>
<tr><td>La consulta definida en el paso previo, es ejecutada sobre la red de ontologías.</td><td colspan="2">La ejecución de la consulta falla, produce un error y el escenario concluye.</td></tr>
<tr><td>Se presentan los resultados obtenidos de la consulta.</td><td colspan="2">La presentación de resultados falla, produce un error y el escenario concluye.</td></tr>
<tr><td>El escenario motivador concluye.</td><td colspan="2"></td></tr>
</table>

Tabla 3.5 – Escenario 3: Verificar información del modelo de objetos de federación

<table>
<tr><td colspan="2">Nombre: Verificar información del modelo de objetos de federación</td><td>Nro: EM 003</td></tr>
<tr><td colspan="2">Objetivo: Corroborar que la información asociada al modelo de objetos de federación, es consistente con la estructura de CS definida en el Escenario 1, y que la información respeta lo establecido por el estándar HLA.</td><td>Versión: 1.0</td></tr>
<tr><td colspan="3">Actores:
Expertos en HLA.
Precondiciones:
La red de ontologías se encuentra abierta en la herramienta Protégé, y lista para su uso.
La red de ontologías está poblada (contiene instancias o individuos de los conceptos).
Evento Disparador:
Ejecución de las reglas de inferencia, de la red de ontologías.
Postcondiciones:
Éxito: La información asociada al modelo de objetos de federación, es consistente con la estructura de CS, y además, respeta lo establecido por el estándar HLA.
Fracaso: La información asociada al modelo de objetos de federación, presenta inconsistencias o no respeta lo establecido por el estándar HLA.
Requerimientos Especiales: -</td></tr>
<tr><th>Flujo Normal</th><th colspan="2">Flujo Excepcional</th></tr>
<tr><td>La red de ontologías infiere información.</td><td colspan="2">La inferencia falla, produce un error y el escenario concluye.</td></tr>
<tr><td>El actor compara si la información inferida, representa la estructura de CS instanciada.</td><td colspan="2">Si la información inferida es incompleta, el actor instancia la información faltante, y luego el escenario concluye.</td></tr>
<tr><td>El actor analiza la información inferida, a efectos de determinar si respeta lo establecido por HLA.</td><td colspan="2">Si la información no respeta lo establecido por HLA, el actor sugiere las causas posibles para identificar la violación al estándar, y luego el escenario concluye.</td></tr>
<tr><td>El escenario motivador concluye.</td><td colspan="2"></td></tr>
</table>

3.3 Diseño de la Red de Ontologías

3.3.1 Ontología SCK

En la sección 2.1 (Cadenas de Suministro) se estableció qué, para hacer uso de la información cuantitativa de una simulación de cadena de suministro, es necesario construir un modelo de simulación que represente la estructura de la misma. Además, en la sección 2.1.1 (Modelo SCOR) se presentó el modelo de referencia de operaciones para cadenas de suministro, que es el modelo más actual y completo que se encuentra en la literatura, para evaluar las actividades y comparar el rendimiento, de la cadena de suministro. Entonces teniendo en cuenta estas aclaraciones, la ontología SCK define la estructura de una CS (como parte de la red de ontologías SCFHLA), basándose en los conceptos de dicho modelo de referencia. Por medio del diseño de una ontología basada en el modelo SCOR, se pretende lograr una especificación con un mayor grado de formalidad, una sintaxis más clara, y sin ambigüedades conceptuales. De esta forma, se obtiene una mejora sobre la definición del modelo de referencia, mitigando sus falencias.

La Tabla 3.6 resume el conjunto de actividades que se realizan, sobre los elementos del modelo de referencia, con el fin de obtener los componentes de la ontología SCK.

Cada uno de los conceptos que se identifican para la evaluación del rendimiento en el modelo de referencia, se transforma en un concepto de la ontología con el mismo nombre. Por ejemplo: para evaluar el rendimiento, es necesario establecer cuál es el objetivo de la cadena de suministro, por lo tanto, la definición del objetivo se transforma en el concepto *Goal* de la ontología SCK (Actividad 1). Además de estos conceptos, se identifican términos relevantes en el modelo de referencia que no se encuentrán definidos como conceptos. Entonces cada uno de ellos, se transforma en un concepto de la ontología. Por ejemplo: las relaciones entre los participantes de la cadena son relevantes para la evaluación del rendimiento, pero en el modelo de referencia solo se mencionan las entradas y salidas de cada proceso, mediante un nombre sin descripción. Por este motivo, se añade el concepto *Relation* a la ontología SCK (Actividad 2).

Tabla 3.6 – Actividades de mapeo entre el modelo SCOR y los componentes de la ontología

Actividad #	Descripción	Tipo de Componente	Ejemplo
1	Cada concepto identificado para la evaluación del rendimiento se transforma en un concepto de la ontología con el mismo nombre	Concepto	La definición del objetivo se transforma en el concepto *Goal*
2	Cada termino relevante identificado se transforma en un concepto de la ontología con el mismo nombre	Concepto	El término "relación" se añade a la ontología, como el concepto *Relation*
3	Se definen las relaciones entre los conceptos identificados	Relación	Se define la relación *measured*, para vincular al concepto *PerformanceAttribute* con el concepto *Metric*
4	Cada nivel de la jerarquía de procesos se transforma en un concepto	Concepto	El nivel paso de la jerarquía de procesos, se transforma en el concepto *Task*
5	La jerarquía de procesos se define mediante la relación *isA*	Relación	Se define la relación *isA*, entre el concepto *SubProcess* y el concepto *BusinessProcess*
6	Un *Process* se descompone en *SubProcess*, a su vez un *SubProcess* se descompone en *Task*. Entonces para representar estas descomposiciones, se definen las relaciones *hasSubprocess* y *hasTask* respectivamente	Relación	El proceso *Make* es de nivel *Process*, el cual posee la relación *hasSubprocess* con los subprocesos *Make-to-Stock*, *Make-to-Order*, y *Engineer-to-Order*, del nivel *SubProcess*

Actividad #	Descripción	Tipo de Componente	Ejemplo
7	Se añade una especificación de los roles de un participante, según el tipo de actividad que realiza	Concepto	La actividad de abastecer se añade a la ontología, como el concepto *Deliver*
8	La jerarquía de roles se define mediante la relación *isA*	Relación	Se define la relación *isA* entre el concepto *Authority* y el concepto *Role*
9	Se añaden los atributos, para definir las propiedades específicas de los conceptos de la ontología	Propiedad	Se añade el atributo *bpLevel* al concepto *BusinessProcess* para modelar la propiedad nivel de los procesos de negocio

Los conceptos identificados y añadidos al modelo semántico poseen vínculos entre ellos, y los mismos se definen en la ontología como relaciones. Por ejemplo: una vez que se identifica el atributo mediante el cual se va a evaluar el objetivo de la cadena de suministro, es necesario determinar la métrica encargada de medir ese atributo. Entonces, el concepto *PerformanceAttribute* se vincula con el concepto *Metric* mediante la relación *measured*, y esta última se vincula con el concepto *PerformanceAttribute*, por medio de la relación *measuringA* (Actividad 3).

Los procesos de negocio del modelo de referencia comparten información común, y se diferencian en el tipo de actividades que desarrollan, a efectos de cumplir con los resultados esperados. Además, teniendo en cuenta que los procesos se pueden descomponer en una jerarquía de niveles (también llamada descomposición vertical de procesos), se diseña cada nivel de esta jerarquía como un nuevo concepto de la ontología. De este modo, el nivel de alcance se define mediante el concepto de *Process*, el nivel de configuración se define a través del concepto de *SubProcess* y, por último, el nivel de paso se define mediante el concepto de *Task* (Actividad 4). Para representar esta relación de jerarquía, cada uno de estos niveles se vincula por medio de una relación *isA* al concepto denominado *BusinessProcess*,

que es el elemento raíz de la descomposición. Por ejemplo: el concepto *Task* se vincula con el concepto *BusinessProcess*, por medio de la relación *isA* (Actividad 5). A su vez, *Process* posee un vínculo con *SubProcess* para representar los subprocesos que componen el proceso, y son del mismo tipo (en la estructura horizontal de procesos), pero de diferente nivel (en la jerarquía de niveles). Entonces, se transforma este vínculo en la relación *hasSubprocess*. Además, *SubProcess* tiene un vínculo con *Task* para representar las tareas que componen el subproceso. En esta relación se cumple la misma condición que en la relación *hasSubprocess,* las tareas son del mismo tipo, pero de diferente nivel que el subproceso que integran o componen. Este vínculo se modela mediante la relación *hasTask*. Por ejemplo: el proceso *Make* es de nivel *Process*, el cual tiene asociados los subprocesos *Make-to-Stock*, *Make-to-Order*, y *Engineer-to-Order* de nivel *SubProcess*. Estos subprocesos son del mismo tipo que el proceso *Make* (de tipo *Make*), pero difieren en su nivel (el proceso es de nivel *Process* mientras que los subprocesos son de nivel *SubProcess*) (Actividad 6).

Los participantes de la cadena tienen un rol asignado, para identificar cuáles son las actividades que cumplen en la CS. En la Actividad 2 se definió el concepto *Role* para modelar el rol asignado a un participante, pero en la misma no se especificó que roles pueden existir dentro de la cadena. Por esto motivo, con el objetivo de mejorar la especificación inicial del concepto *Role*, se definen tipos de roles. De este modo, se añaden a la ontología SCK los conceptos: *Source* (abastecer), *Make* (producir), *Deliver* (entregar) y *Authority* (autoridad) (Actividad 7). Para representar esta especificación, cada uno de estos tipos de roles se asocia, por medio de una relación *isA*, al concepto *Role*. Por ejemplo: el concepto *Authority* se vincula con el concepto *Role*, por medio de la relación *isA* (Actividad 8).

Una vez que se han especificado todos los conceptos y relaciones, necesarios para representar los elementos del modelo de referencia SCOR, se incorporan atributos para definir las propiedades específicas de los conceptos de la ontología. Por ejemplo: el concepto *BusinessProcess* contiene toda la información común a los procesos, entre sus propiedades se destacan: el nivel (atributo *bpLevel*), que hace uso de los números enteros (tipo de dato *integer*), el identificador de proceso (atributo *bpId*) y el tipo de proceso (atributo *bpType*), los cuales hacen uso de las cadenas de caracteres (tipo de dato *string*) (Actividad 9).

Como resultado de las actividades realizadas, se obtiene el modelo presentado en la Figura 3.1.

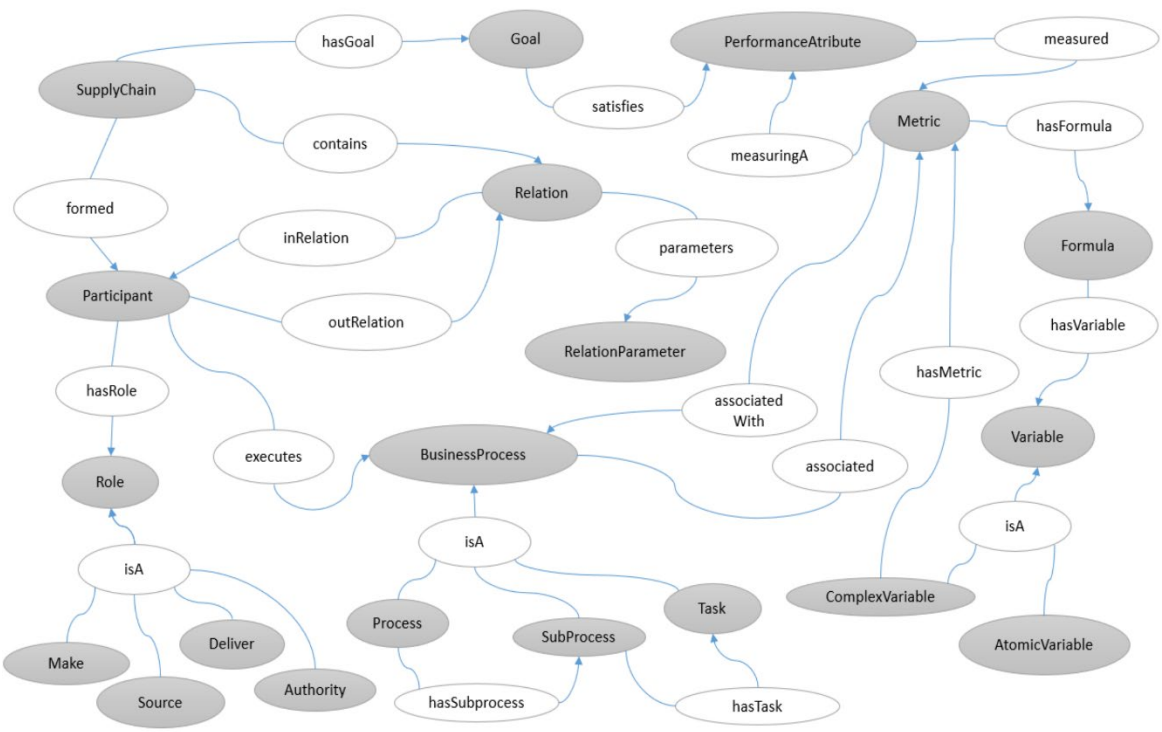


Figura 3.1 – Ontología SCK

Como se observa en la Figura 3.1, una cadena de suministro se modela a través del concepto *SupplyChain*. Una organización que es miembro de una cadena se modela con el concepto *Participant*, y las relaciones existentes entre los miembros se modelan con el concepto *Relation*. Estas relaciones, implican que existe un intercambio de información y/o materiales entre los participantes. Se consideran como relaciones de entrada (relación *inRelation*), a aquellas en donde el participante recibe interacciones de otro participante; y se contemplan como relaciones de salida (relación *outRelation*), a aquellas en donde un participante envía información y/o materiales a otro miembro de la cadena. Estas relaciones tienen un conjunto de parámetros asociados para representar la información a intercambiar, en donde estos parámetros se modelan con el concepto *RelationParameter*. Además el

vínculo entre las relaciones y sus parámetros, se modela a través de la relación *hasParameter* entre los conceptos *Relation* y *RelationParameter*.

A su vez los miembros pueden jugar distintos roles, dentro de la cadena de suministro. Algunos de ellos se definen de acuerdo al modelo de referencia SCOR, como son *Make*, *Deliver* y *Source*; mientras que el rol *Authority* tiene un significado especial. Este último rol, identifica al participante que inicia la tarea de modelado de la estructura de CS, e invita a los demás participantes a unirse a la tarea. El rol *Source* se define como aquel componente que, recibe requerimientos de materias primas y/o servicios, y se encarga de entregar materia prima y/o servicios. El rol *Make* es aquel que, requiere y recibe materia prima, pero a la vez, la transforma en un producto final. Este rol se encuentra asociado generalmente con las fábricas. El rol *Deliver* es aquel que, entrega y recibe productos o servicios, ya sean intermedios o finales. Generalmente, se asocia con el transportista o procesos de logística.

Los participantes tienen procesos que deben ejecutar, para cumplir con sus actividades dentro de la cadena. El vínculo entre los participantes (concepto *Participant*) y los procesos de negocio (concepto *BusinessProcess*), se modela a través de la relación *executes* (relación *ejecuta*). Como se mencionó con anterioridad, los procesos de negocio tienen una descomposición jerárquica de niveles y, por este motivo, se especificaron como subclases del concepto *BusinessProcess*, a los conceptos: *Process*, *SubProcess*, y *Task*, a efectos de representar los distintos niveles de abstracción. El concepto *Process* considera los tipos de procesos involucrados, el alcance de la cadena, los objetivos de rendimiento a alcanzar, y las estrategias de largo plazo. El concepto *SubProcess* establece las estrategias operativas de mediano plazo, así como también las capacidades y limitaciones de los procesos de negocio colaborativos, de la cadena de suministro. Por último, el concepto *Task* define un conjunto de políticas operativas para el corto plazo, además de la configuración detallada, dentro de los procesos de negocio individuales. La relación *hasSubprocess*, define la composición de un proceso por parte de diferentes subprocesos, mientras que la relación *hasTask*, modela la composición de un subproceso por parte de distintas tareas.

Toda cadena de suministro define un objetivo, el cual se pretende alcanzar por medio de la colaboración entre todos sus miembros. En el modelo semántico se define el concepto *Goal*, para representar al objetivo de la cadena. Estas metas, generalmente, se expresan en lenguaje natural y están relacionadas a atributos de rendimiento (concepto *PerformanceAttribute*). Los atributos de rendimiento determinan si la estrategia para alcanzar la meta, está basada en atributos internos o externos. Dado que los atributos de rendimiento delimitan una dirección estratégica, es necesario evaluarlos a través del uso de métricas. Las métricas se representan en el modelo semántico, mediante el concepto *Metric*. Para poder calcular el valor de una métrica, es necesario obtener información de los procesos de negocio de la cadena. Además un proceso de negocio tiene una métrica asociada, a efectos de evaluar en qué grado se satisface el objetivo, de la cadena de suministro. Por estos motivos se define una relación entre *BusinessProcess* y *Metric*, en donde un proceso de negocio está asociado a una métrica (relación *associated*), y esta última está asociada con un proceso de negocio (relación *associatedWith*).

Las métricas asociadas a la medición de un atributo de rendimiento, deben ser calculadas por medio del uso de una fórmula. Se define en la ontología el concepto *Formula*, para representar la fórmula de una métrica. Además, para modelar el vínculo entre los conceptos *Metric* y *Formula* se define la relación *hasFormula*. Una fórmula está definida por una ecuación que se encuentra compuesta de variables. Estas variables se representan mediante el concepto *Variable*, y las mismas pueden ser atómicas o compuestas, dependiendo de si la variable es un valor o requiere de una métrica para su cálculo, respectivamente. El vínculo entre una fórmula y sus variables se representa con la relación *hasVariable*, mientras que la especificación de una variable se representa como una jerarquía entre *Variable* y sus tipos específicos, mediante la relación *isA*. Los tipos específicos de una variable son: *AtomicVariable* (concepto para las variables atómicas) y *ComplexVariable* (concepto para las variables complejas). Este último tipo de variable, requiere de un cálculo para obtener su valor, y en el modelo SCOR, solo las métricas requieren fórmulas para ser calculadas. Por lo tanto, se define un vínculo entre los conceptos *ComplexVariable* y *Metric* denominado

hasMetric, a efectos de representar la necesidad de una métrica, para obtener el valor de la variable compleja.

Los conceptos asociados a una métrica que se modelan en la ontología SCK (*Metric*, *Formula*, *Variable*, *AtomicVariable* y *ComplexVariable*), respetan la estructura de descomposición de métricas del modelo SCOR. Además como se pretende que el usuario de la red, utilice las métricas para evaluar la estructura de CS, se define como parte de la ontología SCK, al conjunto de instancias que representan las métricas del modelo SCOR. De este modo, el usuario puede elegir la métrica que le resulte más adecuada, del conjunto de instancias de métricas del modelo SCOR. Por medio de reglas SWRL, se infieren las relaciones entre los conceptos asociados, a la descomposición de la métrica seleccionada. A continuación, se presenta un ejemplo de los conceptos asociados, los cuales se instancian para una métrica puntual. La métrica con nombre *Delivery Performance to Customer Commit Date* con codificación de métrica *RL.2.2*, tiene como fórmula a: "*[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%"*. Dicha fórmula se descompone en dos variables complejas, las cuales son: *Customer Commit Date Achievement* y *Delivery Location*. Las variables tienen asociadas las métricas con codificación de métrica *RL.3.32* y *RL.3.34*, respectivamente. Por un lado, la métrica con codificación *RL.3.32* tiene como fórmula a: "*[Total number of orders received on time] / [Total number of orders delivered] x 100%*", que se descompone en las variables atómicas *Total Delivered* y *Received On Time*. Por otro lado, la métrica con codificación *RL.3.34* tiene como fórmula a: "*[Total number of orders delivered on the correct location] / [Total number of orders delivered] x 100%*", la cual se descompone en las variables atómicas *Total Delivered* y *Delivered On Correct Location*.

Del mismo modo que se han instanciado los conceptos para la métrica con codificación de métrica *RL.2.2*, se instancian los conceptos de todas las demás métricas del modelo SCOR. Como se puede observar en el Anexo A (Métricas del modelo SCOR), la información de la descomposición de la métrica se corresponde con la información descripta en el modelo SCOR.

3.3.1.1 Especificación de Reglas SWRL

Además de los conceptos, atributos, y relaciones detalladas, el modelo semántico incluye un conjunto de reglas SWRL. Estas reglas tienen como objetivo, garantizar que la estructura de cadena de suministro contenga conceptos, relaciones, y atributos válidos. A continuación, se definen cada una de las reglas SWRL que se incluyen en el modelo semántico, para garantizar dicho objetivo.

Al momento de medir un atributo de rendimiento se selecciona una métrica, la cual se identifica por medio de su nombre. Además cada métrica posee una codificación de métrica, que permite su indexación en el modelo de referencia SCOR. Entonces, para asegurar que cada métrica se asocie a su codificación de métrica correspondiente, se define la regla:

"*Sea una métrica M con nombre N, entonces su codificación de métrica es ID*"

La Ecuación 3.1 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para la métrica con nombre *Perfect Order Fulfillment*, en donde la codificación de métrica es *RL.1.1*. En este ejemplo, el término *Metric* representa el concepto de métrica, la variable *?m* es un individuo (una instancia del concepto *Metric*), el término *metricName* vincula una métrica con la propiedad nombre de métrica y, por último, el término *metricID* asocia una métrica con la propiedad codificación de métrica. De esta manera, la regla garantiza que: la instancia del concepto *Metric* con nombre de métrica *Perfect Order Fulfillment*, se vinculará con la codificación de métrica *RL.1.1*. Además se incorporaron restricciones similares, a efectos de garantizar las vinculaciones entre las métricas del modelo SCOR y las codificaciones de métricas asociadas.

$$Metric(?m) \wedge metricName(?m, \text{“Perfect Order Fulfillment”}) \rightarrow metricID(?m, \text{“RL.1.1”}) \quad \text{(Ecuación 3.1)}$$

Una métrica es una forma de medición estándar, que posee una fórmula para su cálculo. Entonces, para asegurar que cada métrica se asocie a su fórmula correspondiente, se define la regla:

"*Sea una métrica M con codificación de métrica ID y sea una fórmula F con ecuación E, entonces la métrica M tiene como fórmula a F*"

La Ecuación 3.2 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para la relación "tiene como fórmula" (*hasFormula*) en el caso de la métrica con codificación de métrica *RL.1.1*, y la fórmula con ecuación *[Total Perfect Orders] / [Total Number of Orders] x 100 %*. En este ejemplo, los términos *Metric* y *Formula* representan los conceptos de métrica y fórmula, respectivamente; las variables *?m* y *?f* son individuos del tipo *Metric* y *Formula*, respectivamente; el término *metricID* asocia una métrica con la propiedad codificación de métrica, el término *equation* vincula una fórmula con la propiedad ecuación, y el término *hasFormula* relaciona a una métrica con una fórmula. De esta manera, la regla garantiza que: la instancia del concepto *Metric* con codificación de métrica *RL.1.1*, y la instancia del concepto *Formula* con ecuación *[Total Perfect Orders] / [Total Number of Orders] x 100 %*, se vincularán por medio de la relación *hasFormula*. Restricciones similares se incorporaron, a fin de reflejar las asociaciones entre el conjunto de métricas del modelo SCOR y las fórmulas correspondientes.

Metric(?m) ^ Formula(?f) ^ equation(?f, "[Total Perfect Orders] / [Total Number of Orders] x 100 %") ^ metricID(?m, "RL.1.1") -> hasFormula(?m,?f) (Ecuación 3.2)

Una formula posee una ecuación para su cálculo y además una unidad, la cual se deriva de la ecuación asociada. Entonces, para asegurar que cada fórmula se asocie a su unidad correspondiente, se define la regla:

"*Sea una fórmula F con ecuación E, entonces la unidad de la fórmula es U*"

La Ecuación 3.3 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para la fórmula con ecuación *[Total Perfect Orders] / [Total Number of Orders] x 100 %*, en donde la unidad de fórmula es *%*. En este ejemplo, el término *Formula* representa el concepto de fórmula, la variable *?f* es un individuo instancia del concepto *Formula*, el término *equation* vincula a una fórmula con la propiedad ecuación, y el término *formulaUnit* asocia a

una fórmula con la propiedad unidad de fórmula. De esta manera, la regla garantiza que: la instancia del concepto *Formula* con ecuación *[Total Perfect Orders] / [Total Number of Orders] x 100 %*, se vinculará con la unidad de fórmula *%*. Restricciones similares se incorporaron, a efectos de garantizar las vinculaciones entre las fórmulas y las unidades de fórmulas asociadas.

Formula(?f) ^ equation(?f, "[Total Perfect Orders] / [Total Number of Orders] x 100 %") -> formulaUnit(?f, "%") (Ecuación 3.3)

Una métrica posee una codificación de métrica para su identificación, a partir del cual se deriva su nivel. Entonces, para asegurar que cada métrica se asocie a su nivel correspondiente, se define la regla:

"*Sea una métrica M con codificación de métrica ID, entonces el nivel de la métrica es L*"

La Ecuación 3.4 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para las métricas del nivel 2 del atributo confiabilidad (*Reliability*). En este ejemplo, el término *Metric* representa el concepto de métrica, el término *metricID* asocia una métrica con la propiedad codificación de métrica, la variable *?m* es un individuo instancia de *Metric*, la variable *?id* es un individuo del tipo *string*, el término *matches* vincula al individuo *id* con una expresión regular, que representa a las codificaciones de métricas de nivel 2 para el atributo *Reliability*; y el término *metricLevel* asocia a una métrica con la propiedad nivel de métrica. De esta manera, la regla garantiza que: la instancia del concepto *Metric* que sea de nivel 2 y mida al atributo *Reliability*, se vinculará con el nivel de métrica dos. Restricciones similares se incorporaron, a fin de reflejar las asociaciones entre las métricas del modelo SCOR y los niveles correspondientes.

Metric(?m) ^ metricID(?m,?id) ^ swrlb:matches(?id, "^(RL)\\.2\\.[1-4]$") -> metricLevel (?m,2) (Ecuación 3.4)

Una fórmula posee una ecuación para su cálculo. Entonces, para asegurar que cada fórmula se asocie a sus variables correspondientes, se define la regla:

"*Sea una fórmula F con ecuación E y sea una/s variable/s con nombre/s definido/s, entonces la fórmula F tiene como variable a la/s variable/s"*

La Ecuación 3.5 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para la relación "tiene como variable" (*hasVariable*) en el caso de la fórmula con ecuación *Inventory Days Supply + Days Sales Outstanding – Days Payable Outstanding*, y las variables complejas: A con nombre *Inventory Days Supply*, B con nombre *Days Sales Outstanding*, y C con nombre *Days Payable Outstanding*. En este ejemplo, los términos *Formula* y *ComplexVariable* representan el concepto de fórmula y variable compleja, respectivamente; la variable *?f* es un individuo instancia de *Formula*, las variables *?a*, *?b*, y *?c* son individuos del tipo *ComplexVariable*; el término *equation* vincula una fórmula con la propiedad ecuación, el término *variableName* asocia una variable con la propiedad nombre de variable, y el término *hasVariable* relaciona a una fórmula con una variable. De esta manera, la regla garantiza que: la instancia del concepto *Formula* con ecuación *Inventory Days Supply + Days Sales Outstanding – Days Payable Outstanding*, y las instancias del concepto *ComplexVariable* con nombres *Inventory Days Supply, Days Sales Outstanding* y *Days Payable Outstanding*; se vincularán por medio de la relación *hasVariable*. Restricciones similares se incorporaron, a efectos de garantizar las vinculaciones entre las fórmulas y las variables correspondientes.

Formula(?f) ^ ComplexVariable(?a) ^ ComplexVariable(?b)
^ ComplexVariable(?c) ^ equation(?f, "Inventory Days Supply + Days Sales Outstanding – Days Payable Outstanding")
^ variableName(?a, "Inventory Days Supply")
^ variableName(?b, "Days Sales Outstanding")
^ variableName(?c, "Days Payable Outstanding")
-> hasVariable(?f,?a) ^ hasVariable(?f,?b) ^ hasVariable(?f,?c)

(Ecuación 3.5)

Una variable compleja requiere una métrica para su cálculo. Entonces, para asegurar que cada variable compleja se asocie a su métrica correspondiente, se define la regla:

"*Sea una variable compleja VC con nombre N y sea una métrica M con nombre MN, entonces la variable compleja VC tiene como métrica a M"*

La Ecuación 3.6 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para la relación "tiene como métrica" (*hasMetric*) en el caso de la variable compleja con nombre *Inventory Days Supply*, y la métrica con nombre *Inventory Days Supply*. En este ejemplo, los términos *Metric* y *ComplexVariable* representan el concepto de métrica y variable compleja, respectivamente; las variables *?vc* y *?m* son individuos instancias de *ComplexVariable* y *Metric*, respectivamente; el término *variableName* asocia una variable con la propiedad nombre de variable, el término *metricName* vincula una métrica con la propiedad nombre de métrica, y el término *hasMetric* relaciona a una variable compleja con una métrica. De esta manera, la regla garantiza que: la instancia del concepto *ComplexVariable* con nombre *Inventory Days Supply*, y la instancia del concepto *Metric* con nombre *Inventory Days Supply*, se vincularán por medio de la relación *hasMetric*. Restricciones similares se incorporaron, a fin de reflejar las asociaciones entre las variables complejas y las métricas correspondientes.

$$\begin{aligned} &ComplexVariable(?vc) \wedge Metric(?m) \\ &\wedge variableName(?vc, \text{“Inventory Days Supply”}) \\ &\wedge metricName(?m, \text{“Inventory Days Supply”}) \rightarrow \\ &hasMetric(?vc,?m) \end{aligned} \qquad \text{(Ecuación 3.6)}$$

Un proceso se compone de subprocesos que son del mismo tipo, pero de diferente nivel. Entonces, para asegurar que cada proceso se descompone en los subprocesos correspondientes, se define la regla:

"*Sea una proceso P con codificación de proceso ID y sea un/os subproceso/s con codificación de proceso definida, entonces el proceso P tiene como subproceso a el/los subproceso/s"*

La Ecuación 3.7 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para la relación "tiene como subproceso" (*hasSubprocess*) en el caso del proceso con codificación de proceso *sM*, y los subprocesos A, B, y C con codificación de proceso *sM1*, *sM2*, y *sM3*, respectivamente. En este ejemplo, los términos *Process* y *SubProcess* representan el concepto de proceso y subproceso, respectivamente; la variable *?p* es un individuo instancia de *Process*, las variables *?a*, *?b*, y *?c*, son individuos del tipo *SubProcess*; el término *processID* vincula un proceso de negocio con la propiedad codificación de proceso, y el término *hasSubprocess* relaciona un proceso con un subproceso. De esta manera, la regla garantiza que: la instancia del concepto *Process* con codificación de proceso *sM*, y las instancias del concepto *SubProcess* con codificaciones de proceso *sM1*, *sM2*, y *sM3*, se vincularán por medio de la relación *hasSubprocess*. Restricciones similares se incorporaron, a efectos de garantizar las vinculaciones entre los procesos y los subprocesos correspondientes.

Process(?p) ^ SubProcess(?a) ^ SubProcess(?b)
^ SubProcess(?c) ^ processID(?p, "sM")
^ processID(?a, "sM1")
^ processID(?b, "sM2") (Ecuación 3.7)
^ processID(?c, "sM3")
-> hasSubprocess(?p,?a) ^ hasSubprocess(?p,?b)
^ hasSubprocess(?p,?c)

Un subproceso se compone de tareas que son del mismo tipo, pero de diferente nivel. Entonces, para asegurar que cada subproceso se descompone en las tareas correspondientes, se define la regla:

"*Sea un subproceso SB con codificación de proceso ID y sea una/s tarea/s con codificación de proceso definida, entonces el subproceso SB tiene como tarea a la/s tarea/s"*

La Ecuación 3.8 presenta, a modo de ejemplo, la regla detallada en lenguaje SWRL para relación "tiene como tarea" (*hasTask*) en el caso del subproceso con codificación de proceso *sP1*, y las tareas A, B, C, y D con codificación de proceso *sP1.1*, *sP1.2, sP1.3*, y *sP1.4*, respectivamente. En este ejemplo, los términos *SubProcess* y *Task* representan el concepto de subproceso y tarea, respectivamente; la variable *?sb* es un individuo instancia de *SubProcess*, las variables *?a*, *?b*, *?c*, y *?d*, son individuos del tipo *Task*; el término *processID* vincula un proceso de negocio con la propiedad codificación de proceso, y el término *hasTask* relaciona un subproceso con una tarea. De esta manera, la regla garantiza que: la instancia del concepto *SubProcess* con codificación de proceso *sP1*, y las instancias del concepto *Task* con codificaciones de proceso *sP1.1*, *sP1.2*, *sP1.3*, y *sP1.4*, se vincularán por medio de la relación *hasTask*. Restricciones similares se incorporaron, a efectos de garantizar las vinculaciones entre los subprocesos y las tareas correspondientes.

$$\begin{aligned} &SubProcess\ (?sb) \wedge Task\ (?a) \wedge Task\ (?b) \\ &\wedge Task\ (?c) \wedge Task(?d) \wedge processID(?sb, \text{“sP1”}) \\ &\wedge processID(?a, \text{“sP1.1”}) \wedge processID(?b, \text{“sP1.2”}) \\ &\wedge processID(?c, \text{“sP1.3”}) \wedge processID(?d, \text{“sP1.4”}) \\ &\rightarrow hasTask(?sb,?a) \wedge hasTask\ (?sb,?b) \\ &\wedge hasTask\ (?sb,?c) \wedge hasTask\ (?sb,?d) \end{aligned} \qquad \text{(Ecuación 3.8)}$$

A través de los 20 conceptos, 18 relaciones, 25 atributos, y las reglas SWRL definidas; el modelo semántico de la ontología SCK puede generar instancias de los conceptos asociados, a la definición de la estructura de cadena de suministro.

3.3.2 Ontología HLAFed

En la sección 2.2 (Simulación Paralela y Distribuida) se estableció qué, para hacer frente al problema de la interoperabilidad entre simuladores, es necesario garantizar que los eventos enviados por un simulador causen los efectos esperados, por quien diseña el modelo de simulación emisor de los eventos. En otras palabras, se pretende lograr una interpretación

unívoca de los eventos enviados por un simulador, para que de este modo, causen los efectos esperados. Además, en la sección 2.2.1 (Estándar IEEE 1516-2010 HLA Evolved) se presentó el estándar de la IEEE 1516-2010 HLA, que es el modelo más actual y completo que se encuentra en la literatura, para tratar el problema de la interoperabilidad entre simuladores. Entonces teniendo en cuenta estas aclaraciones, la ontología HLAFed define la estructura de una federación HLA (como parte de la red de ontologías SCFHLA), basándose en los conceptos de dicho estándar. Por medio del diseño de una ontología basada en el estándar HLA, se pretende lograr una especificación con toda la información necesaria, para la construcción del modelo de objetos de federación.

La Tabla 3.7 resume el conjunto de actividades que se realizan, sobre los elementos del estándar HLA, con el fin de obtener los componentes de la ontología HLAFed.

Cada uno de los conceptos identificados para la conformación de una federación, la gestión de los servicios del RTI, y la construcción de un modelo de objetos, se transforma en un concepto de la ontología con el mismo nombre. Por ejemplo: para conformar una federación, un federado debe iniciar la federación e invitar al resto de los federados, a que se unan a la misma. De este modo, el concepto de federado es central en la conformación de una federación, y se transforma en el concepto *Federate* de la ontología HLAFed (Actividad 1).

Además de estos conceptos, se identifican términos relevantes en el estándar que no están definidos como conceptos. Entonces cada uno de ellos, se transforma en un concepto de la ontología. Por ejemplo: existe solo un federado encargado de iniciar la federación, pero en el estándar solo se describe que un federado debe cumplir esta función, sin mayores especificaciones. Es por este motivo, que se añade el concepto *Administrator* a la ontología HLAFed (Actividad 2).

Tabla 3.7 – Actividades de mapeo entre el estándar HLA y los componentes de la ontología

Actividad #	Descripción	Tipo de Componente	Ejemplo
1	Cada concepto identificado para la conformación de una federación, la gestión de los servicios del RTI, y la construcción de un modelo de objetos, se transforma en un concepto de la ontología	Concepto	El concepto de federado se transforma en el concepto *Federate*
2	Cada termino relevante identificado, se transforma en un concepto de la ontología, con el mismo nombre	Concepto	Solo un federado tiene la función de iniciar la federación. Dicha función se añade a la ontología, como el concepto *Administrator*
3	Se definen las relaciones entre los conceptos identificados	Relación	Se define la relación *hasFOM*, para vincular al concepto *Federation* con el concepto *FOM*
4	Cada modelo de objeto clasificado según quien lo utilice, se transforma en un concepto	Concepto	El modelo de objetos utilizado por la federación, se transforma en el concepto *FOM*
5	La jerarquía de modelos de objetos, se define mediante la relación *isA*	Relación	Se define la relación *isA*, entre el concepto *SOM* y el concepto *ObjectModel*
6	Un *ObjectClass* puede especializarse en otros objetos, según el nivel de detalle requerido. Entonces está relación, se transforma en la relación *hasSubclass*	Relación	La clase de objeto vehículo, posee la relación *hasSubclass* con las clases de objetos avión y camión, según si el transporte es por vía aérea o terrestre

Actividad #	Descripción	Tipo de Componente	Ejemplo
7	Un *InteractionClass* puede especializarse en otras interacciones, según el nivel de detalle requerido. Entonces está relación, se transforma en la relación *hasSpecialization*	Relación	La clase de interacción enviar-suministro posee la relación *hasSpecialization*, con las clases de interacción enviar-producto-uno y enviar-producto-dos; si es necesario especificar la relación por el producto final a entregar
8	Se añaden los atributos, para definir las propiedades específicas de los conceptos de la ontología	Propiedad	Se añade el atributo *objectiveDescription* al concepto *Objective*, para modelar la propiedad descripción del concepto Objetivo

Los conceptos identificados y añadidos al modelo semántico poseen vínculos entre ellos, que se definen en la ontología como relaciones. Por ejemplo: antes de iniciar la simulación de una federación es necesario determinar el modelo de objetos de federación, el cual contiene la información a intercambiar dentro de la simulación. Entonces, se crea la relación *hasFOM* entre los conceptos *Federation* y *FOM*, para representar la necesidad de una federación, de contar con un modelo de objetos de federación al momento de ejecutar su simulación (Actividad 3).

Los modelos de objetos del estándar HLA comparten información común, y se diferencian según quien los utilice, si es un simulador (federado) o una federación. A partir de esta distinción, el modelo de objetos se refina en los conceptos de: *SOM* y *FOM*, respectivamente. Ambos conceptos se añaden a la ontología, con su correspondiente nombre (Actividad 4). Para representar esta relación de especialización, los dos modelos de objetos (SOM y FOM) se asocian al concepto denominado *ObjectModel*, por medio de la relación *isA*. De este modo, por ejemplo: el concepto *SOM* se vincula con el concepto *ObjectModel*, por medio de la relación *is-a* (Actividad 5).

Una clase de objeto (concepto *ObjectClass*) que forma parte de un modelo de objeto (concepto *ObjectModel*), tiene un vínculo con sí misma, para representar que se puede descomponer en otras clases de objeto, según el nivel de detalle requerido. Esta relación se transforma en la relación *hasSubclass*. Por ejemplo: la clase de objeto vehículo se puede especializar en las subclases avión o camión, dependiendo de si el transporte es, por vía aérea o terrestre (Actividad 6).

Una clase de interacción (concepto *InteractionClass*) que forma parte de un modelo de objeto, tiene un vínculo con sí misma, para representar que se puede refinar en otras clases de interacción, según el nivel de detalle requerido. Esta relación se transforma en la relación *hasSpecialization*. Por ejemplo: la clase de interacción enviar-suministro se puede especializar en las subclases enviar-producto-uno o enviar-producto-dos, si es necesario especificar la relación por el producto final a suministrar (Actividad 7).

Una vez que se han especificado todos los conceptos y relaciones, necesarios para representar los elementos propuestos en el estándar HLA, se incorporan atributos para definir las propiedades específicas de los conceptos de la ontología. Por ejemplo: el concepto *Objective* representa el objetivo que desea alcanzar la federación, entre sus propiedades se destacan: la descripción (atributo *objectiveDescription*) en lenguaje natural que hace uso de cadenas de caracteres (tipo de dato *string*), y la condición de parada para la simulación (atributo *stopCondition*) que hace uso de cadenas de caracteres (tipo de dato *string*) (Actividad 8).

Como resultado de las actividades realizadas, se obtiene el modelo presentado en la Figura 3.2.

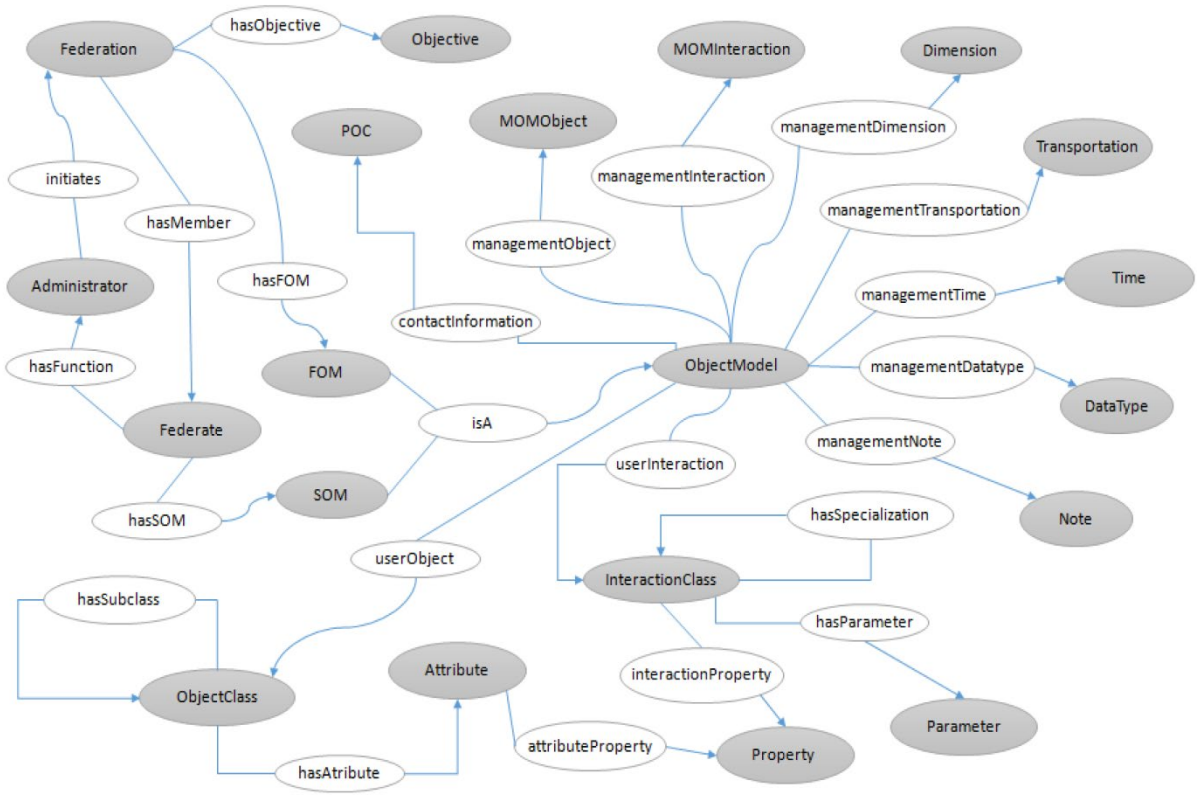


Figura 3.2 – Ontología HLAFed

Como se observa en la Figura 3.2, una federación se modela por medio del concepto *Federation*, mientras que un federado se modela a través del concepto *Federate*. Para representar que una federación, se constituye de un conjunto de federados, se define la relación *hasMember* entre *Federation* y *Federate*. Para crear una federación, existe un único federado que cumple la función de iniciar la misma, e invitar a los demás federados participantes a unirse a ella. En la ontología se define el concepto *Administrator*, para representar al federado encargado de cumplir dicha tarea. Se modela la relación *initiates* entre *Administrator* y *Federation*, para representar cual es la federación que inicia el federado. Además, se modela la relación *hasFunction* entre *Federate* y *Administrator*, para indicar qué federado cumple el rol de iniciador de la federación. El concepto *Federation* representa una simulación distribuida y, por lo tanto, necesita tener asociado una meta a cumplir. De este modo, en la ontología se define el concepto *Objective* para representar el objetivo de la

federación, y además se modela la relación *hasObjective* entre el concepto *Federation* y *Objective*, a efectos de indicar que la federación persigue dicho objetivo.

Como se explicó con anterioridad, los federados y federaciones tienen sus modelos de objetos, para definir qué información sobre los datos y qué interacciones, se intercambian durante la simulación distribuida. Estas definiciones, constituyen la base fundamental del éxito de la simulación distribuida. El concepto *ObjectModel* se define para representar al modelo de objetos, el cual se especializa en los conceptos *FOM* y *SOM*. Este refinamiento del concepto *ObjectModel* se basa en quien utiliza el modelo de objetos, de este modo el concepto *FOM* representa el modelo de objetos asociado a la federación, y el concepto *SOM* representa el modelo de objetos asociado a cada federado. Dado que tanto las federaciones, como los federados, deben tener asociado un FOM o SOM según corresponda, en la ontología se definen las relaciones *hasFOM* y *hasSOM*; para vincular a una federación con su FOM, y a un federado con su SOM, respectivamente.

Un modelo de objetos, abstrae la estructura que debe tener tanto el SOM, como el FOM. Dentro de esta estructura, se encuentra la información de gestión de los servicios del RTI, la información definida por el usuario, y el responsable del modelo de objetos. En cuanto a este último punto, se define el concepto *POC* (por sus siglas en inglés *–Point of Contact-*) para representar la información de contacto, del responsable del modelo de objetos. En el caso de la federación, esta información se asocia con el administrador de la misma, es decir, quien la inicia e invita al resto de los integrantes. En tanto que para el caso de los federados, esta información se asocia con quien diseña el modelo de objetos del federado. La relación *contactInformation* se define para relacionar el concepto de *POC*, con el concepto de *ObjectModel*.

Con respecto a la información definida por el usuario, la misma puede ser de dos tipos: clase de interacción o clase de objeto. Se definen los conceptos de *InteractionClass* y *ObjectClass*, para representar dicha información. Además, se modelan las relaciones *userInteraction* y *userObject*, para vincular cada concepto con el modelo de objetos respectivamente. Por un lado, una clase de objeto que forma parte de un modelo de objetos,

tiene un vínculo con sí misma, para representar que se puede especializar en otras clases de objeto, según el nivel de detalle requerido. Esta relación se modela mediante la relación *hasSubclass*. Por otro lado, una clase de interacción que forma parte de un modelo de objeto, tiene un vínculo con sí misma, para representar que se puede descomponer en otras clases de interacción. De esta forma se establece así, una jerarquía de especialización según el nivel de detalle requerido. Esta relación, se representa como la relación *hasSpecialization*.

Además de los vínculos consigo mismo, el concepto *ObjectClass* puede tener asociado atributos (concepto *Attribute*), que representan las características de la clase de objeto. Para representar el vínculo entre *ObjectClass* y *Attribute*, se define la relación *hasAttribute*.

En el contexto de SD basada en HLA de CS, una federación representa una estructura de CS que se va a evaluar, desde la perspectiva de rendimiento. Para llevar a cabo esta evaluación, en el contexto de esta tesis, se utilizan las métricas del modelo SCOR. Estas métricas se descomponen hasta su mínima expresión (que es una variable atómica), la cual se modela en la ontología SCK con el concepto de *AtomicVariable*. Por medio de estas variables, un federado puede calcular la métrica seleccionada para la evaluación del rendimiento de CS. Para calcular el valor de la métrica, el federado debe recibir notificaciones sobre el valor y la unidad de este tipo de variables. Dichas notificaciones, se encuentran especificadas como las propiedades valor (modelado con el atributo *atomicVaribleValue*) y unidad (modelado con el atributo *atomicVaribleUnit*) del concepto *AtomicVariable*, respectivamente. Las dos propiedades de cada variable atómica, deben estar definidas en el modelo de objetos de federación, a efectos de que se pueda notificar su contenido a los federados. Por este motivo, se define como parte de la ontología a dos instancias del concepto *Attribute*, denominadas como *value* y *unit*.

Un atributo posee propiedades (concepto *Property*) que representan el tipo de dato, quien es su dueño, y como se transporta el atributo, entre otros aspectos. Para relacionar un atributo con sus propiedades, se define la relación *attributeProperty*. Como se mencionó con anterioridad, las instancias *value* y *unit* se definen como parte de la ontología. A efectos de representar las propiedades de estos individuos, se define como parte de la ontología a dos

instancias del concepto *Property*, denominadas *valueProperty* y *unitProperty* respectivamente. Las propiedades tienen definido el tipo de dato que se asocia con el atributo, por lo que la instancia *valueProperty* posee como tipo de dato, el valor *HLAfloat64LE* (tipo de dato *float* de HLA, de 64 bits con codificación *Little endian*); mientras que la instancia *unitProperty* posee como tipo de dato (atributo *propertyDatatype*), el valor *HLAunicodeString* (tipo de dato *string* de HLA, con codificación *unicode*).

Las clases de interacción (concepto *InteractionClass*) pueden tener asociadas parámetros (concepto *Parameter*), los cuales representan la información que contiene la clase de interacción. Para relacionar los conceptos de *InteractionClass* y *Parameter*, se define la relación *hasParameter* en la ontología. Además de tener la posibilidad de utilizar parámetros, las clases de interacción poseen propiedades, al igual que los atributos, que representan los mismos aspectos que para los atributos. Para modelar el vínculo entre los conceptos *InteractionClass* y *Property*, se define la relación *interactionProperty* en la ontología. De acuerdo con (Topçu y Oğuztüzün 2017; 2013; Uygun, Öztemel, y Kubat 2009) las clases de interacción tienen un conjunto de propiedades sugeridas, que se utilizan en la mayoría de los casos; es decir, que las mismas se consideran como propiedades por defecto. Por este motivo, se define como parte de la ontología a la instancia del concepto *Property* denominada *icProperty*. Este individuo tiene las siguientes características: como forma de compartir (atributo *propertySharing*) el valor *PublishSubscribe* (se comparte la interacción en modo publicar y suscribir), como modo de transporte (atributo *transportation*) el valor *HLAreliable* (se utiliza un modo de transporte confiable), y como forma de ordenar la entrega (atributo *order*) el valor *TimeStamp* (la interacción se entrega ordenada por la marca temporal).

En cuanto a la información correspondiente a la administración de los servicios prestados por el RTI durante una simulación, se distinguen dos grupos de conceptos: uno relacionado al valor de ciertos parámetros del FOM, para modificar la configuración preestablecida; y otro grupo, relacionado con la gestión de las comunicaciones con el RTI.

Del primer grupo, el concepto *DataType* representa las características de los tipos de datos que se utilizan, tanto por las interacciones como por los objetos del modelo de objetos.

El concepto *Note* agrega información adicional al modelo de objetos, de forma tal de facilitar el uso de los datos específicos. El concepto *Transportation* define diferentes mecanismos, para transportar los datos entre federados. El concepto *Time*, conceptualiza las diferentes estrategias de administración del tiempo de cada federado. El concepto *Dimension* corresponde a la organización, de un grupo de objetos en dimensiones.

Del segundo grupo (el de gestión de comunicaciones con el RTI), el concepto *MOMObject* representa grupos de objetos y sus atributos, los cuales se utilizan para describir el estado de la federación, y de los federados. Finalmente, *MOMInteraction* representa el conjunto de servicios usados por un federado, para: adaptar, requerir, reportar, o invocar, servicios en nombre de otro federado.

Para cada uno de los conceptos mencionados de ambos grupos, se definen las relaciones necesarias para asociar dichos conceptos con el concepto *ObjectModel*. Esta relación es necesaria, para indicar que todos ellos forman parte de dicho modelo.

3.3.2.1 Especificación de Reglas SWRL

Además de los conceptos, atributos, y relaciones detalladas, el modelo semántico incluye un conjunto de reglas SWRL. Estas reglas tienen como objetivo, garantizar que la federación HLA contenga conceptos, relaciones, y atributos válidos. A continuación, se definen cada una de las reglas SWRL que se incluyen en el modelo semántico, para alcanzar dicho objetivo.

Cada federación debe tener asociado un modelo de objetos de federación, a efectos de poder realizar su simulación. Entonces, para asegurar que cada federación se vincule a un modelo de objetos, con el mismo nombre que la federación, se define la regla:

"*Sea una federación FN con nombre N y sea un fom F con nombre N, entonces FN tiene como fom a F*"

La Ecuación 3.9 formaliza la regla en el lenguaje SWRL, en donde las variables *?fn* y *?f*, son individuos del tipo *Federation* y *FOM* respectivamente; la variable *?n* es el nombre de

la federación y del modelo de objetos; los términos *Federation* y *FOM* representan los conceptos de federación y fom, respectivamente; el término *federationName* y *objectModelName*, vinculan una federación con la propiedad nombre de federación, y un modelo de objetos con la propiedad nombre de modelo de objetos, respectivamente; y, por último, el término *hasFOM* relaciona una federación con un fom. De esta manera, la regla garantiza que: cualquier instancia del concepto *Federation* con su atributo nombre definido, se vinculará con una instancia del concepto *FOM*, que posea el mismo nombre que la federación, por medio de la relación *hasFOM*.

$$Federation(?fn) \wedge federationName(?fn,?n) \wedge FOM(?f) \wedge objectModelName(?f,?n) \rightarrow hasFOM(?fn,?f) \quad \text{(Ecuación 3.9)}$$

Cada clase de objeto definida por el usuario, debe estar contenida en un modelo de objetos de federación, el cual se encuentra asociado a una federación. Entonces, para asegurar que cada clase de objeto se asocie a su modelo de objetos correspondiente, se define la regla:

"*Sea una clase de objeto O, sea un fom F, sea la federación FN y FN tiene como fom a F, entonces F tiene como clase de objeto a O*"

La Ecuación 3.10 formaliza la regla en el lenguaje SWRL, en donde las variables *?o*, *?f*, y *?fn*, son individuos del tipo *ObjectClass*, *FOM*, y *Federation* respectivamente; los términos *ObjectClass*, *FOM*, y *Federation* representan los conceptos de clase de objeto, fom, y federación respectivamente; el término *hasFOM* relaciona una federación con un fom; y el término *userObject* vincula una clase de objeto con un fom. De esta manera, la regla garantiza que: una instancia del concepto *FOM* que se vincula con una federación por medio de la relación *hasFOM,* se vinculará con cualquier instancia del concepto *ObjectClass*, a través de la relación *userObject*.

$$ObjectClass(?o) \wedge FOM(?f) \wedge Federation(?fn) \wedge hasFOM(?fn,?f) \rightarrow userObject(?f,?o) \quad \text{(Ecuación 3.10)}$$

Cada clase de interacción definida por el usuario, debe estar contenida en un modelo de objetos de federación, el cual se encuentra asociado a una federación. Entonces, para asegurar que cada clase de interacción se asocie a su modelo de objetos correspondiente, se define la regla:

"*Sea una clase de interacción CI, sea un fom F, sea la federación FN y FN tiene como fom a F, entonces la F tiene como clase de interacción a CI*"

La Ecuación 3.11 formaliza la regla en el lenguaje SWRL, en donde las variables *?ci*, *?f*, y *?fn*, son individuos del tipo *InteractionClass*, *FOM*, y *Federation* respectivamente; los términos *InteractionClass*, *FOM*, y *Federation* representan los conceptos de clase de interacción, fom, y federación respectivamente; el término *hasFOM* relaciona una federación con un fom; y el término *userInteraction* vincula una clase de interacción con un fom. De esta forma, la regla garantiza que: una instancia del concepto *FOM* que se vincula con una federación por medio de la relación *hasFOM*, se vinculará con cualquier instancia del concepto *InteractionClass*, mediante la relación *userInteraction*.

$$InteractionClass(?ci) \wedge FOM(?f) \wedge Federation(?fn) \wedge hasFOM(?fn,?f) \rightarrow userInteraction(?f,?ci) \quad \text{(Ecuación 3.11)}$$

Cada clase de objeto definida por el usuario posee dos atributos, cada uno de los cuales a su vez, posee un conjunto de propiedades asociadas. Entonces, para asegurar que cada atributo se asocie a sus propiedades correspondientes, se define la regla:

"*Sea la clase de objeto O, el atributo X con nombre value, el atributo Y con nombre unit, O tiene como atributos a X e Y, sea la propiedad P con nombre valueProperty y tipo de dato numérico, sea la propiedad W con nombre unitProperty y tipo de dato cadena de caracteres, entonces X tiene como propiedad a P e Y tiene como propiedad a W*"

La Ecuación 3.12 formaliza la regla en el lenguaje SWRL, en donde las variables *?o*, *?x*, *?y*, *?p*, y *?w*, son individuos del tipo *ObjectClass*, *Attribute*, *Attribute*, *Property*, y *Property* respectivamente; los términos *ObjectClass*, *Attribute*, y *Property* representan los conceptos de la ontología con el mismo nombre; el término *attributeName* vincula un atributo

con la propiedad nombre de atributo, el término *hasAttribute* asocia a una clase de objeto con un atributo; el término *propertyName* relaciona una propiedad con el nombre de propiedad, el término *propertyDatatype* vincula una propiedad con el tipo de dato de propiedad, y el término *attributeProperty* asocia a un atributo con una propiedad. De esta manera, la regla garantiza que: las instancias del concepto *Attribute* denominadas *value* y *unit*, se vincularán por medio de la relación *attributeProperty*, con las instancias del concepto *Property* denominadas *valueProperty* y *unitProperty*, respectivamente.

$$\begin{aligned} &ObjectClass(?o) \wedge Attribute(?x) \wedge attributeName(?x,\text{“ value”}) \wedge \\ &Attribute(?y) \wedge attributeName(?y,\text{“unit”}) \wedge hasAttribute(?o,?x) \\ &\wedge hasAttribute(?o,?y) \wedge Property(?p) \wedge Property(?w) \\ &\wedge propertyName(?p,\text{“valueProperty”}) \\ &\wedge propertyName(?w,\text{“ unitProperty”}) \\ &\wedge propertyDatatype(?p,\text{“HLAfloat64LE”}) \\ &\wedge propertyDatatype(?w,\text{“ HLAunicodeString”}) \\ &\rightarrow attributeProperty(?x,?p) \\ &\wedge attributeProperty(?y,?w) \end{aligned} \qquad \text{(Ecuación 3.12)}$$

Cada clase de interacción definida por el usuario, posee un conjunto de propiedades asociadas. Entonces, para asegurar que cada clase de interacción se vincule con sus propiedades correspondientes, se define la regla:

“*Sea una clase de interacción CI, sea la propiedad P con nombre icProperty, forma de compartir PublishSubscribe, modo de transporte HLAreliable, y forma de ordenar la entrega TimeStamp, entonces CI tiene como propiedad a P”*

La Ecuación 3.13 formaliza la regla en el lenguaje SWRL, en donde las variables *?ci* y *?p*, son individuos del tipo *InteractionClass* y *Property* respectivamente; los términos *InteractionClass* y *Property* representan los conceptos de la ontología con el mismo nombre; el término *propertyName* relaciona una propiedad con el nombre de propiedad, el término

propertySharing asocia una propiedad con una forma de compartir, el término *transportation* vincula una propiedad con un modo de transporte, el término *order* relaciona una propiedad con una forma de ordenar la entrega, y, por último, el término *interactionProperty* asocia una clase de interacción con una propiedad.

De esta manera, la regla garantiza que: todas las instancias del concepto *InteractionClass* se vincularán a través de la relación *interactionProperty*, con una instancia del concepto *Propiedad* denominada *icProperty*, con los valores correspondientes por defecto para las clases de interacción.

$$\begin{aligned} &InteractionClass(?ci) \wedge Property(?p) \\ &\wedge propertyName(?p, \text{“icProperty”}) \\ &\wedge propertySharing(?p,\text{“PublishSubscribe”}) \\ &\wedge transportation(?p,\text{“HLAreliable”}) \wedge order(?p,\text{“TimeStamp”}) \\ &\rightarrow interactionProperty(?ci,?p) \end{aligned} \qquad \text{(Ecuación 3.13)}$$

A través de los 20 conceptos, 22 relaciones, 37 atributos, y las reglas SWRL definidas; el modelo semántico de la ontología HLAFed puede generar instancias de los conceptos asociados, a la definición de una federación HLA.

3.3.3 Integración de la Red SCFHLA

La red de ontologías SCFHLA modela el contexto de simulación distribuida basada en HLA, de cadenas de suministro. La Figura 3.3 presenta un esquema de alto nivel de la red SCFHLA, en donde se identifican los dos dominios tenidos en cuenta, sus principales conceptos, y las meta relaciones entre las ontologías de la red. Las meta relaciones (Díaz et al. 2011) se representan con una línea punteada, mientras que las relaciones dentro de cada uno de los dominios se representan con líneas continuas

La ontología HLAFed conceptualiza las federaciones HLA. Así, los conceptos *Federation*, *Federate*, *ObjectModel*, *FOM*, *SOM*, *ObjectClass*, e *InteractionClass* hacen

referencia a los elementos de: federación, federado, modelo de objetos, modelo de objetos de federación, modelo de objetos de simulador, clases de objetos, y clases de interacciones respectivamente. Todos estos conceptos, son necesarios para conceptualizar una federación HLA. Por otro lado, la ontología SCK especifica los conceptos, tales como: *SupplyChain*, *Relation*, *Participant*, *Metric*, y *BusinessProcess* correspondientes a: cadena de suministro, relaciones, participante, métrica, y proceso de negocio respectivamente. Cada uno de estos conceptos, resulta indispensable para definir una estructura de cadena de suministro. Mediante la integración de las ontologías HLAFed y SCK en la red de ontologías, los analistas de negocio pueden especificar el modelo de interoperabilidad, para la SD basada en HLA de cadenas de suministro.

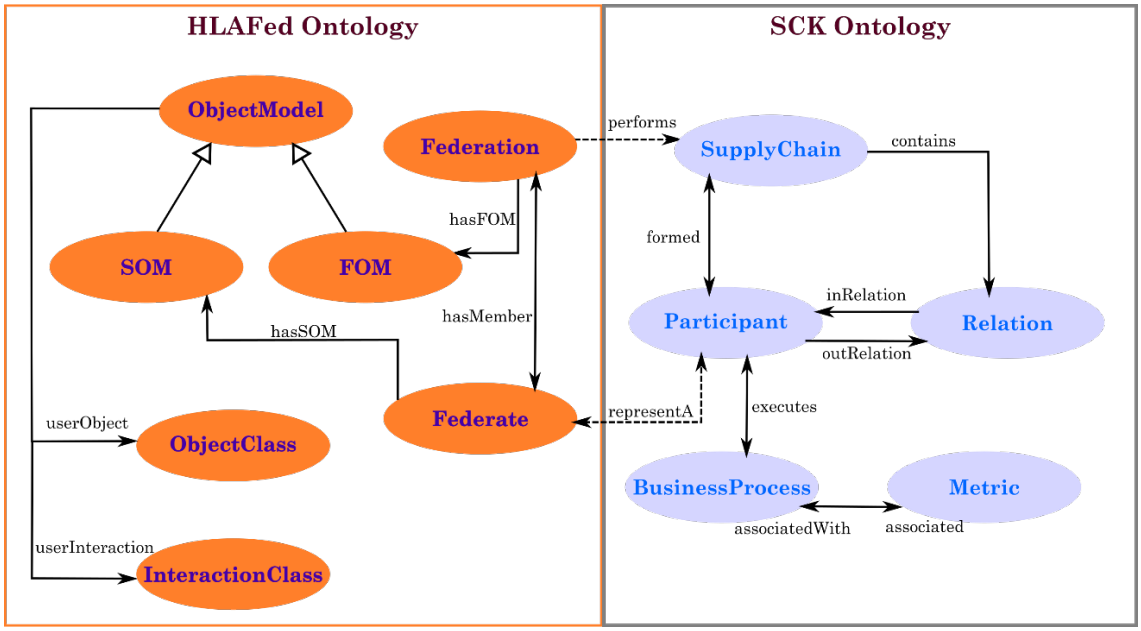


Figura 3.3 – Descripción de alto nivel de red SCFHLA

Los conceptos de federación (*Federate*) y cadena de suministro (*SupplyChain*), se vinculan por medio de la meta relación *performs*, dado que la red de ontologías se utiliza para modelar federaciones HLA que simulan la estructura de cadenas de suministro. Los conceptos de federado (*Federate*) y participante (*Participant*), se relacionan por medio de la meta relación *representA*, ya que cada federado representa el comportamiento de un participante de la estructura de cadena de suministro.

3.3.3.1 Especificación de Reglas SWRL

Además de los conceptos y relaciones detallados, la red de ontologías incluye un conjunto de reglas SWRL. Estas reglas tienen como objetivo, mapear los conceptos de una estructura de cadena de suministro, a los conceptos de una federación HLA. A continuación, se definen cada una de las reglas SWRL que se incluyen en la red de ontologías, con el fin de alcanzar dicho objetivo.

En el contexto de análisis, una cadena de suministro es representada por una federación; por lo tanto, a efectos de asegurar el mapeo entre ambos conceptos, se define la siguiente regla:

"*Sea la cadena de suministro CS con nombre N, entonces se vincula a CS con el concepto federación cuyo nombre de federación es N*"

La Ecuación 3.14 formaliza la regla en el lenguaje SWRL, en donde la variable *?cs* es una instancia del concepto *SupplyChain*, que a su vez se infiere como instancia del concepto *Federation*; la variable *?n* es el nombre asociado a la cadena de suministro, el cual a su vez se infiere también como el nombre de la federación; los términos *SupplyChain* y *Federation* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *supplyChainName* vincula una cadena de suministro con la propiedad nombre de CS, y el término *federationName* asocia una federación con la propiedad nombre de federación. De esta manera, la regla garantiza que: cualquier instancia del concepto *SupplyChain* con su nombre definido, se inferirá como una instancia del concepto *Federation* con el mismo nombre que la CS.

$$SupplyChain(?cs) \wedge supplyChainName(?cs,?n) \rightarrow Federation(?cs) \wedge federationName(?cs,?n) \quad \text{(Ecuación 3.14)}$$

Una federación representa a una cadena de suministro, entonces para asegurar que cada federación se asocie a su cadena de suministro correspondiente, se define la regla:

"*Sea una cadena de suministro CS con nombre N y sea una federación FN con nombre N, entonces la federación FN representa a la cadena de suministro CS*"

La Ecuación 3.15 formaliza la regla en el lenguaje SWRL, en donde las variables *?cs* y *?fn*, son individuos del tipo *SupplyChain* y *Federation* respectivamente; la variable *?n* es el nombre asociado, a la cadena de suministro y a la federación; los términos *SupplyChain* y *Federation* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *supplyChainName* vincula una cadena de suministro con la propiedad nombre de CS, el término *federationName* asocia una federación con la propiedad nombre de federación, y el término *performs* une a una federación con una cadena de suministro. De esta manera, la regla garantiza que: cada instancia del concepto *Federation* con su nombre definido, se vinculará a una instancia del concepto *SupplyChain* con el mismo nombre, por medio de la relación *performs*.

$$SupplyChain(?cs) \wedge supplyChainName(?cs,?n) \wedge Federation(?fn) \wedge federationName(?fn,?n) \rightarrow performs(?f,?cs) \quad \text{(Ecuación 3.15)}$$

Cada participante de la cadena de suministro es representado por un federado, por lo que para asegurar el mapeo entre ambos conceptos, se define la regla:

"*Sea el participante P con nombre N, entonces se vincula a P con el concepto federado cuyo nombre de federado es N*"

La Ecuación 3.16 formaliza la regla en el lenguaje SWRL, en donde la variable *?p* es una instancia del concepto *Participant*, que a su vez se infiere como instancia del concepto *Federate*; la variable *?n* es el nombre asociado al participante, el cual a su vez se infiere también como el nombre del federado; los términos *Participant* y *Federate* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *participantName* relaciona a un participante con la propiedad nombre de participante, y el término *federateName* vincula un federado con la propiedad nombre de federado. De esta manera, la regla garantiza que: cualquier instancia del concepto *Participant* con su nombre definido, se inferirá como una instancia del concepto *Federate* con el mismo nombre que el participante.

Participant(?p) ^ participantName(?p,?n)
-> Federate(?p) ^ federateName(?p,?n) (Ecuación 3.16)

Un federado representa a un participante, entonces para asegurar que cada federado se asocie a su participante correspondiente, se define la regla:

"*Sea un participante P con nombre N y sea un federado FE con nombre N, entonces el federado FE representa al participante P*"

La Ecuación 3.17 formaliza la regla en el lenguaje SWRL, en donde las variables *?p* y *?fe*, son individuos del tipo *Participant* y *Federate* respectivamente; la variable *?n* es el nombre asociado, al participante y al federado; los términos *Participant* y *Federate* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *participantName* relaciona a un participante con la propiedad nombre de participante, el término *federateName* vincula un federado con la propiedad nombre de federado, y el término *representA* asocia a un federado con un participante. De esta manera, la regla garantiza que: cada instancia del concepto *Federate* con su nombre definido, se vinculará a una instancia del concepto *Participant* con el mismo nombre, por medio de la relación *representA*.

Participant(?p) ^ participantName(?p,?n) ^ Federate(?fe)
^ federateName(?fe, ?n) -> representA(?fe,?p) (Ecuación 3.17)

Cuando se constituye una federación, se invita a los federados a participar de la misma y, una vez que aceptan, se vuelven miembros de la federación en cuestión. Dado que en el contexto de análisis, una federación representa a una cadena de suministro; entonces los federados que constituyen la federación, deben ser los participantes que forman la cadena de suministro, representada por la federación. Por lo tanto, para asegurar que cada federación tenga como miembros a sus federados correspondientes, se define la regla:

"*Sea la cadena de suministro CS con nombre CN, el participante P con nombre PN, CS este formada por P, la federación FN con nombre CN y el federado FE con nombre PN, entonces FN tiene como miembro a FE*"

La Ecuación 3.18 formaliza la regla en el lenguaje SWRL, en donde las variables *?cs*, *?p*, *?fn*, y *?fe*, son individuos del tipo *SupplyChain*, *Participant*, *Federation*, y *Federate* respectivamente; la variable *?cn* es el nombre asociado a la cadena de suministro CS y a la federación FN; la variable *?pn* es el nombre asociado al participante P y al federado FE; los términos *SupplyChain*, *Participant*, *Federation*, y *Federate* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *supplyChainName* vincula una cadena de suministro con la propiedad nombre de CS, el término *participantName* relaciona a un participante con la propiedad nombre de participante, el término *federationName* asocia una federación con la propiedad nombre de federación; el término *federateName* vincula un federado con la propiedad nombre de federado, el término *formed* relaciona a una cadena de suministro con un participante, y el término *hasMember* asocia a una federación con un federado. De esta manera, la regla garantiza que: una instancia del concepto *Federation* con su nombre definido, se vinculará con las instancias del concepto *Federate* que sean miembros de la federación, por medio de la relación *hasMember*.

$$SupplyChain(?cs) \wedge supplyChainName(?cs,?cn) \wedge Participant(?p) \wedge participantName(?p,?pn) \wedge formed(?cs,?p) \wedge Federation(?fn) \wedge federationName(?fn,?cn) \wedge Federate(?fe) \wedge federateName(?fe,?pn) \rightarrow hasMember(?fn,?fe) \quad \text{(Ecuación 3.18)}$$

Cada relación de la cadena de suministro, se representa por medio de una clase de interacción; por lo que para asegurar el mapeo entre ambos conceptos, se define la regla:

"*Sea la relación R con nombre N, entonces se vincula a R con el concepto clase de interacción cuyo nombre de clase de interacción es N*"

La Ecuación 3.19 formaliza la regla en el lenguaje SWRL, en donde la variable *?r* es una instancia del concepto *Relation*, que a su vez se infiere como instancia del concepto *InteractionClass*; la variable *?n* es el nombre asociado a la relación, el cual a su vez se infiere también como el nombre de la clase de interacción; los términos *Relation* e *InteractionClass* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *relationName* vincula a una relación con la propiedad nombre de relación, y el término *interactionName* asocia a una clase de interacción con la propiedad nombre de interacción. De esta manera, la regla garantiza que: cualquier instancia del concepto *Relation* con su nombre definido, se inferirá como una instancia del concepto *InteractionClass* con el mismo nombre que la relación.

$$Relation(?r) \wedge relationName(?r,?n) \rightarrow InteractionClass(?r) \wedge interactionName(?r,?n) \quad \text{(Ecuación 3.19)}$$

Cada parámetro de una relación de la cadena de suministro, se representa como un parámetro de clase de interacción. Entonces, para asegurar el mapeo entre ambos conceptos, se define la regla:

"*Sea un parámetro de relación PR con nombre N y tipo de dato T, entonces se vincula a PR con el concepto parámetro de interacción, con nombre de parámetro de interacción N y tipo de dato de parámetro de interacción T*"

La Ecuación 3.20 formaliza la regla en el lenguaje SWRL, en donde la variable *?pr* es una instancia del concepto *RelationParameter*, que a su vez se infiere como instancia del concepto *Parameter*; la variable *?n* es el nombre del parámetro de relación, el cual a su vez se infiere también como el nombre del parámetro; la variable *?t* es el tipo de dato asociado al parámetro de relación, y además se infiere también como el tipo de dato del parámetro; los términos *RelationParameter* y *Parameter* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *relationParameterName* asocia a un parámetro de relación con la propiedad nombre de parámetro de relación, el término *relationParameterDatatype* relaciona a un parámetro de relación con la propiedad tipo de dato de parámetro de relación;

el término *parameterName* vincula a un parámetro con la propiedad nombre de parámetro, y el término *parameterDatatype* asocia a un parámetro con la propiedad tipo de dato de parámetro. De esta manera, la regla garantiza que: cada instancia del concepto *RelationParameter* con su nombre y tipo de dato definidos, se inferirá como una instancia del concepto *Parameter* con el mismo nombre y tipo de dato.

$$\begin{aligned} &RelationParameter(?pr) \wedge relationParameterName(?pr,?n) \wedge \\ &relationParameterDatatype(?pr,?t) \rightarrow \\ &Parameter(?pr) \wedge parameterName(?pr,?n) \wedge \\ &parameterDatatype(?pr,?t) \end{aligned} \quad \text{(Ecuación 3.20)}$$

Cada variable atómica de una formula, representa una medición sobre un elemento. Para representar dicha medición, las variables atómicas poseen dos atributos los cuales son: valor y unidad. Además, cada variable atómica se representa como una clase de objeto, con los mismos atributos en una federación HLA. Entonces, para asegurar el mapeo entre ambos conceptos, se define la regla:

"*Sea una variable atómica VA con nombre N, sea el atributo AT1 con nombre unit, sea el atributo AT2 con nombre value, entonces se vincula a VA con el concepto clase de objeto con nombre de clase de objeto N y con los atributos AT1 y AT2"*

La Ecuación 3.21 formaliza la regla en el lenguaje SWRL, en donde las variables *?at1* y *?at2*, son individuos del tipo *Attribute*; la variable *?va* es una instancia del concepto *AtomicVariable*, que a su vez se infiere como instancia del concepto *ObjectClass*; la variable *?n* es el nombre de la variable atómica, el cual a su vez se infiere también como el nombre de la clase de objeto; los términos *AtomicVariable*, *ObjectClass*, y *Attribute* representan los conceptos, de la red de ontologías, con el mismo nombre; el término *variableName* relaciona a una variable con la propiedad nombre de variable, el término *attributeName* asocia a un atributo con la propiedad nombre de atributo; el término *objectName* vincula a una clase de objeto con la propiedad nombre de objeto, y la relación *hasAttribute* relaciona a una clase de objeto con un atributo. De esta manera, la regla garantiza que: cada instancia del concepto

AtomicVarible con su nombre definido, se inferirá como una instancia del concepto *ObjectClass* con el mismo nombre; y que esta instancia se relacionará con sus dos atributos, por medio de la relación *hasAttribute*.

$$\begin{aligned} &AtomicVariable(?av) \wedge variableName(?av,?n) \wedge Attribute(?at1) \\ &\wedge Attribute(?at2) \wedge attributeName(?at1,\text{“unit”}) \\ &\wedge attributeName(?at2,\text{“value”}) \\ &\rightarrow ObjectClass(?av) \wedge objectName(?av,?n) \\ &\wedge hasAttribute(?av,?at1) \wedge hasAttribute(?av,?at2) \end{aligned} \quad \text{(Ecuación 3.21)}$$

Existe un participante con el rol de autoridad (concepto *Authority*) en la cadena de suministro, este rol identifica al participante que inicia la tarea de modelado de la estructura de cadena de suministro, e invita a los demás participantes a unirse a dicha tarea. Dado que toda federación debe ser iniciada por un federado, entonces se asigna la responsabilidad de iniciar la federación, al federado que representa al participante con el rol de autoridad. Por lo tanto, para asegurar que el federado iniciador de la federación sea el participante con rol de autoridad, se define la regla:

“*Sea una cadena de suministro CS con nombre CN, sea un participante P con nombre N, sea el rol de autoridad A, tenga P asignado el rol A, sea CS formada por P, sea la federación FN con nombre CN, sea el federado FE con nombre N y tenga FN como miembro a FE, entonces se vincula a A con el rol de administrador, FE tiene la función A y A inicia a FN*”

La Ecuación 3.22 formaliza la regla en el lenguaje SWRL, en donde las variables *?cs*, *?p*, *?fn* y *?fe*, son individuos del tipo *SupplyChain*, *Participant*, *Federation*, y *Federate* respectivamente; la variable *?a* es una instancia del concepto *Authority*, que a su vez se infiere como instancia del concepto *Administrator*; la variable *?cn* es el nombre asociado a la cadena de suministro y a la federación, la variable *?n* es el nombre asociado al participante y al federado; los términos *SupplyChain*, *Participant*, *Authority*, *Administrator*, *Federation*, y *Federate* representan los conceptos, de la red de ontologías, con los mismos nombres; el

término *supplyChainName* vincula una cadena de suministro con la propiedad nombre de CS, el término *participantName* relaciona a un participante con la propiedad nombre de participante, el término *federationName* asocia una federación con la propiedad nombre de federación; el término *federateName* vincula un federado con la propiedad nombre de federado, el término *formed* relaciona a una cadena de suministro con un participante, el término *hasMember* asocia a una federación con un federado; el término *hasRole* vincula a un participante con un rol, el término *hasFunction* relaciona a un federado con un administrador, y el término *initiates* asocia a un administrador con una federación.

De esta manera, la regla garantiza que: cada instancia del concepto *Federate* que representa a un participante con el rol de autoridad, se vinculará con una instancia del concepto *Administrator* por medio de la relación *hasFunction*; y esta instancia de administrador inicializará la federación de la cual el federado es miembro, mediante la relación *initiates*.

$$\begin{aligned} &SupplyChain(?cs) \wedge supplyChainName(?cs,?cn) \wedge Participant(?p) \\ &\wedge participantName(?p,?n) \wedge Authority(?a) \wedge hasRole(?p,?a) \\ &\wedge formed(?cs,?p) \wedge Federation(?fn) \wedge federationName(?fn,?cn) \\ &\wedge Federate(?fe) \wedge federateName(?fe,?n) \wedge hasMember(?fn,?fe) \\ &\rightarrow Administrator(?a) \wedge hasFunction(?fe,?a) \wedge initiates(?a, ?fn) \end{aligned} \quad \text{(Ecuación 3.22)}$$

Cada parámetro de una relación de la cadena de suministro, se representa como un parámetro de clase de interacción. Entonces los parámetros que están asociados a una clase de interacción, deben ser los parámetros que están asociados a la relación, representada por la clase de interacción. Por lo tanto, para asegurar que cada clase de interacción se asocie con sus parámetros correspondientes, se define la regla:

"*Sea la relación R con nombre N, sea el parámetro de relación PR con nombre NP, R tenga como parámetro a PR, sea la clase de interacción CI con nombre N y el parámetro P con nombre NP, entonces CI tiene como parámetro a P*"

La Ecuación 3.23 formaliza la regla en el lenguaje SWRL, en donde las variables *?r*, *?pr*, *?ci*, y *?p*, son individuos del tipo *Relation*, *RelationParameter*, *InteractionClass*, y *Parameter* respectivamente; la variable *?n* es el nombre asociado a la relación R y a la clase de interacción CI, la variable *?np* es el nombre asociado al parámetro de relación PR y al parámetro P; los términos *Relation*, *RelationParameter*, *InteractionClass*, y *Parameter* representan los conceptos, de la red de ontologías, con los mismos nombres; el término *relationName* vincula a una relación con la propiedad nombre de relación, el término *relationParameterName* asocia a un parámetro de relación con la propiedad nombre de parámetro de relación, el término *interactionName* relaciona a una clase de interacción con la propiedad nombre de interacción; el término *parameterName* vincula a un parámetro con la propiedad nombre de parámetro, el término *parameters* asocia a una relación con un parámetro de relación; y el término *hasParameter* relaciona a una clase de interacción con un parámetro. De esta manera, la regla garantiza que: una instancia del concepto *InteractionClass* con su nombre definido, se vinculará solo con las instancias del concepto *Parameter* que le correspondan, por medio de la relación *hasParameter*.

$$\begin{aligned} &Relation(?r) \wedge relationName(?r,?n) \wedge RelationParameter(?pr) \\ &\wedge relationParameterName(?pr,?np) \wedge parameters\ (?r,?pr) \\ &\wedge InteractionClass(?ci) \wedge interactionName(?ci,?n) \\ &\wedge Parameter(?p) \wedge parameterName(?p,?np) \\ &\rightarrow hasParameter(?ci,?p) \end{aligned} \qquad \text{(Ecuación 3.23)}$$

El objetivo de la cadena de suministro, se representa como el objetivo de la federación. Entonces a efectos de asegurar el mapeo entre ambos conceptos, y que el objetivo mapeado se asocie con la federación correspondiente, se define la regla:

"*Sea la cadena de suministro CS con nombre CN, sea el objetivo de cadena de suministro OC con nombre NO y descripción DO, CS tenga como objetivo a OC y sea la federación FN con nombre CN, entonces se vincula a OC con el concepto objetivo de federación con nombre NO y descripción DO, en donde FN tiene como objetivo a OC*"

La Ecuación 3.24 formaliza la regla en el lenguaje SWRL, en donde las variables *?cs* y *?fn*, son individuos del tipo *SupplyChain* y *Federation*; la variable *?oc* es una instancia del concepto *Goal*, que a su vez se infiere como instancia del concepto *Objective*; la variable *?cn* es el nombre asociado a la cadena de suministro y a la federación, la variable *?no* es el nombre asociado al objetivo de cadena de suministro y al objetivo de federación; la variable *?do* es la descripción asociada al objetivo de cadena de suministro y al objetivo de federación, los términos *SupplyChain*, *Goal*, *Federation*, y *Objective* representan los conceptos, de la red de ontologías, con los mismos nombres; el término *supplyChainName* vincula una cadena de suministro con la propiedad nombre de CS, el término *goalName* asocia a un objetivo de cadena de suministro con la propiedad nombre de objetivo de CS; el término *goalDescription* relaciona a un objetivo de CS con la propiedad descripción de objetivo de CS, el término *federationName* vincula una federación con la propiedad nombre de federación; el término *objectiveName* asocia a un objetivo de federación con la propiedad nombre de objetivo de federación, el término *objectiveDescription* relaciona a un objetivo de federación con la propiedad descripción de objetivo de federación; el término *hasGoal* vincula una cadena de suministro con un objetivo de CS, y el término *hasObjective* asocia a una federación con un objetivo de federación.

De esta manera, la regla garantiza que: una instancia del concepto *Goal* con su nombre y descripción definidos, se inferirá como una instancia del concepto *Objective* con el mismo nombre y descripción. Además, esta instancia se asociará a la instancia del concepto *Federation* correspondiente, por medio de la relación *hasObjective*.

$$\begin{aligned} &SupplyChain(?cs) \wedge supplyChainName(?cs,?cn) \wedge Goal(?oc) \\ &\wedge goalName(?oc,?no) \wedge goalDescription(?oc,?do) \\ &\wedge hasGoal\ (?cs,?oc) \wedge Federation(?fn) \wedge \\ &federationName(?fn,?cn) \\ &\rightarrow Objective(?oc) \wedge objectiveName(?oc,?no) \wedge \\ &objectiveDescription(?oc,?do) \wedge hasObjective(?fn,?oc) \end{aligned} \quad \text{(Ecuación 3.24)}$$

3.4 Implementación de la Red de Ontologías

La red de ontologías SCFHLA se implementa utilizando *Protégé* (Stanford 2016). La misma es una herramienta basada en *Java*, que es extensible, que provee un ambiente *plug and play*; y es flexible para crear prototipos de manera rápida, así como también para el desarrollo de aplicaciones (Knublauch et al. 2005). Además, soporta completamente OWL 2 y la especificación RDF-Schema, ambos estándares de la *World Wide Web Consortium* (W3C). Las ontologías que se implementan sobre *Protégé*, se pueden exportar en diferentes formatos, entre ellos RDF y OWL. Las ontologías que se especifican en OWL 2, proveen clases, propiedades, individuos, y valores de datos, que son intercambiados como documentos RDF. Los modelos RDF tienen la ventaja, por sobre otros esquemas de metadatos, que estructuran la información en grafos, lo que facilita su recuperación.

El lenguaje OWL 2 presenta tres versiones, las cuales son: *Lite*, *DL*, y *Full*, que varían de acuerdo al nivel de expresividad que posee cada uno. La versión *Lite* es la versión con más baja expresividad de las tres, dado que permite definir jerarquías de clasificación, y restricciones simples; pero posee a su favor, una alta eficiencia para razonar, y la garantía de terminación cuando lleva a cabo razonamientos. La versión *DL* posee una expresividad intermedia, dado que posee mayor expresividad que la versión *Lite* y menor expresividad que la versión *Full*. Además, la misma soporta casi por completo los conceptos de la lógica descriptiva, y permite realizar deducciones o razonamientos con garantías de terminación. La versión *Full* es la versión con mayor expresividad de las tres, dado que utiliza por completo la expresividad de OWL, y posee libertad sintáctica sobre RDF; pero como desventaja, no posee garantías de terminación al realizar inferencias (Negri et al. 2016).

La Figura 3.4 presenta el proceso de implementación que se lleva a cabo, para construir la red de ontologías SCFHLA, en donde cada ontología de dominio se implementa con el lenguaje OWL 2 en su versión DL. Se elige esta versión de OWL 2, dado que presenta un adecuado nivel de expresividad, y garantiza la terminación al realizar inferencias lógicas. Además, cada modelo semántico se construye por separado para concentrarse en el diseño de cada dominio específico; y luego se importa cada modelo semántico, en un nuevo

documento, para modelar la red de ontologías SCFHLA. Este nuevo documento, incorpora la definición de las dos meta relaciones presentadas en la Figura 3.3. Además este documento, añade las reglas SWRL para mapear los conceptos de la ontología SCK, a los conceptos de la ontología HLAFed; las cuales fueron presentadas en las Ecuaciones 3.14 a 3.24. Todos los elementos definidos en las ontologías se han especificado en el idioma inglés, a efectos de mantener la compatibilidad con los estándares utilizados, para el modelado de cada uno de los dominios.

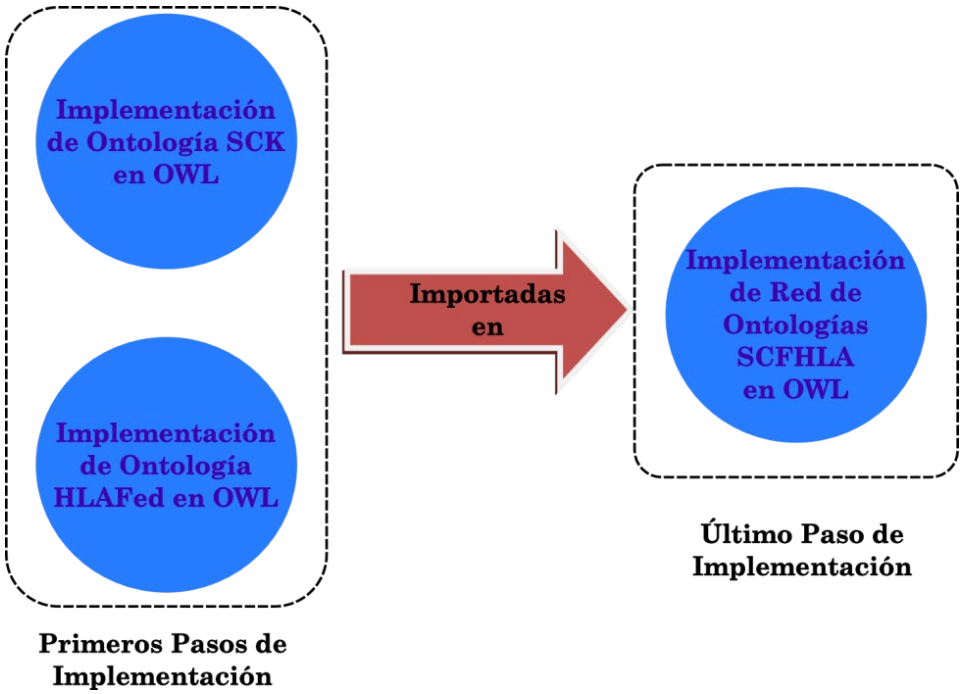


Figura 3.4 – Proceso de implementación para construir la red SCFHLA

La Figura 3.5 visualiza la implementación de la red de ontologías SCFHLA, sobre la vista lógica de *Protégé*. La clase seleccionada, que se presenta en el panel izquierdo de la vista, se corresponde con el concepto Federación (*Federation*), mientras que las propiedades asociadas al concepto se presentan en el panel derecho de la vista. El panel derecho contiene dos secciones, las cuales son: anotaciones (*Annotations*) y descripción (*Description*). En la sección de anotaciones, se utiliza la propiedad denominada comentario (*comment*), para describir de forma coloquial el significado del concepto en cuestión. Esta propiedad, se basa en el tipo de dato cadena de caracteres (tipo de dato *string*). En la sección de descripción, se

incluye el conjunto de axiomas que especifican el concepto. Cada axioma del tipo equivalente a (*Equivalent to*) modela un vínculo desde la clase Federación hacia otra clase o tipo de dato; en otras palabras, representa las relaciones modeladas para relacionar el concepto seleccionado, con el resto de los conceptos y atributos de la ontología. Por ejemplo: la relación *performs* vincula el concepto *Federation* con el concepto *SupplyChain*, en tanto que la relación *federationName*, relaciona el concepto *Federation* con el tipo de dato cadena de caracteres. Los axiomas incluidos en esta sección, mantienen las relaciones para el concepto *Federation*, presentadas en la Figura 3.3.

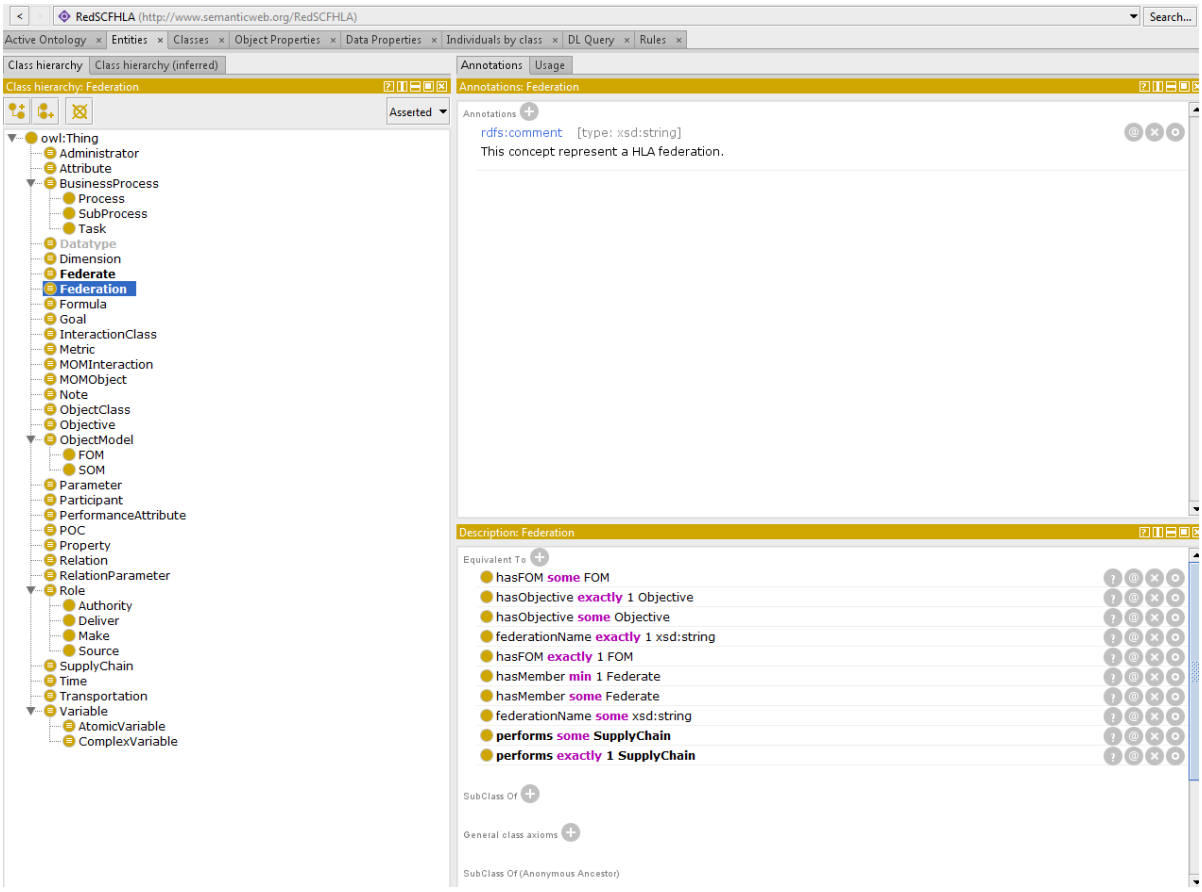


Figura 3.5 – Implementación de la red SCFHLA sobre la vista lógica de *Protégé*

Las reglas SWRL se definen en la red de ontologías, por medio del uso de *Protégé*. La Figura 3.6 presenta la implementación de las reglas SWRL, sobre la vista de reglas (*Rules*). La regla seleccionada en la figura, es la formulada en la Ecuación 3.19, en donde se mapea

una relación de cadena de suministro, a una clase de interacción de una federación HLA. Se puede observar que, las reglas que se encuentran en negrita, son aquellas reglas definidas sobre el documento de especificación de la red de ontologías SCFHLA. El resto de las reglas, son aplicables a otros aspectos de las ontologías SCK y HLAFed; y además, se importan de los documentos de especificación de cada ontología.

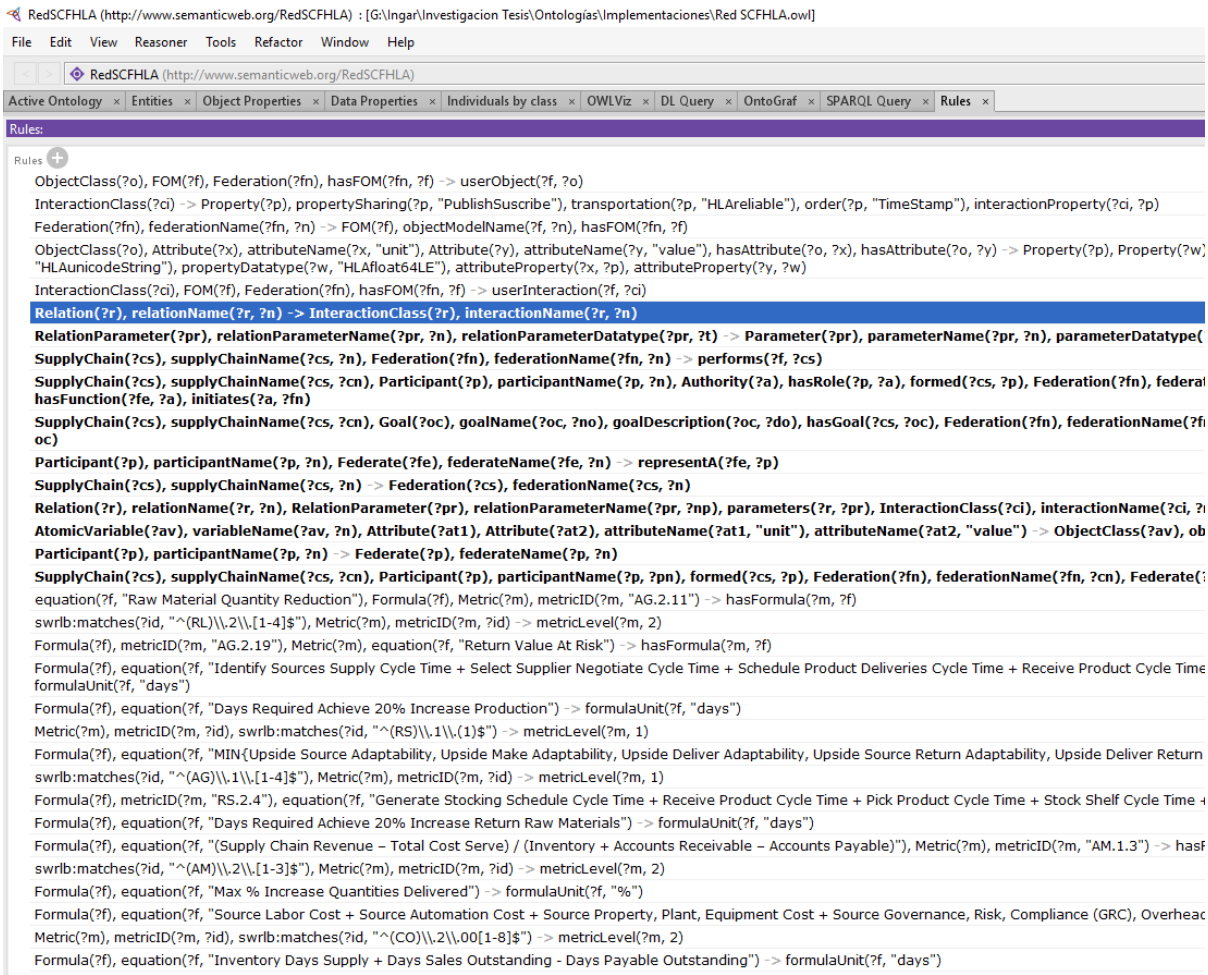


Figura 3.6 – Implementación de las reglas SWRL en *Protégé*

3.5 Evaluación

Al igual que en el desarrollo de software, en la ingeniería ontológica la etapa de evaluación es sumamente importante. En este caso dentro del proceso de diseño y desarrollo

de la red ontologías, se analiza el diseño final implementado (en el caso de esta tesis, en *Protégé*) para detectar potenciales problemas, los cuales surgen por la falta de experiencia de modelado. En esta etapa, se incluyen actividades de verificación y validación, que van a ser realizadas sobre la red de ontologías.

3.5.1 Detección y Resolución de Problemas de Modelado

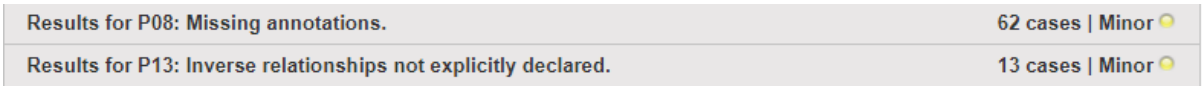
La detección y resolución de problemas de modelado, es parte de la actividad de verificación. El objetivo de esta actividad, es garantizar la correcta implementación de la red de ontologías. En este sentido, garantizar la correcta implementación de la red de ontologías desarrollada, se refiere a evitar una serie de errores de modelado, los cuales son cometidos con frecuencia, por quienes definen modelos durante la actividad de diseño.

Una de las herramientas de verificación, que más se utiliza en el ámbito de la ingeniería ontológica, es *OOPS!* (por sus siglas en inglés –*OntOlogy Pitfall Scanner!*-). Esta herramienta es una aplicación web, que es independiente de cualquier entorno de desarrollo de ontologías. Además, permite detectar los principales problemas de modelado y diseño de ontologías, entre los que se pueden destacar los siguientes: existencia de errores estructurales, funcionales, de usabilidad y perfiles, de consistencia, de completitud y de concisión (Gómez-Pérez 2004; Noy y McGuinness 2001; Rector et al. 2004; Hogan et al. 2010; Archer, Goedertier, y Loutas 2012; Heath y Bizer 2011; Staab y Studer 2009). El objetivo de esta herramienta es brindar soporte a los desarrolladores, por medio de la descripción de los problemas que se detectan, en relación a una ontología particular; y, además, a través de un reporte de problemas de modelado, que se clasifican según el nivel de importancia, en: crítico, importante, y menor. Un problema crítico, *es aquel que debe ser corregido para no afectar la consistencia, el razonamiento, y la aplicación de la ontología*. Un problema importante, *es un inconveniente que, aunque no es crítico para el funcionamiento de la ontología, es deseable que sea solucionado*. Por último, un problema menor, se refiere a *un aspecto que no es vital que sea resuelto, pero en caso de ser solucionado, mejora el modelo de la ontología*.

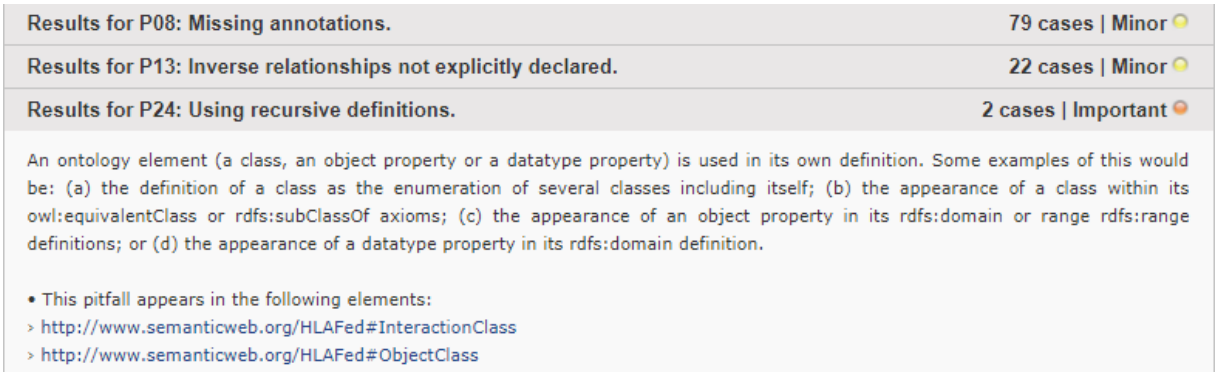
Para cada problema detectado, la herramienta muestra un ejemplo, e indica la parte del modelo que lo causa. Las actividades involucradas con la resolución de los problemas detectados, no forman parte del alcance de dicha aplicación. Entonces, quienes diseñan e implementan la ontología (o red de ontologías en este caso) deben encargarse de realizar las modificaciones necesarias, a efectos de resolver los problemas identificados sobre el modelo. Una vez realizadas las modificaciones, es necesario volver a ejecutar la verificación, y comprobar que todos los problemas han sido resueltos.

Para detectar los problemas de modelado sobre una ontología, la herramienta web posee dos alternativas. Una alternativa, permite al usuario ingresar la *URI* (por sus siglas en inglés -*Uniform Resource Identifier*-) de la ontología. En tanto que, la otra alternativa, permite al usuario ingresar el contenido del archivo OWL/RDF en donde se almacena la ontología, para un análisis directo. Además, la herramienta permite al usuario utilizar una *URL* (por sus siglas en inglés –*Uniform Resource Locator*-), e ingresarla como una URI.

Dado que la red de ontologías SCFHLA se encuentra compuesta por dos ontologías de dominio (en documentos independientes), para llevar a cabo la actividad de verificación de la misma, se toma la decisión de ejecutar la verificación sobre cada uno de los archivos OWL desarrollados. La Figura 3.7, presenta los problemas detectados en las ontologías SCK (a) y HLAFed (b). Como se puede observar los problemas detectados en la verificación, tienen un nivel de importancia menor, a excepción del último problema detectado en la ontología HLAFed (P24), el cual posee un nivel de importancia importante.



(a) Ontología SCK



(b) Ontología HLAFed

Figura 3.7 – Errores de modelado detectados en la red de ontologías SCFHLA

En ambos casos se han dado los mismos problemas, con la salvedad que en el caso de la ontología HLAFed, existe un problema adicional relacionado con el uso de definiciones recursivas. En cuanto al problema de modelado detectado "P08: Falta de anotaciones" (*Missing annotations* en la Figura 3.7), el mismo es un problema de importancia menor y se debe a la falta de etiquetas (o *labels* en *Protégé*), para la descripción del concepto o la definición de sinónimos. En ambas ontologías se añade la propiedad denominada comentario (*comment* en *Protégé*), como anotación, con el fin de describir de forma coloquial el concepto en cuestión, para resolver el inconveniente detectado.

El problema de modelado detectado "P13: Relaciones inversas no declaradas explícitamente" (*Inverse relationships not explicity declared* en la Figura 3.7), el mismo es un error que no es un verdadero problema. Su importancia es menor, y dada la naturaleza de los dominios modelados, no todas las relaciones tienen una relación inversa. En el dominio de cadenas de suministros, como en el de federaciones HLA, la mayoría de las relaciones son unidireccionales y, por lo tanto, es innecesario definir relaciones inversas. En algunos casos se añaden relaciones inversas, con el objetivo de garantizar la navegabilidad bidireccional entre los conceptos de la ontología.

El problema de modelado detectado "P24: Utilizar definiciones recursivas" (*Using recursive definitions* en la Figura 3.7b), si bien se detalla como un problema importante, en realidad no es un verdadero inconveniente. En la Figura 3.7b se observa que para el caso de la ontología HLAFed, el problema de las definiciones recursivas se encuentra en los conceptos *ObjectClass* e *InteractionClass*, dado que ambos tienen una relación con sí mismo, para representar que se pueden definir jerarquías de conceptos (relación *hasSubclass* y *hasSpecialization*, en cada concepto respectivamente). Estas relaciones recursivas representan la posibilidad válida, que define el estándar HLA, de refinar los conceptos de *ObjectClass* e *InteractionClass* tanto como sea necesario, para la conformación de una federación HLA. De acuerdo a lo establecido en el estándar HLA, es posible modelar jerarquías de clases de objeto y de clases de interacción, las cuales definen un concepto de forma más específica comparado con sus conceptos padres (como en la programación orientada a objetos). Por este motivo las relaciones recursivas no se eliminan de cada una de las ontologías, dado que están modelando la posibilidad de definir jerarquías de clases de objeto o interacción, establecidas en el estándar HLA para las federaciones HLA.

Luego de analizar los documentos OWL de cada una de las ontologías, y de corregir los problemas detectados en la Figura 3.7 que requerían una solución; la red de ontologías SCFHLA se verifica haciendo uso de la misma herramienta web (OOPS!), que se utilizó en el caso de las ontologías. Dado que el documento OWL final de la red de ontologías, importa las definiciones incluidas en los otros modelos semánticos, la verificación a realizar, solo analiza la información definida como parte del diseño de la red (es decir las meta relaciones y reglas SWRL definidas en la sección 3.3.3 Integración de la Red SCFHLA). Como resultado de realizar este proceso de verificación, se ha detectado que: existen problemas de modelado, debido a la falta de relaciones inversas. En este caso, la relación *performs* es direccionada, y no posee una relación inversa. Por esta situación, la herramienta detecta de forma positiva la existencia del error de modelado, pero al igual que en los otros casos, esta situación no representa realmente un problema.

3.5.2 Especificación y Ejecución de Consultas SPARQL

La especificación y ejecución de consultas SPARQL es parte de la actividad de validación, para garantizar que la red de ontologías cumpla con su objetivo. En este sentido la red de ontologías desarrollada, debe responder a las preguntas de competencia planteadas.

Como se mencionó en la sección 2.4.1 (Ontologías y Redes de Ontologías), el lenguaje SPARQL es uno de los más poderosos, para extraer información de las ontologías que se implementan con OWL 2. Este lenguaje permite realizar consultas en grafos RDF, y como los documentos OWL 2 se pueden transformar en documentos RDF, entonces SPARQL se puede aplicar sobre las ontologías para realizar consultas.

Con el objetivo de validar la red de ontologías, las preguntas de competencia definidas en la Tabla 3.2, se especifican como consultas SPARQL. Para obtener resultados sobre las consultas a desarrollar, se requiere de un paso previo, el cual consiste en poblar la red de ontologías con individuos. Si no se realiza este paso previo, las consultas SPARQL no pueden devolver una respuesta, dado que la red de ontologías no contiene individuos con la información solicitada. Una vez que se efectuado este paso previo, se procede a activar el razonador mediante la opción: *Reasoner -> Start reasoner* de *Protégé*, para ejecutar las reglas SWRL y, de esta forma, obtener las propiedades inferidas de los individuos. Luego de ejecutar las reglas y realizar las inferencias, se guarda la red de ontologías poblada, y sobre esta red de ontologías se realizan las consultas, para obtener los individuos que responden a las preguntas de competencia. Además, se ejecutan diversas pruebas a efectos de garantizar que las consultas respondan con la información esperada, a cada pregunta de competencia.

En la Figura 3.8 se presenta, a modo de ejemplo, la consulta definida para resolver la pregunta de competencia con ID 2 "*¿Qué atributo de rendimiento es evaluado por la métrica m?*". Como se puede observar, la consulta se compone básicamente de tres cláusulas: *SELECT*, *WHERE*, y *FILTER*. La cláusula *SELECT* define una variable denominada cómo *?atributoRendimiento*, y se establece como condición, que dicha variable cumpla con la restricción expresada en la cláusula *WHERE*. En este contexto, la variable representa el nombre del atributo de rendimiento, que evalúa la métrica con nombre de métrica igual a *m*.

La búsqueda del atributo de rendimiento, queda restringida conforme lo especificado en la cláusula WHERE, donde se indica que:

- El primer paso es buscar todos los individuos con nombre de métrica *m*, cuyo valor que se especifica en la cláusula FILTER, de la línea 7.
- El siguiente paso es buscar todos los individuos, que se vinculan al conjunto de individuos con nombre de métrica *m*, por medio de la relación *measuringA*. Al realizar esta tarea, se obtienen todas las instancias de atributos de rendimiento, asociadas a la métrica con nombre de métrica *m* (línea 4).
- El siguiente paso es verificar que, las instancias de atributos de rendimiento (obtenidas en el paso previo), sean del tipo *PerformanceAttribute* (línea 5).
- El último paso es obtener el nombre del atributo de rendimiento, por medio del data property *attribute* (línea 6).

```
1  SELECT DISTINCT  ?atributoRendimiento
2  WHERE {
3          ?metric SCK:metricName ?var.
4          ?metric SCK:measuringA ?atributo.
5          ?atributo rdf:type SCK:PerformanceAttribute.
6          ?atributo SCK:attribute ?atributoRendimiento.
7          FILTER (?var = m)
8        }
```

Figura 3.8 – Pregunta de competencia ID 2 especificada como consulta SPARQL

Las demás preguntas de competencia, se especifican siguiendo los mismos lineamientos que en el caso de la pregunta de competencia con ID 2, presentada en la Figura 3.8.

En la Figura 3.9 se visualiza, a modo de ejemplo, el resultado de la ejecución de la consulta SPARQL, asociada a la pregunta de competencia con ID 2, y con valor de la variable *m* igual a "*Delivery Performance to Customer Commit Date*". El resultado de la ejecución de esta consulta, el cual se observa en el panel inferior con el nombre de

atributoRendimiento, indica que: el atributo de rendimiento evaluado por la métrica, con nombre de métrica "*Delivery Performance to Customer Commit Date*" (Entrega en Fecha Solicitada por Cliente), es el de *Reliability* (Confiabilidad). Este resultado, coincide con lo especificado en el modelo de referencia SCOR, para dicha métrica. De esta forma, se puede corroborar que la información brindada por la consulta, es correcta.

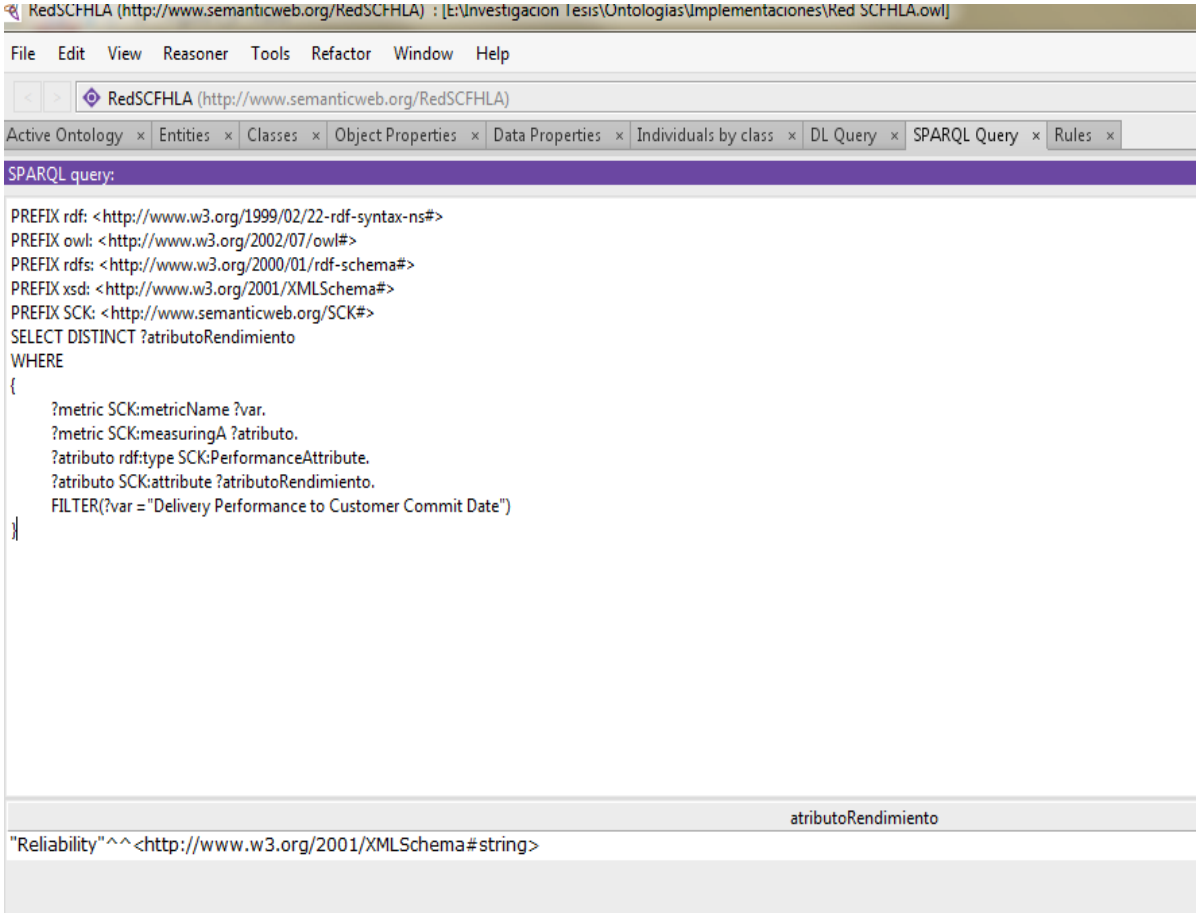


Figura 3.9 – Ejecución de la consulta SPARQL de la pregunta ID 2

3.6 Conclusiones

En este capítulo se presenta la red de ontologías SCFHLA, la cual tiene como objetivo proporcionar un marco conceptual, que permita la especificación del modelo de interoperabilidad, para SD basada en HLA de cadenas de suministro.

La red de ontologías propuesta, permite a los usuarios del dominio de cadenas de suministro concentrarse en la definición de la estructura de CS, sin necesidad de contar con un conocimiento específico sobre los conceptos de HLA. Dichos conceptos de HLA, resultan necesarios para especificar el modelo de interoperabilidad, para SD basada en HLA de cadenas de suministro. Además, la red SCFHLA presenta un vocabulario común para el desarrollo de las tareas de modelado, el cual se encuentra basado en el modelo de referencia SCOR y el estándar HLA.

A través de la definición de un conjunto de reglas SWRL, la red de ontologías provee un mecanismo para verificar, que la información del modelo de interoperabilidad es consistente. En otras palabras, se especifican reglas que guían a los modeladores en la especificación del modelo de CS y, además, permiten el mapeo de los conceptos utilizados en la definición de la estructura de cadena de suministro, en los conceptos necesarios de SD que conforman el modelo de interoperabilidad.

En este sentido la red de ontologías propuesta, define el vocabulario asociado con la conformación de una federación HLA, y el modelado de la estructura de CS. Por medio del vocabulario, la red de ontologías permite modelar y relacionar la información necesaria del modelo de interoperabilidad, para SD basada en HLA de cadenas de suministro. Al utilizar esta información para la construcción del FOM, se garantiza que el modelo de objetos de federación obtenido, cumple con la condición de ser semánticamente interoperable

Capítulo 4 **Diseño de una Plataforma Basada en Ontologías para la Construcción del Modelo de Objetos de Federaciones HLA**

En este capítulo se presentan las problemáticas asociadas a la construcción de un modelo de objetos de federación, a partir de las cuáles se especifican los requerimientos para la plataforma basada en ontologías. Además, se presenta el diseño de la arquitectura de la plataforma y la definición de sus componentes. Entre los componentes de la plataforma se destaca un algoritmo de transformación, el cual cumple la función de utilizar la información del modelo de interoperabilidad de la red de ontologías, para la construcción de un modelo de objetos de federación.

4.1 Problemáticas Asociadas a la Construcción de un Modelo de Objetos de Federación

La red de ontologías SCFHLA presentada en el capítulo anterior, permite modelar la información requerida por el modelo de interoperabilidad, para SD basada en HLA de cadenas de suministro, a partir de la definición de una estructura de CS. Además, por medio del conjunto de reglas definidas, se asegura la consistencia entre la estructura de cadena de suministro, y la información del modelo de interoperabilidad. A partir de esta información, se construye un modelo de objetos de federación, que cumple con la condición de ser semánticamente interoperable. Estas son grandes ventajas, con respecto a las herramientas actuales para la construcción del FOM.

Tanto los analistas de negocios como la comisión directiva de una cadena de suministro, conforman el principal grupo de usuarios de una SD basada en HLA de CS. Para este grupo, algunos de los cuestionamientos que surgen asociados a la red de ontologías, son:

- ¿Cómo utilizar la red de ontologías SCFHLA? La red se encuentra implementada en *Protégé* y es muy probable que ninguno de los usuarios de este grupo, tenga el conocimiento requerido para crear instancias de los conceptos, seleccionar el razonador adecuado, ejecutar las reglas SWRL, analizar los resultados de las inferencias, y guardar la red poblada, entre otras actividades necesarias; para obtener la información requerida por el modelo de interoperabilidad.
- Por medio de la red de ontologías se obtiene la información del modelo de interoperabilidad, pero ¿cómo se construye un documento que contenga esta información?, y además ¿este documento está listo para su uso, en una simulación distribuida basada en HLA? En otras palabras, la información del modelo de interoperabilidad es condición necesaria, pero no suficiente para su posterior uso en la simulación. Para que esta información se encuentre lista para su uso en la simulación distribuida, es necesario construir un documento XML que contenga dicha información, y respete los lineamientos definidos en la especificación del OMT (IEEE 2010b).
- ¿Cómo se puede asegurar que el modelo de objetos de federación, contiene toda la información requerida para la simulación distribuida, de la estructura de cadena de suministro? Los usuarios no cuentan con herramientas o mecanismos para asegurar que efectivamente así sea. Esto se debe a que por el momento, las herramientas disponibles son editores, que solo garantizan la consistencia sintáctica del documento del FOM. En otras palabras, solo verifican que la información contenida en el mismo, se encuentra en el formato correcto, y con las etiquetas definidas en XML. Estas herramientas no detectan la falta de etiquetas necesarias, para la simulación distribuida basada en HLA.

- Dado que las herramientas disponibles para la construcción del FOM, requieren de un alto grado de intervención manual; el proceso de construcción se vuelve propenso a errores, y limitado a los usuarios con conocimientos específicos sobre HLA. Entonces, ¿cuáles son las actividades que requieren intervención de los usuarios y cuáles se pueden automatizar, en el proceso de construcción del modelo de objetos de federación?

4.2 Especificación de Requerimientos de la Plataforma Basada en Ontologías

Las problemáticas descriptas en la sección previa, deben ser consideradas al momento de diseñar y construir el modelo de objetos de federación. Por este motivo, a partir de las mismas, se definen como requerimientos de la plataforma basada en la red de ontologías SCFHLA, las siguientes características:

- *Definir un método de construcción del modelo de objetos de federación, que sea simple de utilizar.* A efectos de cumplir con este requerimiento, se especifica una plataforma basada en la red de ontologías SCFHLA, la cual permita la construcción del FOM en pasos simples e intuitivos, para los usuarios. Para este caso, se toma como usuarios al principal grupo de la red de ontologías (los analistas de negocio y comisión directiva). La plataforma debe encapsular la red de ontologías, mediante una herramienta para la construcción del modelo de objetos de federación. Además, la plataforma debe contar con un vocabulario familiar a los usuarios, y un conjunto de componentes simples de utilizar.
- *Construir el modelo de objetos de federación, con el formato adecuado y alineado a las especificaciones del OMT.* Esta es una de las principales problemáticas, por la cual es necesario diseñar e implementar en la plataforma basada en ontologías, un algoritmo de transformación Este algoritmo, utiliza la información definida en la red de ontologías sobre el modelo de objetos de

federación, y la transforma en un documento que cumple con el formato y especificaciones necesarias, para su utilización en una simulación distribuida basada en HLA, de cadenas de suministro.

- *Definir un mecanismo para verificar que el modelo de objetos de federación es completo*. A efectos de cumplir con esta necesidad, se toma la decisión de construir el modelo de objetos de federación a partir de la información de la estructura de cadena de suministro definida en la red de ontologías. Por medio del algoritmo de transformación, el cual tomará como entradas la información de la red de ontologías y de un documento de configuración, se construirá el modelo de objetos de federación el cual cumple con la condición de ser completo (es decir posee todas las etiquetas requeridas).
- *Disminuir la intervención humana, el tiempo y esfuerzo necesarios en el proceso de construcción del modelo de objetos de federación*. Para cumplir con este requerimiento se utilizará el conjunto de reglas definidas en la red de ontologías para automatizar la tarea de inferencia entre los conceptos de cadenas de suministro y federaciones HLA. Además, se diseñará e implementará un algoritmo de transformación para automatizar la tarea de construcción del modelo de objetos de federación a partir de la red de ontologías y de un documento de configuración. Mediante el uso de estos componentes, se pretende cumplir con la disminución del tiempo y esfuerzo requeridos para la construcción del modelo de objetos de federación. A su vez dado que ambas tareas se automatizarán, esto implica una disminución de la intervención humana requerida y, por lo tanto, una disminución de los posibles errores introducidos por dicha intervención.

En las siguientes secciones, se presenta la arquitectura de la plataforma basada en la red de ontologías SCFHLA. La misma se utiliza, para el diseño y construcción del modelo de objetos de federación. Se especifica además, el diseño de los componentes de la plataforma, con especial énfasis en el algoritmo de transformación.

4.3 Arquitectura de la Plataforma Basada en Ontologías

En la Figura 4.1, se presenta la arquitectura de la plataforma basada en la red de ontologías SCFHLA.

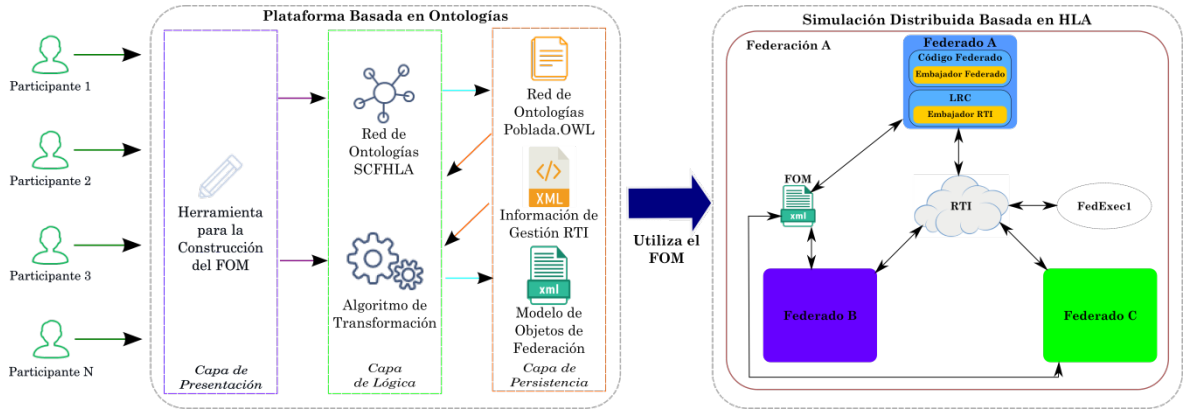


Figura 4.1 – Arquitectura de la plataforma basada en ontologías

Como se observa en la Figura 4.1, la plataforma tiene interacciones con dos componentes: los usuarios y la simulación distribuida basada en HLA. Cada uno de los participantes de la cadena de suministro es un usuario de la plataforma, representado con color verde y la etiqueta participante. Estos usuarios utilizan la plataforma, para definir de forma colaborativa, la estructura de la CS que integran, y a partir de esta información, construir el modelo de objetos de federación. El componente de simulación distribuida basada en HLA, representa una simulación distribuida que contiene una federación HLA, la cual hace uso del modelo de objetos de federación, construido por la plataforma basada en ontologías. La descripción de cada una de las partes que componen una federación HLA, se describió en detalle en la sección 2.2.1 (Estándar IEEE 1516-2010 HLA Evolved).

La plataforma se define en una arquitectura de tres bandas. Esta arquitectura en bandas, es sumamente utilizada cuando se encuentran componentes distribuidos geográficamente, como, por ejemplo, en aplicaciones web. Dado que los participantes de la cadena de suministro se encuentran distribuidos, y que la plataforma debe brindar soporte a los mismos,

en la construcción colaborativa del modelo de objetos de federación; resulta adecuado utilizar la arquitectura en tres bandas, para el diseño de la plataforma.

En la capa de presentación, se define una herramienta para la construcción del modelo de objetos de federación. Esta herramienta permite que los usuarios creen un diagrama, e inviten al resto de los participantes de la cadena de suministro, a unirse al mismo. En este diagrama, los usuarios pueden trabajar de forma colaborativa, para definir la estructura de la cadena de suministro (es decir modelar a los participantes y los vínculos entre ellos, dentro de la cadena en cuestión). La herramienta en la capa de presentación, utiliza la red de ontologías SCFHLA, y a partir de las instancias de los conceptos definidos en la misma, permite modelar en un diagrama la estructura de CS. Además, dicha herramienta permite ejecutar el algoritmo de transformación, para generar el modelo de objetos de federación en el formato adecuado.

La capa de lógica posee dos componentes: la red de ontologías SCFHLA y el algoritmo de transformación. En cuanto a la red de ontologías, este componente se diseña e implementa para que la herramienta (de la capa de presentación), pueda instanciar la red y utilizar sus reglas, a efectos de obtener la información del modelo de interoperabilidad. Además, por medio de la ejecución de las reglas, se verifica si la estructura de cadena de suministro es válida (es decir los conceptos solo se relacionan, con otros conceptos y atributos válidos, según la definición de la ontología SCK).

Una vez validado el modelo de interoperabilidad, la red de ontologías poblada, se almacena en un documento con formato OWL. En cuanto al algoritmo de transformación, este componente se diseña e implementa para que la herramienta (de la capa de presentación), pueda utilizar la información de la red de ontologías, para la construcción del modelo de objetos de federación. Para llevar a cabo la construcción del FOM, se utiliza: el documento OWL de la red de ontologías, y un documento XML (de configuración) que contiene la información de gestión de los servicios del RTI. De esta forma, se puede garantizar que: el modelo de objetos de federación obtenido, se construye con el formato adecuado, y con toda la información necesaria, para su posterior uso en la simulación distribuida basada en HLA.

En la capa de persistencia, se almacenan los documentos generados por la red de ontologías y el algoritmo de transformación; es decir, se almacena la red de ontologías poblada y el modelo de objetos de federación, respectivamente. También en esta capa, se almacenan los documentos que contienen las implementaciones, de la red de ontologías y el algoritmo de transformación. Además de estos documentos, en esta capa se encuentra un documento XML, con la información sobre cómo gestionar los servicios del RTI. En resumen, este componente de la arquitectura contendrá los documentos, que se emplearán en las aplicaciones de la capa de lógica, y en la simulación distribuida de cadenas de suministro basada en HLA.

El empleo de la plataforma, se realiza a partir de la ejecución de un flujo de trabajo predefinido, el cual se presenta en la Figura 4.2.

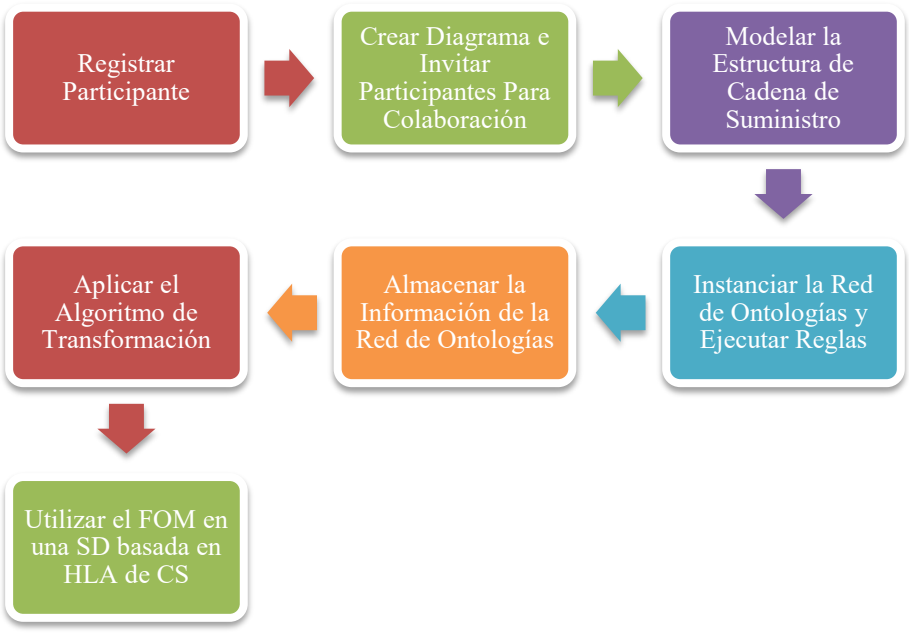


Figura 4.2 – Flujo de trabajo para utilizar la plataforma basada en ontologías

Para poder utilizar la plataforma basada en ontologías, la primera actividad que propone el flujo de trabajo a efectos de construir el modelo de objetos de federación, es registrar a los participantes en la herramienta. En esta actividad, cada participante crea un usuario con toda

su información de contacto, a efectos de que sea posible localizarlo y enviarle invitaciones, para participar de la construcción de un diagrama de la estructura de cadena de suministro.

En la segunda actividad, un participante crea un diagrama de CS e invita al resto de los participantes a colaborar en su construcción. Luego, una vez que los participantes aceptan la invitación para colaborar en la construcción del diagrama, entonces la tercera actividad a realizar, implica que los participantes modelen la estructura de CS en el diagrama, en base a los conceptos definidos en la red de ontologías SCFHLA.

En la cuarta actividad, por medio de la herramienta se instancia la red de ontologías, con la estructura de CS modelada en el diagrama. Además, a través de la herramienta se aplican las reglas SWRL, sobre la estructura de CS instanciada en la red de ontologías. Si la ejecución de las reglas detecta problemas en el modelado, es decir que la estructura de CS no es válida, entonces se deben realizar los cambios necesarios sobre el modelo de CS, a efectos de lograr que la ejecución de las reglas SWRL, de como resultado una estructura de CS válida. Una vez validada la estructura de CS, los conceptos de la misma se encuentran asociados a los conceptos de SD basada en HLA, por medio de la ejecución de las reglas SWRL antes mencionadas. Por lo tanto, la red de ontologías contiene toda la información asociada al modelo de interoperabilidad, para la SD basada en HLA de la estructura de CS.

La quinta actividad implica, almacenar la red de ontologías con la información del modelo de interoperabilidad, en un documento OWL. En la sexta actividad, el algoritmo de transformación se ejecuta desde la herramienta de construcción del FOM. Luego, se obtiene como salida del algoritmo un documento XML, que representa al modelo de objetos de federación, contiene toda la información requerida, y el formato necesario, para su posterior uso en una SD basada en HLA de CS.

Finalmente, la séptima actividad consiste en utilizar el modelo de objetos de federación, el cual fue construido con la herramienta, en la simulación distribuida basada en HLA, de la CS modelada en la tercera actividad.

4.3.1 Definición de los Componentes

En cuanto a la herramienta de construcción del FOM, la misma permite modelar la información relacionada con la estructura y comportamiento de una CS, de forma colaborativa por los analistas de negocio, de cada organización participante de la cadena. Cada uno de los participantes, debe modelar sus procesos de negocio interorganizacionales, y las relaciones con los demás participantes de la cadena. Para modelar esta información, la herramienta se basa en los conceptos definidos en la ontología SCK, la cual forma parte de la red de ontologías SCFHLA. La herramienta utiliza la información de la estructura de CS modelada, para crear instancias en la red de ontologías, a efectos de verificar si la estructura es válida según la definición de la ontología SCK.

La herramienta de construcción del modelo de objetos de federación, debe tener especificadas las siguientes funcionalidades:

- Registrar participante y modificar la información de contacto asociada a un participante.
- Crear, modificar, o eliminar, diagrama de estructura de cadena de suministro.
- Enviar solicitudes de colaboración para un diagrama, y gestionar los participantes que pueden colaborar en un diagrama.
- Poblar la red de ontologías SCFHLA.
- Ejecución de las reglas SWRL definidas en la red de ontologías, para corroborar dos aspectos: la validez de la estructura de CS, y el mapeo de los conceptos de cadenas de suministro, a conceptos de federaciones HLA
- Ejecución del algoritmo de transformación, para construir el modelo de objetos de federación.

A efectos del desarrollo de esta tesis, estas funcionalidades solo han sido detalladas. Por este motivo, con el objetivo de construir el diagrama de la estructura de cadena de suministro, y de llevar a cabo la ejecución de las reglas SWRL, se utiliza la herramienta *Protégé*.

De este modo, las actividades uno y dos del flujo de trabajo propuesto en la Figura 4.2, no se llevarán a cabo, al momento de realizar la prueba de concepto del capítulo 5.

En cuanto a la red de ontologías SCFHLA, si bien la especificación detallada se brindó en el capítulo 3 (Red de Ontologías para Especificar un Modelo de Interoperabilidad en la SD basada en HLA de CS), en esta sección se hace énfasis en el uso de la red, para la vinculación de conceptos entre los dominios de la misma. Por medio de la especificación de un conjunto de reglas definidas en SWRL, la red de ontologías realiza el mapeo de conceptos entre los dominios, y permite obtener información consistente, para la construcción del modelo de objetos de federación. Sin este mapeo fundamental, no es posible obtener la información necesaria, para la construcción del FOM.

La Figura 4.3, esquematiza las actividades necesarias para realizar el proceso de mapeo, entre los conceptos de ambos dominios. Como se puede observar en la Figura 4.3, la primera actividad es la definición de la estructura de CS, por parte de los participantes de cada organización que integran la cadena. En la segunda actividad, se lleva a cabo la tarea de poblar la red de ontologías SCFHLA, luego de lo cual se ejecutan las reglas de mapeo; y de esta forma, se infieren los conceptos relacionados con la federación HLA (los cuales se encuentran definidos en la ontología HLAFed).

Una vez inferidos los conceptos, las instancias contenidas en la red de ontologías poseen toda la información requerida, para ser añadida a un documento que cumpla la función de modelo de objetos de federación, de la federación asociada a la estructura de cadena de suministro modelada anteriormente.

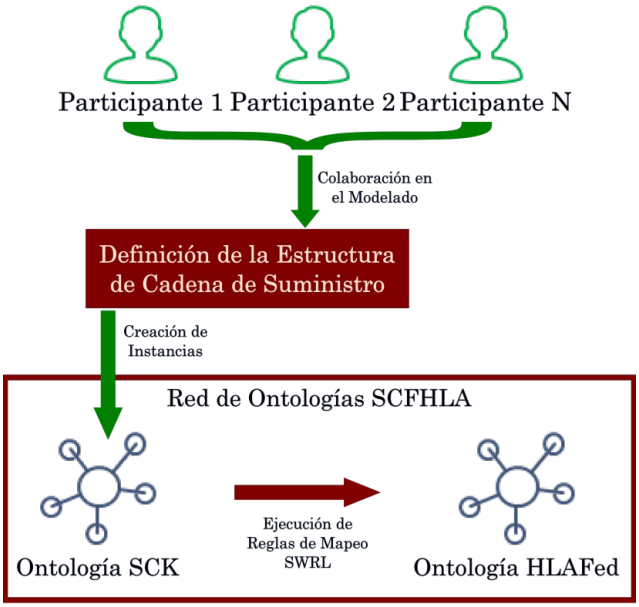


Figura 4.3 – Actividades para el mapeo de conceptos entre los dominios de la red SCFHLA

El componente denominado *algoritmo de transformación*, es abordado en el siguiente apartado, debido a su complejidad inherente.

4.3.2 Algoritmo de Transformación para la Construcción del Modelo de Objetos de Federación

Para que un documento sea utilizado como modelo de objetos de federación, necesariamente se debe encontrar en formato XML. Este documento contiene un conjunto de etiquetas, como la que se presenta en la Ecuación 4.1, en donde: *Inicio-Etiqueta* y *Fin-Etiqueta* deben coincidir en sus nombres, y *cuerpo* define la especificación de la etiqueta junto con sus propiedades.

<Inicio-Etiqueta> cuerpo </Fin-Etiqueta> (Ecuación 4.1)

Todas las etiquetas del documento siguen una estructura de árbol estrictamente jerárquico, en donde existe un único nodo raíz del documento, el cual es la etiqueta denominada *ObjectModel*.

En el nivel inmediato inferior a la raíz, el documento se divide en varias secciones, en donde las dos más relevantes son: la estructura de objetos y la estructura de interacciones. Para la estructura de objetos la etiqueta raíz se denomina *objects*, en tanto que para la estructura de interacciones la etiqueta raíz se denomina *interactions*.

En el siguiente nivel inferior, como hijo de la etiqueta *objects*, se encuentra la etiqueta *HLAobjectRoot*; en tanto que como hijo de la etiqueta *interactions*, se encuentra la etiqueta *HLAinteractionRoot*. En el nivel inferior a estas últimas etiquetas mencionadas, existen partes comunes encargadas de la gestión y acceso, a los servicios brindados por el RTI. Estas partes se encuentran agrupadas bajo la etiqueta *HLAManager*, y no dependen de la estructura de la federación asociada a la cadena de suministro, sino que dependen de la implementación del RTI que se utilice.

Las partes propias de cada federación y que, por lo tanto, dependen de la configuración de la federación modelada, se pueden identificar como agrupadas dentro de las etiquetas: *UserBaseClass* para el caso de los objetos, y *UserInteractionBase* para el caso de las interacciones.

En la Figura 4.4 se presenta la estructura descripta anteriormente, para el documento que cumple la función de modelo de objetos de federación. En el nivel inferior de las etiquetas de gestión de servicios del RTI, como de las etiquetas propias de la federación, se definen dos estructuras:

- La primera es la estructura de objetos, los cuales poseen atributos, y estos últimos poseen propiedades.
- La segunda es la estructura de interacciones, las que poseen parámetros y propiedades.

En la Figura 4.5 se exhiben las dos estructuras descriptas, las cuales se encuentran en los niveles inferiores a las etiquetas *HLAManager*, *UserBaseClass* y *UserInteractionBase*.

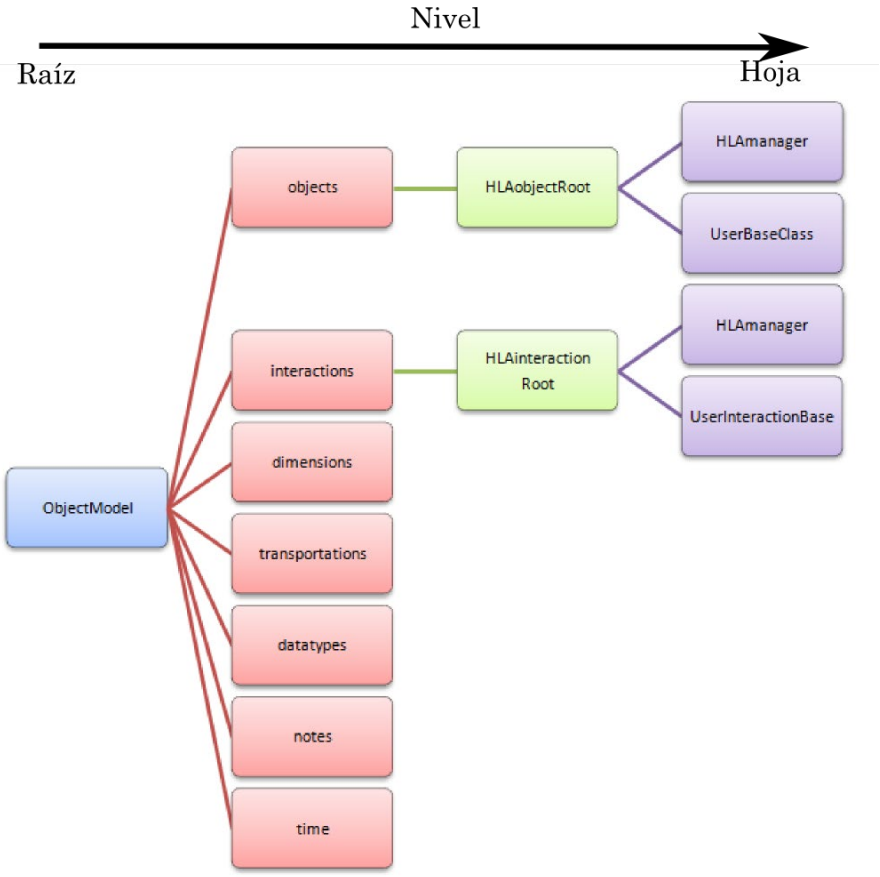


Figura 4.4 – Estructura del documento del modelo de objetos de federación

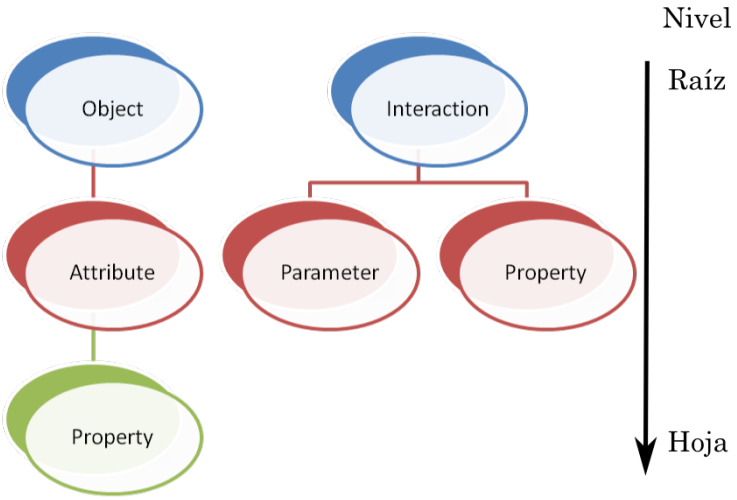


Figura 4.5 – Estructura de objetos e interacciones del modelo de objetos de federación

La estructura de un documento OWL, es similar a la estructura descripta para un documento XML, que cumple la función de modelo de objetos de federación. Entre sus diferencias, se puede destacar que la información no se dispone de una manera evidente, y además, que no se sigue una estructura estrictamente de árbol. Por estos motivos, no es tan sencillo incorporar directamente, la información de la red de ontologías almacenada en un documento OWL, al documento que cumple la función de FOM. Es por ello que, la información de la red de ontologías es procesada por el algoritmo de transformación, para convertirla al formato adecuado del modelo de objetos de federación.

El objetivo del algoritmo de transformación, es utilizar la información de la red de ontologías SCFHLA, para construir el documento del modelo de objetos de federación. El algoritmo utiliza como entradas el documento OWL de la red de ontologías, y un documento XML de configuración, que contiene la información de gestión y acceso a los servicios del RTI. El documento de configuración posee la misma estructura que la presentada en la Figura 4.4, sin los objetos (bajo la etiqueta *UserBaseClass*) y las interacciones (bajo la etiqueta *UserInteractionBase*), propias de cada federación. De esta forma, el algoritmo de transformación garantiza que el documento del modelo de objetos de federación, es construido con el formato adecuado, y con toda la información necesaria, para su posterior uso en la simulación distribuida basada en HLA, de la estructura de cadena de suministro modelada con la red de ontologías. En otras palabras, el algoritmo garantiza que el documento del FOM, cumple con la condición de ser completo.

Para cumplir con su objetivo, el algoritmo de transformación se plantea en dos etapas. En la primera etapa, se obtiene la información asociada a la gestión y acceso a los servicios del RTI. Dado que esta información es estática, si no se modifica la implementación del RTI utilizado, entonces es posible obtener dicha información de un archivo de configuración, con los parámetros utilizados por defecto (o que se utilizan en la mayoría de los casos), como sugieren (Topçu y Oğuztüzün 2017; 2013; öZdikis, Durak, y OguztüZüN 2010). Para esta tesis, se utiliza la implementación del RTI denominada *PoRTIco* (Pokorny, Fraser, y Burns 2016) y, por lo tanto, el documento de configuración es específico para esta implementación.

En la segunda etapa, se obtiene la información de la federación asociada a la estructura de cadena de suministro, modelada por los participantes. Una vez realizadas ambas etapas, el algoritmo construye el documento XML con la información obtenida, a partir de las dos etapas. En la Figura 4.6 se exhibe el proceso para la construcción del FOM, a partir del uso del algoritmo de transformación.

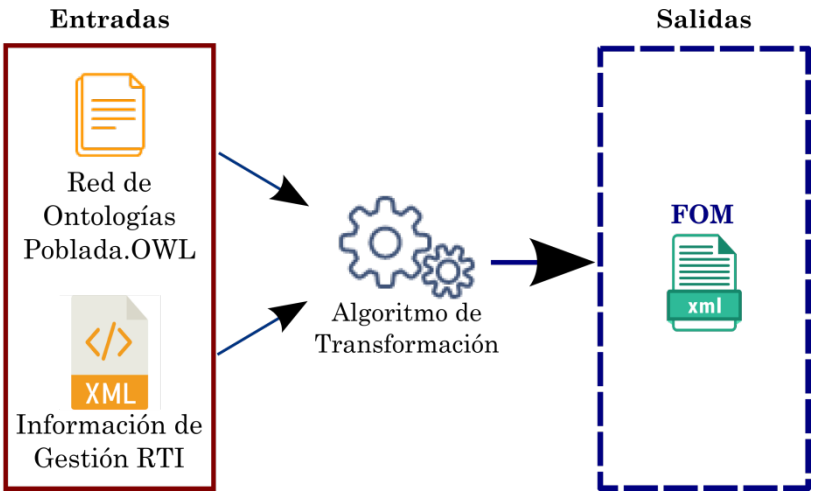


Figura 4.6 – Construcción del FOM mediante el algoritmo de transformación

4.3.2.1 Etapa 1: Información de Gestión del RTI

Esta etapa implica la creación de un documento XML, que contenga la misma información que el archivo de configuración a utilizar. Por lo tanto, se crea un nuevo documento que contiene, una copia de la información almacenada en el archivo de configuración de, en nuestro caso, el RTI *PoRTIco*.

4.3.2.2 Etapa 2: Información de la Red de Ontologías

El algoritmo toma como entrada un documento OWL almacenado en formato RDF/XML, que es el formato por defecto de la herramienta *Protégé*. Las instancias creadas y guardadas en el documento OWL, se denominan individuos de la red de ontologías. En esta etapa se realiza una búsqueda de los individuos instanciados, a efectos de añadir la

información propia de la federación HLA, al documento creado en la etapa anterior bajo las etiquetas correspondientes. Los individuos de la red de ontologías que son del tipo objeto (concepto *ObjectClass*), se insertan en el documento XML bajo la etiqueta *UserBaseClass*. En cambio, los individuos de la red que son del tipo interacción (concepto *InteractionClass*), se insertan en el documento XML bajo la etiqueta *UserInteractionBase*. A efectos de recorrer el documento OWL y obtener los individuos instanciados, se utiliza la estrategia de búsqueda primero en profundidad, dada la estructura jerárquica que poseen los objetos e interacciones del documento (la cual fue presentada en la Figura 4.5).

Para realizar la búsqueda de individuos dentro del documento OWL, el algoritmo de transformación realiza las siguientes actividades:

Actividad 1: *Encontrar a un Individuo*. Como se mencionó anteriormente, cada una de las instancias creadas en la red, es un individuo que se encuentra bajo la etiqueta identificadora *NamedIndividual*.

Actividad 2*: Determinar el tipo del individuo obtenido en la actividad 1*. Cada individuo posee una etiqueta denominada *type*, la cual define cual es el tipo del individuo, de acuerdo a la jerarquía presentada en la Figura 4.5. Es decir que, la etiqueta *type* puede contener, uno de los siguientes valores: *ObjectClass* (para el tipo objeto), *InteractionClass* (para el tipo interacción), *Attribute* (para el tipo atributo), *Parameter* (para el tipo parámetro) o *Property* (para el tipo propiedad). Dentro de la información que contiene un individuo, se encuentran los valores de cada uno de los atributos, y las relaciones con los conceptos definidos para la federación HLA modelada. En la Figura 4.7 se observa, a modo de ejemplo, cómo se encuentra almacenado un individuo, en el documento OWL con formato RDF/XML. Este individuo es del tipo *InteractionClass*, su nombre de individuo es *Deliver Supplies*, su nombre de interacción coincide con el nombre de individuo, sus parámetros son *Date Of Delivery*, *Item*, y *Quantity*; y por último, las propiedades de la interacción se encuentran almacenadas en otro individuo, denominado *InteractionsProperties*.

En las siguientes actividades, se busca un individuo del tipo *ObjectClass* o *InteractionClass*, y luego se buscan todas sus características asociadas. Si el individuo no es

de ninguno de estos tipos, no es necesaria su transformación, y se procede a buscar el siguiente individuo. El algoritmo sigue este criterio, para respetar la estrategia de búsqueda seleccionada, y la estructura jerárquica del documento OWL.

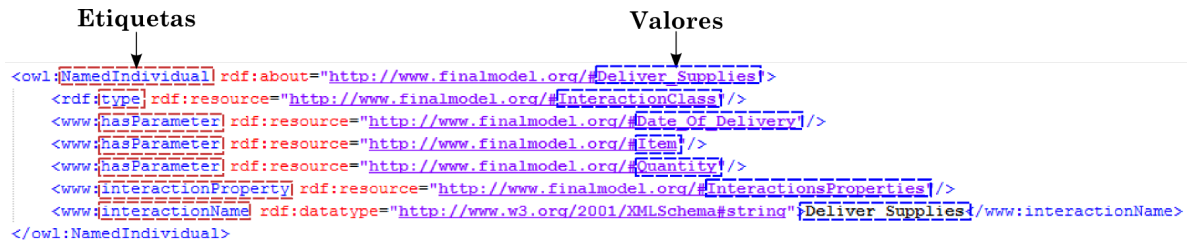


Figura 4.7 – Parte de un documento OWL con formato RDF/XML

Actividad 3: *Si un individuo es del tipo ObjectClass, entonces almacenar sus características*. En este caso, las características asociadas a un objeto se encuentran bajo la etiqueta *object[nombreCaracteristica]*, en donde *[nombreCaracteristica]* puede tomar uno, de los siguientes valores posibles: *Name*, *Sharing*, o *Semantics*. Luego, se obtienen todas las características del objeto y se pasa a la siguiente actividad.

Actividad 3A: *Una vez obtenidas todas las características de un individuo del tipo ObjectClass, entonces buscar sus atributos*. En este caso, los atributos asociados a un objeto se encuentran bajo la etiqueta *hasAttribute*, la cual representa la relación definida en la red de ontologías con el mismo nombre. De cada uno de los atributos se obtiene el nombre, luego se busca un individuo con la etiqueta *NamedIndividual*, cuyo valor de etiqueta sea el nombre del atributo obtenido; y, finalmente, se verifica que la etiqueta *type* contenga el valor *Attribute*.

Actividad 3A.1: *A partir de un atributo de un objeto, entonces buscar sus propiedades*. Una vez encontrado el individuo del tipo *Attribute*, se obtiene su nombre de la etiqueta *attributeName*, y se almacena. Las propiedades de un atributo se encuentran bajo la etiqueta *attributeProperty*, de la cual para cada propiedad se obtiene el nombre, luego se busca un individuo con la etiqueta *NamedIndividual*, cuyo valor de etiqueta sea el nombre de la propiedad obtenida; y, finalmente, se verifica que la etiqueta *type* contenga el valor *Property*. Cuando no existan más propiedades asociadas al atributo, entonces volver a la actividad 3A.

Actividad 3A.1.1*: A partir de la propiedad de un atributo, entonces almacenar sus características*. Una vez encontrado el individuo del tipo *Property* en la actividad 3A, se obtiene su nombre y se almacena. Las características de la propiedad, se encuentran bajo la etiqueta *property[nombreCaracteristica]* o bajo la etiqueta *Nombre*, donde *[nombreCaracteristica]* puede tomar uno, de los siguientes valores posibles: *Sharing*, *Semantics*, o *Datatype*. En tanto que la etiqueta *Nombre* puede tomar uno, de los siguientes valores posibles: *order*, *transportation*, *dimensions*, *ownership*, *updateCondition*, o *updateType*. Luego, se almacenan todas las características de la propiedad.

Actividad 4*: Almacenar el objeto con sus atributos y propiedades*. Esta actividad implica guardar, toda la información recopilada sobre un individuo del tipo *ObjectClass*, junto con sus atributos (individuos del tipo *Attribute*, que se encuentran asociados al objeto) y las propiedades de sus atributos (individuos del tipo *Property*, asociados a los atributos).

Actividad 5*: Si un individuo es del tipo InteractionClass, entonces almacenar su nombre*. En este caso el nombre de una interacción, se encuentra bajo la etiqueta *interactionName*. Se obtiene el nombre de la interacción, y se pasa a la siguiente actividad.

Actividad 5A*: A partir de una interacción, entonces buscar sus propiedades*. En este caso las propiedades asociadas a una interacción, se encuentran bajo la etiqueta *interactionProperty*, que representa la relación definida en la red de ontologías con el mismo nombre. De cada una de las propiedades se obtiene el nombre, luego se busca un individuo con la etiqueta *NamedIndividual*, cuyo valor de etiqueta sea el nombre de la propiedad obtenida; y, finalmente, se verifica que la etiqueta *type* contenga el valor *Property*. Repetir esta actividad, hasta que no haya más propiedades asociadas a la interacción.

Actividad 5A.1*: A partir de la propiedad de una interacción, entonces almacenar sus características.* Una vez encontrado el individuo del tipo *Property*, se obtiene su nombre y se almacena. Las características de la propiedad, se encuentran bajo la etiqueta *property[nombreCaracteristica]* o bajo la etiqueta *Nombre*, donde *[nombreCaracteristica]* puede tomar uno, de los siguientes valores posibles: *Sharing* o *Semantics*. Mientras que la

etiqueta *Nombre* puede tomar uno, de los siguientes valores posibles: *order*, *transportation*, o *dimensions*. Luego, se almacenan todas las características de la propiedad.

Actividad 5B*:* *A partir de una interacción, entonces buscar sus parámetros*. En este caso los parámetros asociados a una interacción, se encuentran bajo la etiqueta *hasParameter*, que representa la relación definida en la red de ontologías con el mismo nombre. De cada uno de los parámetros se obtiene el nombre, luego se busca un individuo con la etiqueta *NamedIndividual*, cuyo valor de etiqueta sea el nombre del parámetro obtenido; y, finalmente, se verifica que la etiqueta *type* contenga el valor *Parameter*. Repetir esta actividad, hasta que no haya más parámetros asociados a la interacción.

Actividad 5B.1*:* *A partir de un parámetro de una interacción, entonces almacenar sus características*. Una vez encontrado el individuo del tipo *Parameter*, se obtiene su nombre de la etiqueta *parameterName*, y se almacena. Las características del parámetro, se encuentran bajo la etiqueta *parameter[nombreCaracteristica]*, donde *[nombreCaracteristica]* puede tomar uno, de los siguientes valores posibles: *Semantics* o *Datatype*.

Actividad 6*:* *Almacenar la interacción con sus propiedades y parámetros*. Esta actividad implica guardar, toda la información recopilada sobre un individuo del tipo *InteractionClass*, junto con sus propiedades (individuos del tipo *Property*, asociados a la interacción) y sus parámetros (individuos del tipo *Parameter*, asociados a la interacción).

Actividad 7*:* *Repetir hasta que no haya más individuos del tipo ObjectClass o InteractionClass*. Esta actividad implica realizar todas las actividades anteriores, en búsqueda de la información almacenada en los individuos de la red de ontologías. Una vez realizada esta actividad, se puede dar por finalizada la ejecución del algoritmo de transformación, y por concluida la tarea de construcción del documento del modelo de objetos de federación.

La Figura 4.8 presenta un diagrama de actividad UML (OMG 2013), a efectos de resumir las actividades requeridas, para la búsqueda de individuos en el documento OWL, de la red de ontologías SCFHLA.

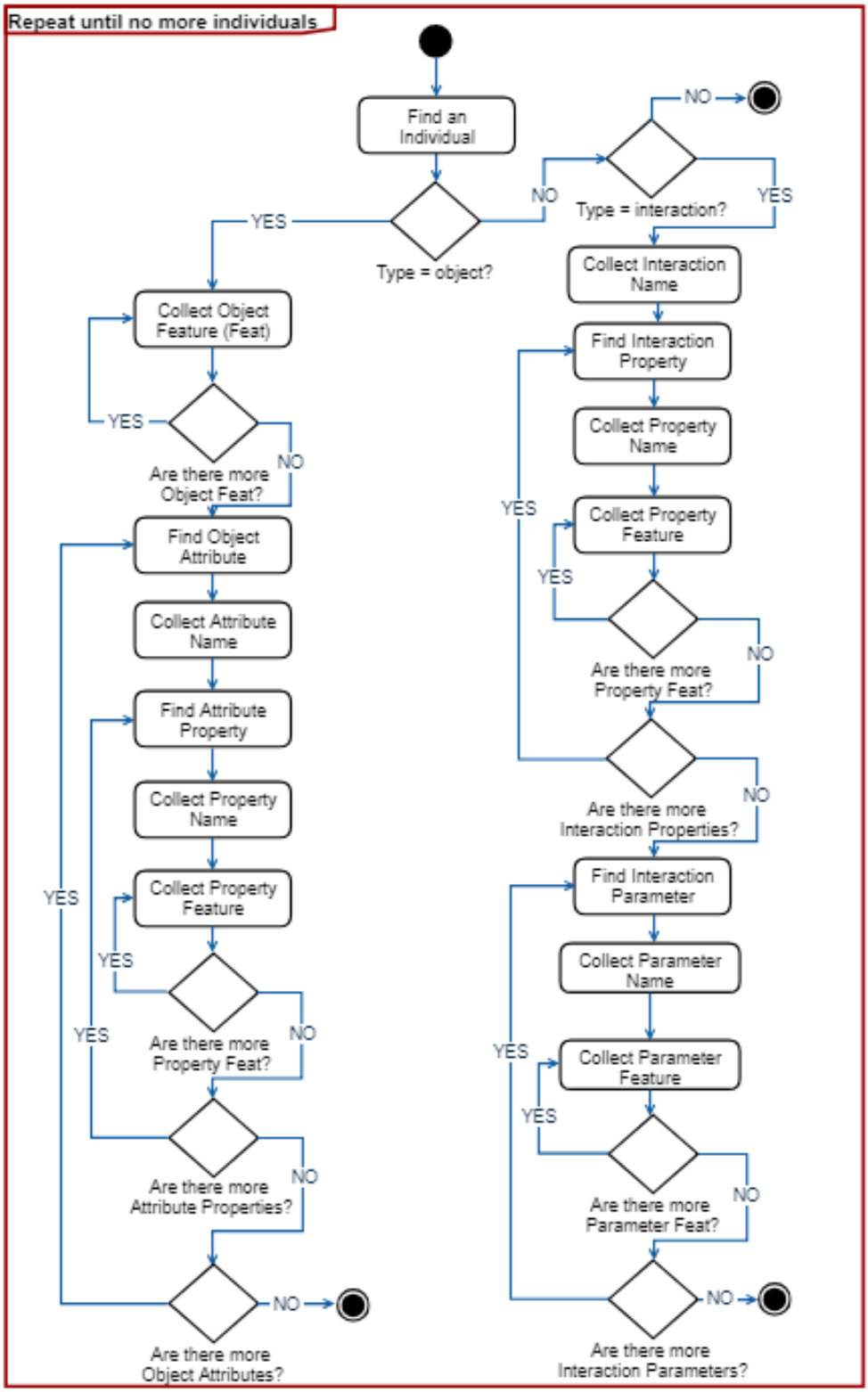


Figura 4.8 – Actividades requeridas para la búsqueda de individuos en documento OWL

El algoritmo de transformación descripto, se implementa por medio de la librería de Java JDOM (por sus siglas en inglés –*Java Document Object Model*-) (“JDOM” 2017). Para que el algoritmo pueda comenzar su ejecución, es necesario inicializar dos parámetros: la ubicación del documento OWL de la red de ontologías poblada, y la ubicación del documento XML con la información de configuración. Una vez finalizado el algoritmo, se obtiene un documento XML el cual contiene toda la información, para ser utilizado como modelo de objetos de federación de la federación HLA a simular.

4.4 Conclusiones

En este capítulo se presentan las problemáticas asociadas, a la construcción de un modelo de objetos de federación. A partir de estas problemáticas, se especifican los requerimientos que debe cumplir, la plataforma basada en la red de ontologías SCFHLA desarrollada en el capítulo 3. Además se define la arquitectura para la plataforma, en donde se presentan cada uno de los componentes, y se describe cual es el flujo de trabajo, para hacer uso de la misma. La arquitectura definida, pretende soportar las necesidades de los participantes que integran una cadena de suministro, a efectos de definir de forma colaborativa la estructura de cadena de suministro. A partir de dicha estructura, se construye el modelo de objetos de federación, necesario para la SD basada en HLA.

La plataforma basada en la red de ontologías, posee dos componentes centrales: uno de ellos es la red de ontologías SCFHLA, mientras que el otro es el algoritmo de transformación. Para utilizar la red de ontologías, se plantea un flujo de actividades, las cuales son: registrar un participante, crear un diagrama de CS e invitar al resto de los participantes a colaborar, modelar la estructura de cadena de suministro, instanciar la red de ontologías, ejecutar reglas SWRL, almacenar la red de ontologías poblada en un documento OWL, ejecutar el algoritmo de transformación; y, por último, hacer uso del FOM construido, en una SD basada en HLA de la cadena de suministro modelada.

El algoritmo permite transformar la información del modelo de interoperabilidad de la red de ontologías, en un documento XML que se adecua a las definiciones, de la plantilla de modelo de objetos de HLA. Para realizar esta transformación, el algoritmo utiliza el documento OWL de la red de ontologías y, además, un documento de configuración XML, con la información de gestión de los servicios del RTI.

El uso del algoritmo de transformación, garantiza que el documento del modelo de objetos de federación sea completo. En otras palabras, se cumple que el documento contiene todas las etiquetas requeridas, para ser utilizado en una simulación distribuida basada en HLA, de cadenas de suministro.

Además, a través del uso de la red de ontologías y el algoritmo de transformación, se ha reducido el esfuerzo, tiempo, y acciones manuales requeridas, para la construcción del modelo de objetos de federación. Con el desarrollo de estas herramientas, se pretende realizar un aporte, desde la ingeniería de software, para disminuir los conocimientos necesarios sobre HLA, a la hora de construir un modelo de objetos de federación.

Capítulo 5 **Prueba de Concepto**

A efectos de validar el enfoque propuesto, en este capítulo se presenta un caso de estudio, donde se describe cómo utilizar los componentes de la red de ontologías SCFHLA y el algoritmo de transformación. El objetivo de este capítulo, es presentar las bondades asociadas con el uso de los componentes propuestos, para alcanzar la interoperabilidad semántica en una SD basada en HLA, de cadena de suministro. Para lograr este objetivo se presenta un caso de estudio, como prueba de concepto, sobre la factibilidad de utilizar el enfoque propuesto, para: a partir de un modelo de interoperabilidad conceptual, que se define en la red de ontologías SCFHLA, obtener el modelo de objetos de federación en el formato adecuado y verificando que es completo, por medio del uso de un algoritmo de transformación. En este capítulo se presentan las actividades requeridas, para obtener el modelo de objetos de federación, a partir de la descripción del escenario de cadena de suministro, propuesto como caso de estudio.

5.1 Contexto

La plataforma basada en ontologías incluye en su capa de lógica, dos componentes los cuales son: la red de ontologías SCFHLA y el algoritmo de transformación. Este algoritmo se diseña para la construcción del modelo de objetos de federación, de una SD basada en HLA; además, el algoritmo contempla que para la ejecución de dicha simulación distribuida, se utilice la infraestructura de tiempo de ejecución denominada *PoRTIco* (Pokorny, Fraser, y Burns 2016). En base a los elementos definidos en el capítulo 3 (Red de Ontologías para Especificar un Modelo de Interoperabilidad en la Simulación Distribuida basada en HLA de Cadenas de Suministro) y en el capítulo 4 (Diseño de una Plataforma Basada en Ontologías

para la Construcción del Modelo de Objetos de Federaciones HLA), se lleva a cabo una prueba de conceptos, que ilustra la forma en la cual se obtiene el modelo de objetos de federación, a partir de la descripción de un escenario de cadena de suministro.

5.2 Descripción del Escenario

Sea una cadena de suministro denominada *Silla CS*, dedicada a la producción y venta de sillas de caño, con dos posibles modelos de producto: uno con asiento blanco y otro con asiento negro. Esta cadena se encuentra conformada por siete participantes, de los cuales tres son proveedores de insumos, uno es la fábrica donde se producen las sillas, y tres son clientes mayoristas vendedores de muebles. Todos los participantes de la cadena de suministro se encuentran radicados en Argentina.

En cuanto a los participantes que son proveedores, cada uno de ellos entrega uno de los insumos necesarios, para la fabricación de las sillas. De este modo, los tres proveedores se corresponden con: el proveedor de asientos que se denomina *Asientos Contemporáneos*, el de caños que se denomina *Nizametal*, y el de pintura que se denomina *Ferretería Industrial Centro*. El proveedor de asientos se encuentra radicado en la ciudad de Paraná, provincia de Entre Ríos, mientras que los otros dos proveedores se encuentran establecidos en la ciudad de Santa Fe, provincia de Santa Fe.

El participante que actúa como la fábrica de la cadena se denomina *Sillaforte*, el cual es el encargado de la fabricación de sillas, y su posterior venta a las mueblerías mayoristas. Esta organización se encuentra ubicada en la ciudad de Santa Fe, provincia de Santa Fe.

Por el lado de los participantes que se desarrollan como clientes en la cadena, cada uno se encuentra distribuido geográficamente a lo largo de Argentina, y compra los dos tipos de productos ofrecidos por *Sillaforte*. De este modo, el primer cliente se denomina *Muebles del Plata*, y se encuentra ubicado en la ciudad autónoma de buenos aires (CABA); el segundo cliente se denomina *Mueblería Cuyo*, y se radica en la ciudad de Mendoza, provincia de Mendoza; por último, el tercer cliente se denomina *La Norteña*, y se ubica en la ciudad de Tucumán, provincia de Tucumán.

Todos los participantes de la cadena de suministro, mantienen en sus organizaciones un cierto nivel de inventario, tanto para los materiales como para los productos finales. Además, cada participante produce (como en el caso de la fábrica y los proveedores) o se abastece (como en el caso de los clientes mayoristas) de productos, solo ante la recepción de órdenes de venta; y, de este modo, cada participante posee una política de inventario, tendiente a minimizar la cantidad de unidades en inventario.

Cada uno de los participantes, planifica y gestiona sus operaciones de acuerdo con el proceso de negocio que se presenta en la Figura 5.1, el cual se ejecuta para cada uno de los productos que comercializa la organización. Cabe destacar que, el proceso de negocio de la Figura 5.1, se modela mediante la notación de modelado de procesos de negocio BPMN (por sus siglas en inglés -*Business Process Model And Notation*-) (OMG 2014).

Como se puede observar de la Figura 5.1, durante el transcurso de la semana el sector de ventas carga los pedidos de producto, los cuales serán satisfechos en la próxima semana. A su vez, en paralelo con esta tarea, el sector de administración estima el nivel de inventario del producto para el final de la semana, de acuerdo con el plan maestro de producción. En el caso de que la cantidad en existencia del producto sea inferior a cero, entonces se emite una orden de abastecimiento, de los productos o materiales faltantes. Cuando se ha recibido el suministro solicitado, o en caso de que la cantidad en existencia del producto sea suficiente, entonces se procede a realizar la entrega de los productos solicitados por los clientes, durante la semana anterior. Al final de cada semana, el sector de administración genera un plan maestro de producción, para la siguiente semana.

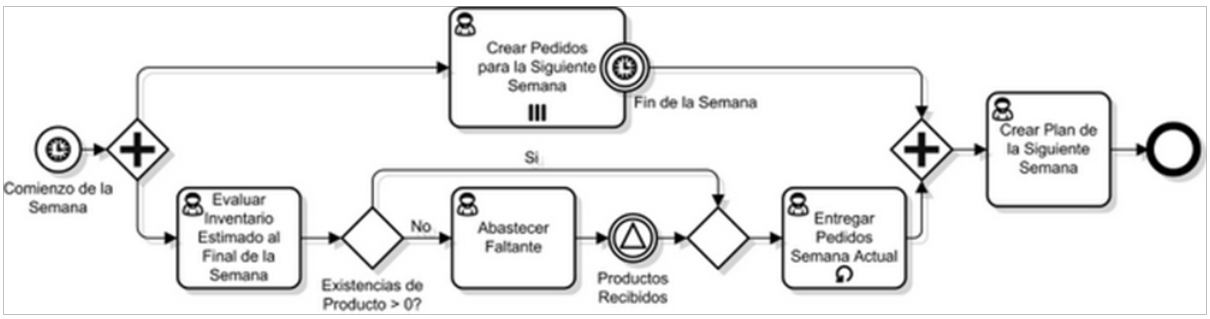


Figura 5.1 – Planificación de las operaciones de cada participante de silla CS

En cuanto a las relaciones entre los participantes de la CS, se establece que los proveedores reciben órdenes de suministro de la fábrica, y entregan el producto o material a la fábrica. En tanto que, la fábrica emite órdenes de suministro a sus proveedores, recibe los materiales o productos de los proveedores, recibe órdenes de compra de sillas por parte de los clientes mayoristas; y, finalmente, envía el producto final a los mayoristas. Por otro lado, los clientes mayoristas emiten órdenes de compra hacia la fábrica, para solicitar la mercadería necesaria; y, a su vez, reciben los pedidos de la mercadería solicitada. En la Figura 5.2, se esquematiza la estructura de la cadena de suministro denominada *Silla CS*, donde se presentan sus participantes y las relaciones entre ellos.

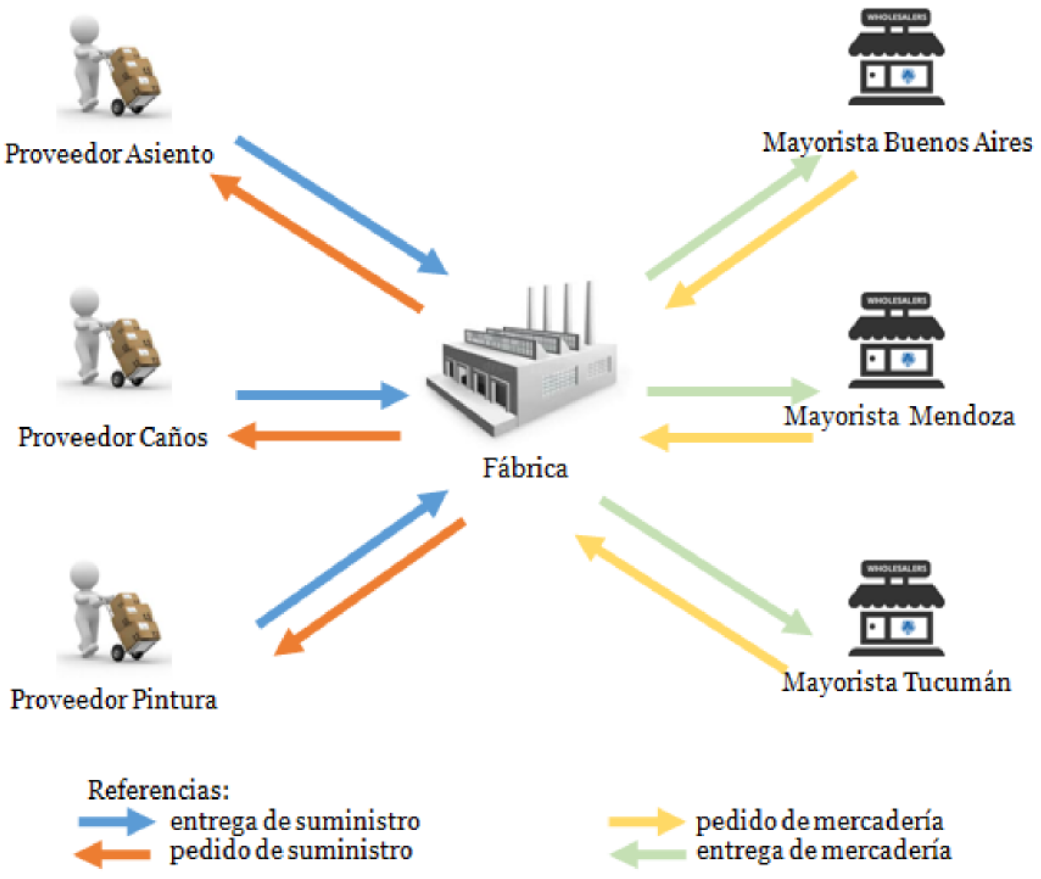


Figura 5.2 – Estructura de la cadena de suministro silla CS

La Tabla 5.1 resume, cuáles son los elementos de la estructura de la cadena de suministro descripta, que se modelan para poder hacer uso del enfoque propuesto.

Tabla 5.1 – Elementos a modelar para definir la estructura de CS

Elementos de la Estructura de CS
Silla CS
Proveedor Asiento
Proveedor Caños
Proveedor Pintura
Fábrica
Mayorista Buenos Aires
Mayorista Mendoza
Mayorista Tucumán
Relación Entrega de Suministro
Relación Pedido de Suministro
Relación Entrega de Mercadería
Relación Pedido de Mercadería

5.3 Modelado de la Cadena de Suministro

Para modelar la cadena de suministro con los conceptos de la red de ontologías SCFHLA, además de la estructura de la cadena, se requiere información asociada con la evaluación del rendimiento de la misma. Esta información se encuentra ligada, al uso del modelo de la cadena, para su posterior simulación. Entonces para modelar el escenario propuesto en la sección anterior, la primera tarea a realizar, es modelar la información de la estructura de la cadena de suministro (resumida en la Tabla 5.1). Luego la siguiente tarea, es modelar la información asociada a la evaluación del rendimiento. Esta última información, contiene la representación de los procesos interorganizacionales de la cadena, la información intercambiada en las relaciones entre los participantes, el objetivo de la CS a alcanzar, el

atributo de rendimiento que define la dirección estratégica, y las métricas para evaluar el grado en que se cumple, con el objetivo propuesto.

En cuanto a la representación gráfica utilizada en las figuras de este capítulo, en las cuales se describen los elementos instanciados de la red de ontologías SCFHLA, se sigue un formato similar al propuesto en (Gómez-Pérez, Fernández-López, y Corcho 2010), en donde:

- Los nodos con líneas punteadas, representan instancias de conceptos.
- Las flechas con líneas punteadas, representan relaciones del tipo *instanceOf*.
- Las flechas con líneas sólidas, representan una relación que no es del tipo *instanceOf*.
- Los nodos grises, representan conceptos de la red de ontologías.
- Las rectangulos con líneas punteadas, representan valores de los atributos.

En cuanto a las relaciones entre los participantes, los mismos acuerdan sobre los materiales o productos que van a intercambiar, y los parámetros que definen la información a intercambiar. Para ello, se identifican las cuatro relaciones presentadas en la Figura 5.2, las cuales se describen a continuación.

La relación *Pedido de Mercadería* entre los clientes mayoristas y la fábrica, representa una solicitud de sillas de caño, por parte de un cliente mayorista hacia la fábrica. En esta relación, se identifican los siguientes parámetros: *Cantidad* que representa la cantidad de las sillas a solicitar, *Item* que representa un identificador único del producto solicitado, y *Fecha Solicitada* que define cuando el producto debe ser entregado al cliente mayorista. La relación *Entrega de Mercadería* entre la fábrica y los clientes mayoristas, representa la entrega de sillas de caño, por parte de la fábrica hacia un cliente mayorista. En esta relación, se identifican los siguientes parámetros: *Cantidad* para representar la cantidad de sillas entregadas, *Item* que es el identificador único del producto entregado, y *Fecha Entrega* que representa cuando las sillas han sido efectivamente entregadas al cliente mayorista.

La relación *Pedido de Suministro* entre la fábrica y los proveedores, representa una solicitud de alguno de los insumos que proveen los proveedores, por parte de la fábrica hacia un proveedor. En esta relación, se identifican los siguientes parámetros: *Cantidad* para

solicitar una determinada cantidad a suministrar, *Item* que identifica unívocamente el suministro solicitado, y *Fecha Solicitada* que establece cuando el suministro debe ser entregado a la fábrica. La relación *Entrega de Suministro* entre los proveedores y la fábrica, representa una entrega del suministro solicitado, por parte de un proveedor hacia la fábrica. En esta relación, se identifican los siguientes parámetros: *Cantidad* para representar la cantidad entregada del suministro, *Item* que es el identificador único del suministro entregado, y *Fecha Entrega* que representa cuando el suministro ha sido efectivamente entregado a la fábrica

Los participantes definen un modelo de objetos de federación con el nombre *Silla CS*, a efectos de vincular al mismo con la federación que representa a la CS, en una etapa posterior.

En la Figura 5.3, se presentan las relaciones entre los participantes de la cadena de suministro *Silla CS*. Como criterio, se adoptan los mismos nombres de los participantes, así como los colores y nombres de las relaciones, que se utilizan en la Figura 5.2.

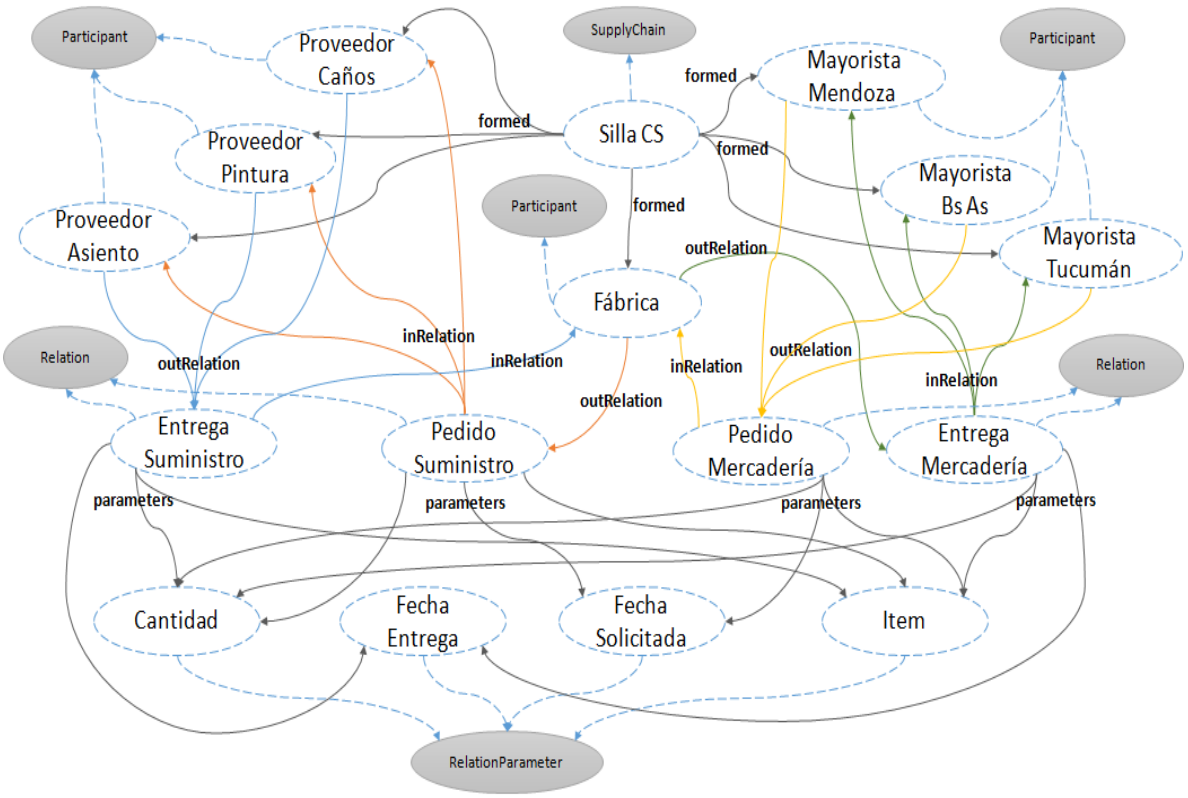


Figura 5.3 – Relaciones entre los participantes de silla CS

En el caso de *Silla CS*, los participantes acuerdan en que el objetivo de la estructura de cadena de suministro es: *lograr que el 95% de los pedidos, se entreguen dentro del plazo estipulado por el cliente para la entrega*. Al definir este objetivo, surge la pregunta de ¿porque los participantes lo han acordado como objetivo a alcanzar? La respuesta es de vital importancia para mantener competitiva a la cadena de suministro, y responde al tipo de producto que se ofrece a los clientes. Los modelos de sillas de caño, son un producto que no se diferencia por su calidad, sino por otros aspectos como pueden ser: su precio, tiempos de respuesta, o nivel de servicio, entre otros. En otras palabras, los clientes eligen a quien le van a comprar las sillas de caño, en base a un precio más económico, a plazos de entrega más cortos, o a un nivel de confiabilidad alto en cuanto al cumplimiento de las entregas. Por este motivo, los participantes concuerdan en que la cadena de suministro se tiene que enfocar, en una estrategia que permita asegurar un alto nivel de servicio, y además cumplir con los plazos de entrega acordados con el cliente.

Al definir una estrategia para asegurar un alto nivel de servicio, los participantes se centran en lograr la satisfacción de los clientes. De este modo, este aspecto resulta ser la ventaja competitiva, que diferencia a la CS de sus competidores; así como también, permite que la cadena logre un buen nivel de competitividad, en el segmento de mercado en el cual compite. Para cumplir con esta estrategia, se opta por un atributo de rendimiento que otorgue una visión externa a la cadena de suministro, que se enfoque en la satisfacción de los clientes. Por lo tanto, el atributo de rendimiento que se selecciona para la cadena, es el denominado en el modelo SCOR como *Reliability* (*Confiabilidad*).

Una vez seleccionado el atributo de rendimiento para *Silla CS*, los participantes deciden cuál de las métricas asociadas al atributo, va a ser utilizada para evaluar la cadena de suministro. De las métricas del modelo SCOR asociadas a la confiabilidad, se detallan solo las de nivel uno y dos, dado que las de nivel tres relevan porcentajes o valores específicos. Por lo que las métricas de nivel tres, no permiten realizar un diagnostico sobre las causas, que provocan el rendimiento de la cadena. Entonces, el conjunto de métricas disponibles para seleccionar es: *Perfect Order Fulfillment* (cumplimiento perfecto de orden), *% of Orders Delivered In Full* (porcentaje de órdenes entregadas completas), *Delivery Performance to Customer Commit Date* (entrega en fecha solicitada por el cliente), *Documentation Accuracy*

(documentación correcta), y *Perfect Condition* (en condición perfecta). De estas opciones, la métrica *Delivery Performance to Customer Date* es la única que evalúa si se cumple o no con la fecha acordada con el cliente final, mientras que las demás métricas, evalúan otros aspectos de la entrega del producto a los clientes. Esta métrica analiza todo el ciclo, entre el punto de entrada de una orden (cuando la fábrica recibe órdenes de compra de los clientes mayoristas) y el punto de entrega de la orden (cuando los clientes reciben el producto final). Por este motivo, los participantes acuerdan seleccionar la métrica *Delivery Performance to Customer Commit Date*.

En cuanto a los procesos de cada organización, los mismos se definen de acuerdo a las relaciones entre los participantes y la política de trabajo de cada uno. Los proveedores de *Silla CS*, reciben órdenes de suministro de la fábrica y envían los suministros a la fábrica; por lo que, los procesos requeridos para llevar a cabo tales relaciones son *Plan* (Planear) y *Deliver* (Entregar) respectivamente. Además, la fábrica emite órdenes de suministro a los proveedores, recibe los materiales de los proveedores, recibe órdenes de compra de sillas por parte de los clientes mayoristas; y, finalmente, envía el producto final a los mayoristas. Por lo que, los procesos de la fábrica identificados para realizar estas actividades son: *Source* (Abastecer) para las primeras dos actividades, *Plan* para la recepción de órdenes de compra de las mueblerías mayoristas, y *Deliver* para enviar las sillas a los clientes. Por último, los clientes mayoristas emiten órdenes de compra hacia la fábrica, para solicitar la mercadería que necesitan; y a su vez, reciben los pedidos de la mercadería solicitada. Para realizar ambas tareas, los clientes mayoristas han identificado el proceso *Source*.

Cabe destacar que, como se menciono anteriormente en la sección 5.2 (Descripción del Escenario), todos los participantes tienen una política de producción por orden. Es decir que la prioridad, es cubrir la demanda de productos con el inventario. En caso de que el inventario no sea suficiente, se producen las unidades necesarias para satisfacer la demanda; de modo tal que la cantidad de unidades en existencia sea la mínima posible. Entonces para realizar las actividades de la política de producción por orden, se identifica que cada proceso *Source*, posee un subproceso *Source Make to Order Product* (Abastecer Producto Por Orden); y cada proceso *Deliver*, cuenta con un subproceso *Deliver Make to Order Product* (Entregar Producto Por Orden). Además, de acuerdo al proceso de negocio de la Figura 5.1, cada

participante crea un plan maestro producción para sus operaciones semanales. Por lo que, para realizar las actividades relacionadas con la creación del plan, cada proceso *Plan*, contiene un subproceso *Plan Make (*Planear para Producir).

Por último, resta decidir los roles de cada uno de los participantes. Se identifica a los proveedores con el rol *Source*, dado que su función principal en la CS, es abastecer de suministros a la fábrica. Por otro lado, la fábrica se vincula con el rol *Make*, debido a que transforma los insumos que recibe de los proveedores, en el producto final que luego vende la cadena de suministro. Finalmente, los clientes mayoristas se asocian con el rol *Deliver*, ya que son los encargados de recibir las sillas de caño y comercializarlas. Además de estos roles, los participantes deciden quién tendrá el rol de autoridad, dentro de la cadena de suministro. Como el participante que representa a la fábrica, es el único que interactúa con todos los demás, se decide que este participante sea el que cumpla el rol *Authority* (autoridad).

La Figura 5.4 presenta la métrica seleccionada para la CS *Silla CS*, la relación de esta métrica con los subprocesos de cada participante, el atributo de rendimiento que evalúa el objetivo propuesto, los roles de cada participante, y los participantes de la cadena.

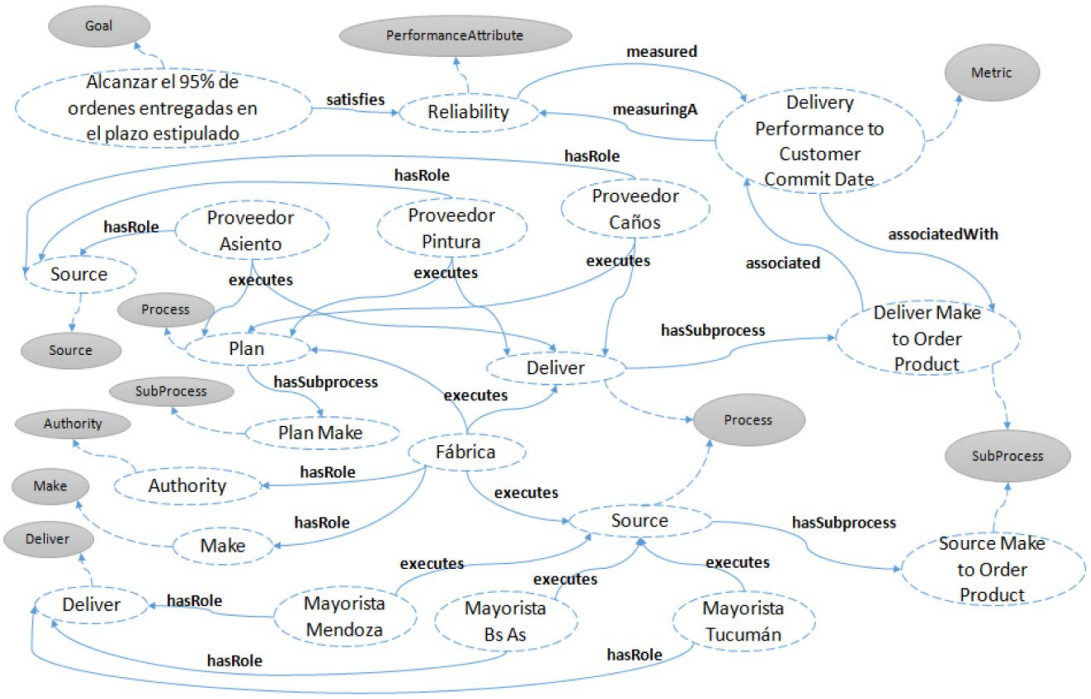


Figura 5.4 – Métricas, atributos de rendimiento y su relación con los procesos en silla CS

La Tabla 5.2 resume cuales son los elementos, para la evaluación del rendimiento de la cadena de suministro descripta, que se deben modelar para utilizar el enfoque propuesto.

Tabla 5.2 – Elementos a modelar para la evaluación del rendimiento de CS

Elementos para la Evaluación del Rendimiento de CS
Parámetro de Relación Cantidad
Parámetro de Relación Item
Parámetro de Relación Fecha Solicitada
Parámetro de Relación Fecha Entrega
Objetivo *CS*
Atributo de Rendimiento Reliability
Metrica Delivery Performance to Customer Commit Date
Proceso Plan
Proceso Source
Proceso Deliver
Subproceso Source Make to Order Product
Subproceso Deliver Make to Order Product
Subproceso Plan Make
Rol Source
Rol Deliver
Rol Make
Rol Authority
FOM Silla CS

5.4 Inferencia de la Federación HLA

En la sección anterior los participantes definieron toda la información necesaria, para instanciar un modelo de cadena de suministro, en la red de ontologías SCFHLA. Siguiendo el flujo de trabajo propuesto en la Figura 4.2, la siguiente actividad implica utilizar la información definida para instanciar la red de ontologías, y luego ejecutar las reglas SWRL. Por medio de la ejecución de las reglas SWRL, se obtienen los conceptos derivados del modelo de cadena de suministro, y se infieren los conceptos de la federación HLA, asociados a la cadena denominada *Silla CS*.

A continuación, se detalla cómo se aplican las reglas SWRL definidas en las ecuaciones 3.1 a 3.24, al caso de estudio propuesto. Vale aclarar que, las reglas se activan o no, dependiendo del escenario propuesto; pero en cualquier caso, las reglas se aplican para realizar las derivaciones y mapeos necesarios, entre los dominios de la red de ontologías.

A partir de la regla definida en la Ecuación 3.1, a la métrica con nombre *Delivered Performance to Customer Commit Date*, elegida en el escenario para medir el atributo *Reliability*, se le asocia el valor "*RL.2.2*" como codificación de métrica, para su atributo *metricID*. A partir de la regla definida en la Ecuación 3.4, a la métrica con codificación de métrica *RL.2.2*, se le asocia el valor dos como nivel de métrica, para su atributo *metricLevel*.

A partir de regla definida en la Ecuación 3.2, la métrica con codificación de métrica RL.2.2, y la fórmula con el valor "*[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%*" para el atributo *equation*, se vinculan por medio de la relación inferida *hasFormula*. Según lo define la Ecuación 3.3, a la formula con el valor "*[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%*" para el atributo *equation*, se le asocia el valor de "*%*" (porcentaje) como unidad, para su atributo *formulaUnit*.

A partir de la regla definida en la Ecuación 3.5, la formula con el valor "*[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%*" para el atributo *equation*, y las variables complejas nombradas como *Customer Commit Date Achievement* y *Delivery Location*, se vinculan mediante la relación inferida

hasVariable. De acuerdo a lo establecido en la Ecuación 3.6, la variable compleja con nombre *Customer Commit Date Achievement* y la métrica con nombre *Customer Commit Date Achievement Time Customer Receiving*, se vinculan a través de la relación inferida *hasMetric*. A su vez, la variable compleja con nombre *Delivery Location* y la métrica con nombre *Delivery Location Accuracy*, se vinculan por medio de la relación inferida *hasMetric*.

A efectos de continuar derivando información, se vuelven a aplicar las reglas definidas en las ecuaciones 3.1 a 3.4. De este modo, se espera vincular las métricas *Customer Commit Date Achievement Time Customer Receiving* y *Delivery Location Accuracy*, con sus respectivas fórmulas. En ambos casos, las métricas son de nivel tres y, por lo tanto, las variables de sus fórmulas son atómicas. Además, ambas fórmulas poseen la unidad "*%*" (porcentaje), como unidad para su atributo *formulaUnit*.

A su vez, por medio de la regla definida en la Ecuación 3.5, se vinculan las variables atómicas con nombre *Received On Time* y *Total Delivered*, a la formula de la métrica con nombre *Customer Commit Date Achievement Time Customer Receiving*, mediante la relación inferida *hasVariable*. Por último, al volver a aplicar la regla de la Ecuación 3.5, se vinculan las variables atómicas con nombre *Delivered On Correct Location* y *Total Delivered*, a la formula de la métrica con nombre *Delivery Location Accuracy*, a través de la relación inferida *hasVariable*.

En la Figura 5.5, se presentan los individuos instanciados a partir de la información del escenario propuesto. Además, la Figura 5.5 presenta las relaciones y atributos derivados, a partir de la ejecución de las reglas SWRL descriptas anteriormente.

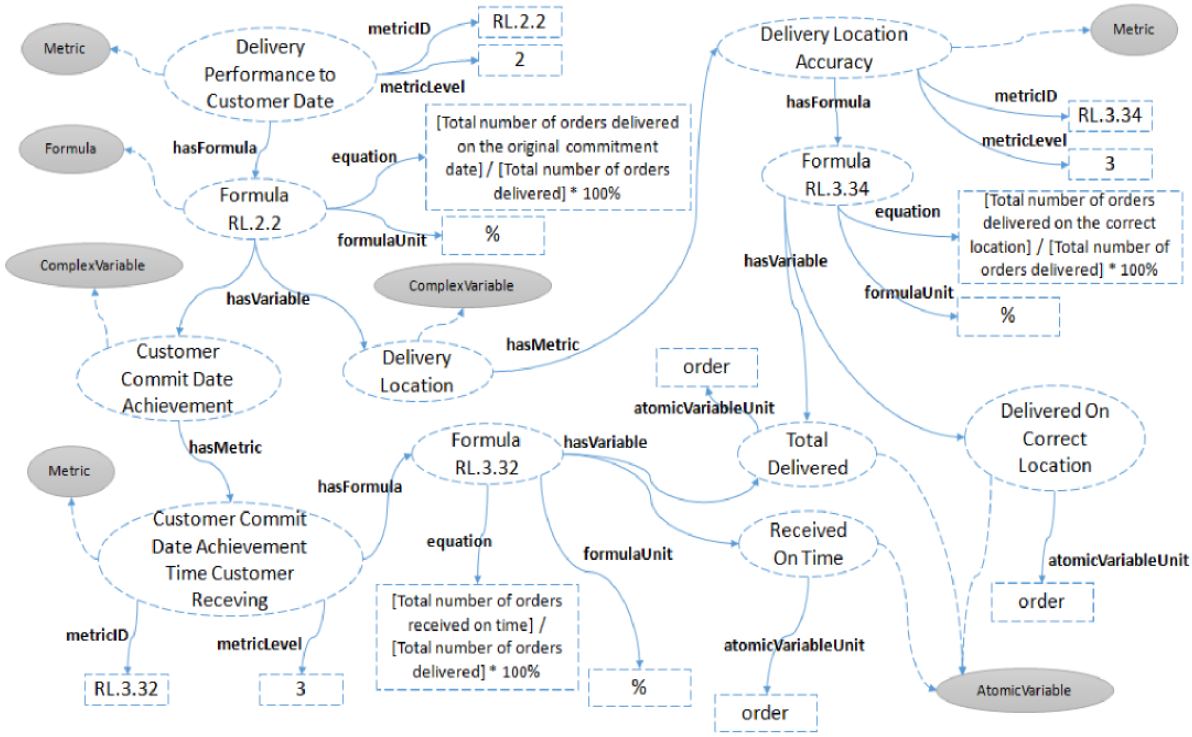


Figura 5.5 – Descomposición de la métrica seleccionada para la cadena silla CS

A continuación, se realizan los mapeos necesarios para vincular los conceptos de cadenas de suministro con los conceptos de federaciones HLA.

Dada la regla definida en la Ecuación 3.14, la cadena de suministro con nombre *Silla CS* se mapea como instancia del concepto *Federation*, con el mismo nombre como valor del atributo *federationName*. De acuerdo a lo establecido en la Ecuación 3.16, los participantes con nombres *Proveedor Asiento*, *Proveedor Pintura*, *Proveedor Caños*, *Fábrica*, *Mayorista Mendoza*, *Mayorista Bs As*, y *Mayorista Tucumán*, se mapean como instancias del concepto *Federate*, con los mismos nombres como valores del atributo *federateName* respectivamente.

De acuerdo a lo establecido en la Ecuación 3.18, los federados que representan a los participantes de la cadena, se vinculan con la federación que representa a la cadena *Silla CS*. Para realizar esta vinculación, se infiere la relación *hasMember* entre la federación con nombre *Silla CS*, y cada uno de los federados. Como lo describe la Ecuación 3.22, existe un participante de la cadena de suministro que cumple el rol de autoridad, y por lo tanto, el federado que representa a este participante, se vuelve el administrador de la federación. Entonces la instancia del concepto *Authority*, asociada al participante que cumple el rol de

autoridad, se mapea como instancia del concepto *Administrator*, y además se vincula con el federado *Fábrica* por medio de la relación *hasFunction*. La instancia del concepto *Administrator*, se vincula con la federación *Silla CS* a través de la relación *initiates*.

Por medio de la Ecuación 3.15, se vincula a la federación con la cadena de suministro que representa. Para realizar esta asociación, se infiere la meta relación *performs* entre la federación *Silla CS* y la cadena de suministro con el mismo nombre. Además, de acuerdo con la Ecuación 3.17, se asocia a cada federado con el participante de la cadena que representa. Por lo que, para realizar este vinculo, se infiere la meta relación *representA* entre cada federado y cada participante, que posean el mismo nombre. Como se describe en la Ecuación 3.9, una federación HLA debe tener un modelo de objetos de federación asociado. Por este motivo, se infiere la relación *hasFOM* entre la federación HLA y una instancia del concepto *FOM*, con el mismo nombre que la federación.

Por último, para culminar con la conformación de la federación HLA, que representa a la cadena de suministro *Silla CS*, se mapea el objetivo de la cadena como objetivo de la federación. Según la Ecuación 3.24, dado el objetivo de la cadena *Silla CS*, con el atributo *goalDescription* en el valor "*lograr que el 95% de los pedidos, se entreguen dentro del plazo estipulado por el cliente para la entrega*"; entonces se mapea este objetivo de CS, como instancia del concepto *Objective*, con el atributo *objectiveDescription* en el mismo valor que el atributo *goalDescription*. Además, se asocia a la federación *Silla CS* con la instancia del concepto *Objective*, por medio de la relación inferida *hasObjective*.

En la Figura 5.6 se presentan los mapeos, las relaciones, y atributos, inferidos a partir de la ejecución de las reglas SWRL descriptas anteriormente.

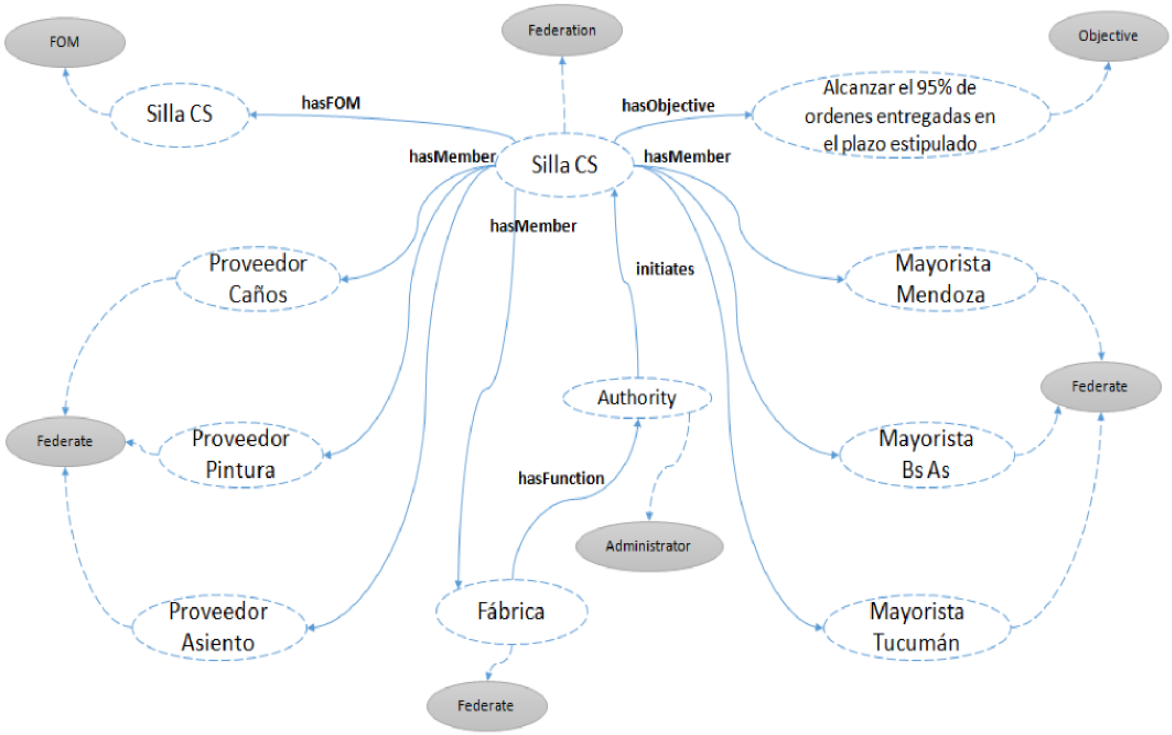


Figura 5.6 – Conformación de la federación silla CS

Hasta el momento, mediante la ejecución de las reglas SWRL descriptas, se ha conformado la federación HLA. Además de conformar la federación, es necesario mapear todos los conceptos relacionados con el modelo de objetos de federación.

A partir de la regla definida en la Ecuación 3.19, se mapean las relaciones nombradas como *Entrega Suministro*, *Pedido Suministro*, *Pedido Mercadería*, y *Entrega Mercadería* entre los participantes de la cadena, como instancias del concepto *InteractionClass* (es decir, como interacciones en la federación HLA), las cuales poseen los mismos nombres respectivamente. De acuerdo a lo expresado en la Ecuación 3.20, cada parámetro de una relación, se mapea como instancia de un parámetro de clase de interacción. Por lo que, los parámetros del escenario con nombre *Cantidad*, *Fecha Entrega*, *Fecha Solicitada*, e *Item* (instancias del concepto *RelationParameter*), se mapean como instancias del concepto *Parameter* con los mismos nombres respectivamente.

Además de mapear los parámetros, también se mapean los tipos de datos de parámetro. De este modo, *Cantidad* e *Item* tienen como tipo de dato de parámetro, un numero decimal

con precisión doble (que se representa con el tipo de dato *float*); mientras que *Fecha Solicitada* y *Fecha Entrega* tienen como tipo de dato de parámetro, una fecha (representado con el tipo de dato *date*). Según se define en la Ecuación 3.21, cada variable atómica con sus atributos *atomicVariableUnit* y *atomicVariableValue*, se mapea como una clase de objeto con los mismos atributos. Entonces, las variables atómicas del escenario con los nombres *Received On Time*, *Delivered On Correct Location*, y *Total Delivered*, se mapean como instancias del concepto *ObjectClass* con los mismos nombres respectivamente. Además, se infiere la relación *hasAttribute*, entre la clase de objeto y las instancias del concepto *Attribute* denominadas *unit* y *value*.

Como lo establece la Ecuación 3.10, cada clase de objeto mapeada a partir de las variables atómicas, debe estar contenida en un modelo de objetos de federación. Por lo que, las clases de objeto *Received On Time*, *Delivered On Correct Location*, y *Total Delivered*, se vinculan con el FOM cuyo nombre es *Silla CS*, a través de la relación inferida denominada *userObject*. Como lo define la regla de la Ecuación 3.11, cada clase de interacción mapeada a partir de las relaciones entre participantes, debe estar contenida en un modelo de objetos de federación. Debido a este motivo, las clases de interacción *Entrega Suministro*, *Pedido Suministro*, *Pedido Mercadería*, y *Entrega Mercadería*, se vinculan con el FOM cuyo nombre es *Silla CS*, mediante la relación inferida denominada *userInteraction*.

De acuerdo con la Ecuación 3.23, se asegura que los parámetros vinculados a las clases de interacción, se corresponden con los parámetros vinculados a las relaciones de cadena de suministro. Por lo tanto, para el escenario propuesto se asocia la clase de interacción *Entrega Suministro*, con los parámetros *Cantidad*, *Item*, y *Fecha Entrega*, por medio de la relación inferida *hasParameter*. De este mismo modo, se procede con las demás interacciones y cada uno de sus respectivos parámetros.

Como se expreso en la Ecuación 3.12, los atributos de una clase de objeto, se asocian con sus propiedades. Entonces, el atributo *Unit* se vincula con la propiedad *unitProperty*, a través de la relación inferida *attributeProperty*. Para el caso del atributo *Value*, se vincula con la propiedad *valueProperty*, mediante la relación inferida *attributeProperty*. A partir de la regla definida en la Ecuación 3.13, cada clase de interacción, se asocia a un conjunto de

propiedades que se utilizan por defecto. Por lo que, las interacciones *Entrega Suministro*, *Pedido Suministro*, *Pedido Mercadería*, y *Entrega Mercadería*, se vinculan con la propiedad *icProperty*, por medio de la relación inferida *interactionProperty*.

En la Figura 5.7 se presentan las instancias de los conceptos, que se encuentran relacionados con el modelo de objetos de la federación denominada *Silla CS*. Luego de la ejecución de las reglas, se han mapeado todos los conceptos del modelo de cadena de suministro, a los conceptos de la federación HLA.

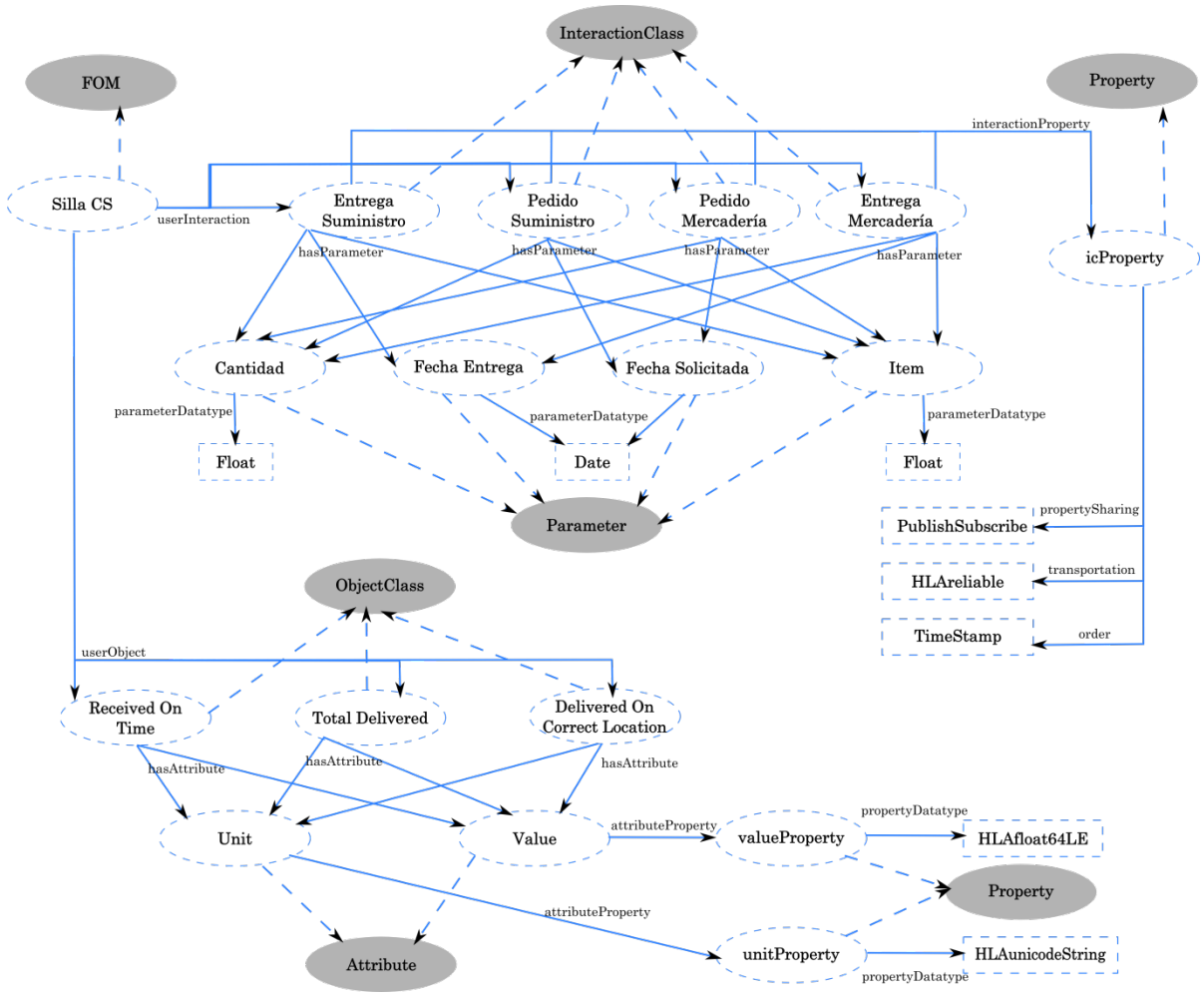


Figura 5.7 – Conceptos relacionados con el modelo de objetos de federación de silla CS

La Tabla 5.3 resume cuales son los elementos inferidos y cuales son los elementos asociados a otros elementos, luego de la aplicación de las reglas SWRL, para la federación HLA.

Tabla 5.3 – Elementos inferidos y asociados de la federación HLA

Elementos Inferidos por las Reglas SWRL	Elementos Asociados por las Reglas SWRL
Federación Silla CS	**Fórmula *[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%***
Objetivo Federación	**Variable Compleja** ***Customer Commit Date Achievement***
Federado Proveedor Caños	**Variable Compleja *Delivery Location***
Federado Proveedor Pintura	**Métrica *Customer Commit Date Achievement Time Customer Receiving***
Federado Proveedor Asiento	**Métrica *Delivery Location Accuracy***
Federado Mayorista Mendoza	**Fórmula *[Total number of orders received on time] / [Total number of orders delivered] x 100%***
Federado Mayorista Buenos Aires	**Fórmula *[Total number of orders delivered on the correct location] / [Total number of orders delivered] x 100%***
Federado Mayorista Tucumán	**Variable Atómica *Received On Time***
Federado Fábrica	**Variable Atómica *Total Delivered***
Administrador Authority	**Variable Atómica *Delivered On Correct Location***
Clase de Interacción Entrega Suministro	**Propiedad *icProperty***
Clase de Interacción Pedido Suministro	**Atributo *Unit***
Clase de Interacción Entrega Mercadería	**Atributo *Value***
Clase de Interacción Pedido Mercadería	**Propiedad *unitProperty***
Clase de Objeto Received On Time	**Propiedad *valueProperty***
Clase de Objeto Total Delivered	
Clase de Objeto Delivered On Correct Location	

5.5 Transformación a un Modelo de Objetos de Federación

De acuerdo al flujo de trabajo presentado en la Figura 4.2, una vez que se han ejecutado las reglas SWRL, sobre la red de ontologías SCFHLA, la siguiente actividad a realizar es: almacenar la información derivada y mapeada, de la red de ontologías. En este caso, esta actividad requiere almacenar la red de ontologías con la información inferida, en el formato RDF/XML, desde el menú de la herramienta *Protégé*.

La siguiente actividad consiste en aplicar el algoritmo de transformación, por lo cual se llevan a cabo las dos etapas del mismo. La primera etapa, consiste en recopilar la información de gestión de los servicios del RTI, para lo cual se utiliza la información del archivo de configuración; y, a partir del mismo, se crea un nuevo documento XML que contiene la información de configuración. Luego, la segunda etapa implica la búsqueda y transformación, de los individuos de la red de ontologías sobre el documento OWL. Para realizar esta actividad, por un lado se transforman las clases de objeto con sus atributos, y propiedades; mientras que, por otro lado, se transforman las clases de interacción con sus propiedades, y parámetros.

La separación de la transformación en dos partes, responde a la estrategia que se utiliza, para recorrer el documento OWL de la red de ontologías, la cual se denomina “primero en profundidad”. A continuación, se desarrolla la explicación sobre cómo llevar a cabo esta transformación, para el escenario propuesto.

5.5.1 Transformar las clases de objeto, atributos y propiedades

Se recorre el documento en la búsqueda de los individuos. En esta parte del algoritmo, se busca a cada individuo cuyo atributo *type* contenga el valor *ObjectClass*, es decir, se buscan los individuos que son instancia del concepto *ObjectClass*, de la red de ontologías. Cada individuo de este tipo, se transforma en un elemento equivalente del tipo *objectClass*, en el documento XML. En el escenario propuesto, las tres instancias de este tipo son: *Received On Time*, *Total Delivered*, y *Deliver On Correct Location*. Para el caso del individuo *Total Delivered*, se almacena su nombre el cual contiene el valor *Total Delivered*; y

además se almacena la forma en que se comparte, la cual contiene el valor *PublishSubscribe*. Esta información, se encuentra en las etiquetas *objectName* y *objectSharing* respectivamente. Además, tanto el nombre como la forma en que se comparte un objeto, se guardan en el documento XML, bajo las etiquetas *name* y *sharing* respectivamente.

La próxima información que se encuentra al recorrer el individuo, es aquella vinculada a los atributos, los cuales a su vez, también son individuos. Los atributos se identifican en la clase de objeto, por medio de la etiqueta *hasAttribute*, la cual almacena el nombre del individuo. Cada individuo de este tipo, se transforma en un elemento equivalente del tipo *attribute*, en el documento XML. Para el escenario propuesto los atributos son dos, y se denominan como *Unit* y *Value*. Para obtener la información de los atributos se continúa con la búsqueda de individuos, pero ahora del tipo *Attribute*, y con la etiqueta *NamedIndividual* en el valor *Unit* primero, y luego en el valor *Value*.

Cuando se encuentra el individuo con el valor *Unit* en la etiqueta *NamedIndividual*, se verifica que su etiqueta *type* contenga el valor *Attribute*, y entonces se almacena su nombre, el cual se encuentra en la etiqueta *attributeName* con el valor *Unit*.

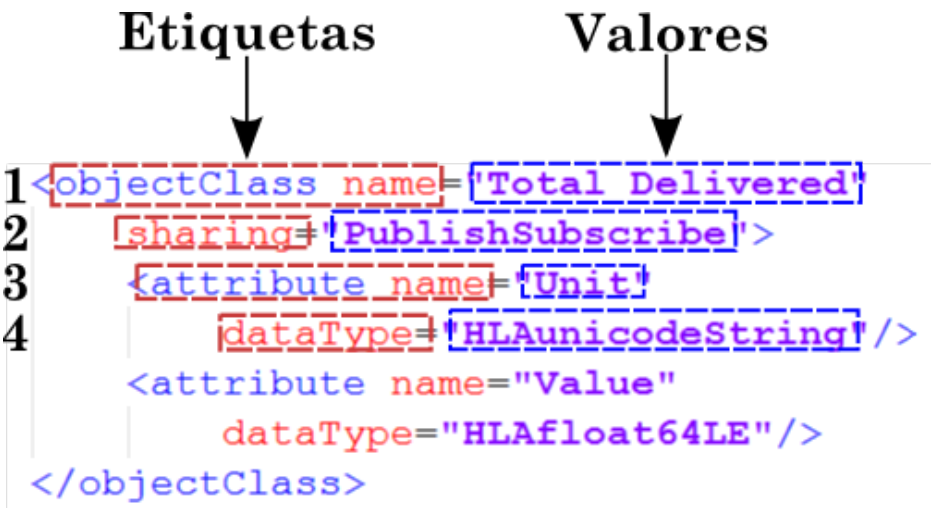
Al continuar el recorrido del documento OWL, se encuentran las propiedades del atributo, las cuales también son individuos. Las propiedades del atributo se identifican con la etiqueta *attributeProperty*, que almacena el nombre del individuo. Cada característica de los individuos de este tipo, se transforma en un elemento equivalente con el mismo nombre que la característica identificada, y el nombre de esta característica se utiliza como etiqueta en el documento XML. Para el caso del atributo *Unit* existe una propiedad que se denomina *UnitProperty*, y para obtener su información, se continúa con la búsqueda de individuos pero ahora del tipo *Property*, y además con la etiqueta *NamedIndividual* en el valor *UnitProperty*.

Al encontrar el individuo con el valor *UnitProperty* en la etiqueta *NamedIndividual*, se verifica que su etiqueta *type* contenga el valor *Property*, y entonces se almacenan sus características. En este caso dicha propiedad posee una única característica, que es el tipo de dato, y se encuentra en la etiqueta *propertyDatatype* con el valor *HLAunicodeString*. Esta característica se almacena en el documento XML, con la etiqueta *datatype*, dentro de la etiqueta de su atributo correspondiente.

Una vez guardada la característica, se procede a repetir el proceso de búsqueda descripto en los tres párrafos precedentes, para el individuo *Value* del tipo *Attribute*. Una vez que se ha guardado el atributo *Value* y sus propiedades, se ha finalizado la transformación del individuo *Total Delivered*, del tipo *ObjectClass*. El procedimiento descripto, se repite de manera similar para los otros dos individuos de este tipo; luego de lo cual, se da por finalizada la transformación de las clases de objeto junto con sus atributos y propiedades, para el escenario propuesto.

En la Figura 5.8 se presenta como es la estructura de los individuos del tipo *ObjectClass*, tanto en el documento OWL (a), como en el XML (b). En la Figura 5.8 se exhibe, a modo de ejemplo, al individuo *Total Delivered*. Además, se destacan los pasos seguidos para su búsqueda en el documento OWL, y para su posterior inserción en el documento XML.

(a) Documento OWL



(b) Documento XML

Figura 5.8 – Transformación de un objeto con sus atributos y propiedades

5.5.2 Transformar las clases de interacción, propiedades y parámetros

Se recorre el documento en la búsqueda de los individuos. En esta parte del algoritmo, se busca a cada individuo cuyo atributo *type* contenga el valor *InteractionClass*, es decir, se buscan los individuos que son instancia del concepto *InteractionClass*, de la red de ontologías. Cada individuo de este tipo, se transforma en un elemento equivalente del tipo *interactionClass,* en el documento XML. En el escenario propuesto, las cuatro instancias de este tipo son: *Entrega Suministro*, *Pedido Suministro*, *Pedido Mercadería*, y *Entrega Mercadería*. Para el caso del individuo *Pedido Suministro*, se almacena su nombre el cual contiene el valor *Pedido Suministro*, y se encuentra en la etiqueta *interactionName*. Además, el nombre se guarda en el documento XML, bajo la etiqueta *name*.

Al continuar el recorrido del documento OWL, se encuentran las propiedades de la interacción que, a su vez, son otro individuo. Las propiedades se identifican en la clase de interacción, con la etiqueta *interactionProperty* que almacena el nombre del individuo. Cada característica de los individuos de este tipo, se transforma en un elemento equivalente con el mismo nombre que la característica identificada, y el nombre de esta característica se utiliza como etiqueta en el documento XML. Para el caso de la clase de interacción *Pedido Suministro* existe una propiedad que se denomina *icPropety*, y para obtener su información,

se continúa con la búsqueda de individuos pero ahora del tipo *Property*, y además con la etiqueta *NamedIndividual* en el valor *icProperty*.

Al encontrar el individuo con el valor *icProperty* en la etiqueta *NamedIndividual*, se verifica que su etiqueta *type* contenga el valor *Property*, y entonces se almacenan sus características. En este caso la propiedad posee tres características, las cuales son: la forma en que se comparte, con el valor *PublishSubscribe* en la etiqueta *propertySharing*; el modo en que se transporta, con el valor *HLAreliable* en la etiqueta *transportation*; y por último, con qué criterio se ordena, con el valor *TimeStamp* en la etiqueta *order*. Estas características se almacenan en el documento XML, con las etiquetas *sharing*, *transportation*, y *order* respectivamente, dentro de la etiqueta de su interacción correspondiente.

Una vez guardadas las características, se procede a continuar recorriendo el documento OWL, a efectos de encontrar los parámetros de la clase de interacción. Estos parámetros de clase de interacción, son a su vez individuos. Los parámetros de una clase de interacción se identifican, por medio de la etiqueta *hasParameter*, la que almacena el nombre del individuo. Cada individuo de este tipo, se transforma en un elemento equivalente del tipo *parameter*, en el documento XML. Para el escenario propuesto los parámetros son tres, y se denominan como: *Fecha Solicitada*, *Cantidad*, e *Item*. Para obtener la información de los parámetros se continúa con la búsqueda de individuos, pero ahora del tipo *Parameter*, y con la etiqueta *NamedIndividual* en el valor *Fecha Solicitada* primero, luego en el valor *Cantidad*, y por último en el valor *Item*.

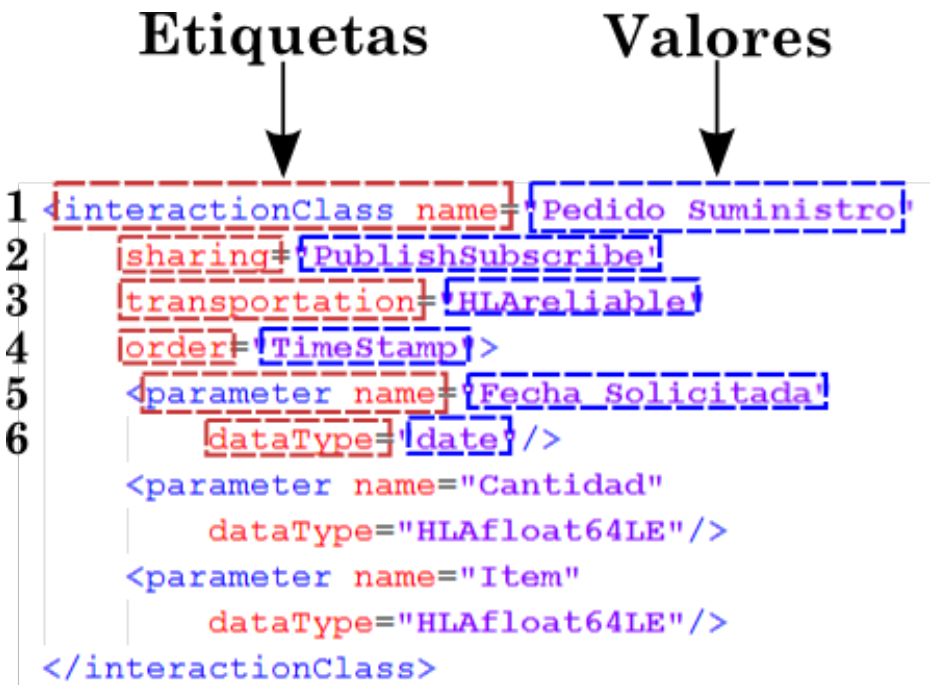
Cuando se encuentra el individuo con el valor *Fecha Solicitada* en la etiqueta *NamedIndividual*, se verifica que su etiqueta *type* contenga el valor *Parameter*, y entonces se almacena su nombre, el cual se encuentra en la etiqueta *parameterName* con el valor *Fecha Solicitada*. Luego se almacenan sus características. En este caso, la única característica que posee este parámetro es su tipo de dato, y se encuentra en la etiqueta *parameterDatatype* con el valor *Date*. Esta característica se almacena en el documento XML, con la etiqueta *datatype*, dentro de la etiqueta de su parámetro correspondiente.

Una vez guardada la característica, se repite el proceso de búsqueda descripto en los dos párrafos precedentes, para el individuo *Cantidad* y para el individuo *Item*, ambos del tipo

Parameter. Esta búsqueda se realiza de manera similar, a la detallada para el individuo *Fecha Solicitada* del tipo *Parameter*. Una vez que se han guardado los parámetros *Cantidad* e *Item* junto con sus características, se ha finalizado la transformación del individuo denominado *Pedido Suministro* del tipo *InteractionClass*. El procedimiento descripto, se repite de manera similar para los otros tres individuos de este tipo; luego de lo cual, se finaliza la transformación de las clases de interacción junto con sus propiedades y parámetros, para el escenario propuesto.

En la Figura 5.9 se presenta como es la estructura de los individuos del tipo *InteractionClass*, tanto en el documento OWL (a), como en el XML (b). En la Figura 5.9 se presenta, como ejemplo, al individuo *Pedido Suministro*. Además, se resaltan los pasos seguidos para la búsqueda en el documento OWL, y para su posterior inserción en el documento XML.

(a) Documento OWL



(b) Documento XML

Figura 5.9 – Transformación de una interacción con sus propiedades y parámetros

Una vez terminada la transformación de las interacciones junto con sus propiedades y parámetros, el algoritmo finaliza y produce como salida el documento en formato XML, el cual puede ser utilizado como modelo de objetos de federación, para el escenario propuesto. En el Anexo B se presenta el documento obtenido, como salida del algoritmo de transformación.

La Tabla 5.4 resume cuales son los elementos del escenario descripto, cómo se modelan dichos elementos con los conceptos de la red de ontologías SCFHLA, tanto en el dominio de cadenas de suministro como de federaciones HLA, y además, cuales son las etiquetas XML que se obtienen para los elementos del escenario, por medio del algoritmo de transformación.

Tabla 5.4 – Equivalencias entre los elementos del escenario, los conceptos de la red y las etiquetas XML

Elementos	Red de Ontologías SCFHLA		Etiquetas XML
	Ontología SCK	Ontología HLAFed	
Silla CS	**SupplyChain**	**Federation**	**-**
Objetivo *lograr que el 95% de los pedidos se entreguen dentro del plazo estipulado por el cliente para la entrega*	**Goal**	**Objective**	**-**
Proveedor Caños	**Participant**	**Federate**	**-**
Proveedor Pintura	**Participant**	**Federate**	**-**
Proveedor Asiento	**Participant**	**Federate**	**-**
Fábrica	**Participant**	**Federate**	**-**
Mayorista Mendoza	**Participant**	**Federate**	**-**
Mayorista Buenos Aires	**Participant**	**Federate**	**-**
Mayorista Tucumán	**Participant**	**Federate**	**-**
Relación Entrega Suministro	**Relation**	**InteractionClass**	**Líneas 538 a 548**
Relación Pedido Suministro	**Relation**	**InteractionClass**	**Líneas 527 a 537**
Relación Entrega Mercadería	**Relation**	**InteractionClass**	**Líneas 560 a 570**
Relación Pedido Mercadería	**Relation**	**InteractionClass**	**Líneas 549 a 559**
Cantidad	**RelationParameter**	**Parameter**	**Líneas 533 a 534**
Item	**RelationParameter**	**Parameter**	**Líneas 535 a 536**
Fecha Solicitada	**RelationParameter**	**Parameter**	**Líneas 531 a 532**

Elementos	Red de Ontologías SCFHLA		Etiquetas XML
	Ontología SCK	Ontología HLAFed	
Metrica *Delivery Performance to Customer Commit Date*	Metric	-	-
Fecha Entrega	RelationParameter	Parameter	Líneas 542 a 543
Atributo de Rendimiento *Reliability*	PerformanceAttribute	-	-
Proceso *Plan*	Process	-	-
Proceso *Source*	Process	-	-
Proceso *Deliver*	Process	-	-
Subproceso *Source Make to Order Product*	SubProcess	-	-
Subproceso *Deliver Make to Order Product*	SubProcess	-	-
Subproceso *Plan Make*	SubProcess	-	-
Rol *Source*	Source	-	-
Rol *Deliver*	Deliver	-	-
Rol *Make*	Make	-	-
Rol *Authority*	Authority	Administrator	-
Variable Compleja *Customer Commit Date Achievement*	ComplexVariable	-	-
Fórmula *[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%*	Formula	-	-
FOM *Silla CS*	-	FOM	-

Elementos	Red de Ontologías SCFHLA		Etiquetas XML
	Ontología SCK	Ontología HLAFed	
Métrica *Customer Commit Date Achievement Time Customer Receiving*	Metric	-	-
Variable Compleja *Delivery Location*	ComplexVariable	-	-
Métrica *Delivery Location Accuracy*	Metric	-	-
Fórmula *[Total number of orders received on time] / [Total number of orders delivered] x 100%*	Formula	-	-
Fórmula *[Total number of orders delivered on the correct location] / [Total number of orders delivered] x 100%*	Formula	-	-
Variable Atómica *Received On Time*	AtomicVariable	ObjectClass	Líneas 34 a 40
Variable Atómica *Total Delivered*	AtomicVariable	ObjectClass	Líneas 27 a 33
Variable Atómica *Delivered On Correct Location*	AtomicVariable	ObjectClass	Líneas 41 a 47
Propiedad *icProperty*	-	Property	Líneas 528 a 530
Atributo *Unit*	-	Attribute	Líneas 29 a 30
Atributo *Value*	-	Attribute	Líneas 31 a 32
Propiedad *unitProperty*	-	Property	Línea 30
Propiedad *valueProperty*	-	Property	Línea 32

5.6 Conclusiones

En este capítulo se presenta un escenario de cadena de suministro, a efectos de utilizar los componentes propuestos en esta tesis, para la construcción del modelo de objetos de federación. De este modo, se realiza una prueba de concepto sobre la factibilidad de utilizar el enfoque propuesto, para especificar un modelo de interoperabilidad. Además, se presenta como a partir de este modelo de interoperabilidad, se construye el modelo de objetos de federación en el formato adecuado, y se garantiza que la información es correcta.

Por medio del escenario presentado, se especifican las tareas necesarias para que los analistas de negocios, utilicen los componentes definidos (red de ontologías SCFHLA y algoritmo de transformación), a efectos de construir el modelo de objetos de federación. Este documento es fundamental, para desarrollar una simulación distribuida basada en HLA, del escenario de cadena de suministro propuesto.

Mediante las actividades que se desarrollan en esta prueba de concepto, se verifica la factibilidad de construir un modelo de interoperabilidad, el cual cumple con la condición de ser semánticamente interoperable. A través del uso de la red de ontologías SCFHLA y el algoritmo de transformación, el caso propuesto manifiesta los beneficios del enfoque, en cuanto a reducción del conocimiento necesario, así como también la disminución del tiempo y esfuerzo, invertido por los participantes en las tareas realizadas.

El escenario desarrollado, presenta la amplia gama de consideraciones que se deben tener en cuenta, al momento de construir un modelo de objetos de federación. Este escenario pretende ser una guía, sobre cómo utilizar los componentes desarrollados en esta tesis, a efectos de que sea factible producir uno de los documentos necesarios, para construir una simulación distribuida basada en HLA de cadenas de suministro.

Capítulo 6 **Conclusiones y Trabajos Futuros**

6.1 Conclusiones

En el contexto actual donde las relaciones entre las organizaciones, suelen ser definidas para ciertas y determinadas situaciones, donde la asociación para la colaboración o trabajo mancomunado en una cadena de suministro, se produce con una dinámica cambiante y por un corto plazo de tiempo; donde las organizaciones son reticentes a compartir información sobre sus procesos de negocio internos, con el resto de las organizaciones que se vinculan; resulta evidente que en esta situación, es necesario contar con herramientas que brinden soporte, para la toma de decisión conjunta o colaborativa entre los eslabones de la CS. Dichas herramientas son esenciales en los contextos de CS, donde no existe un participante predominante o que posee la autoridad de tomar decisiones por el resto.

A su vez para lograr un objetivo, que beneficie a todos los involucrados en la cadena, es necesario coordinar de manera eficiente los flujos de materiales, dinero, e información. Esta coordinación implica compartir recursos, establecer metas conjuntas, designar roles o responsabilidades, y evaluar el desempeño de la configuración, adoptada por las organizaciones. En un escenario como el descripto, en donde se presentan todas estas características de interdependencia entre las organizaciones, resulta insuficiente la evaluación del desempeño individual de los participantes. Dicha evaluación del desempeño individual, promueve las mejoras al nivel intraorganizacional, pero deja sin considerar las dependencias del nivel interorganizacional de la cadena de suministro.

Dada la complejidad creciente de la CS, la evaluación de su desempeño requiere el análisis de múltiples escenarios de configuración, la comprensión de interacciones entre los participantes, una forma unívoca de medición y comparación del rendimiento entre distintas

cadenas de suministro (independientemente de si son o no de la misma industria), un entendimiento común de la información intercambiada, a nivel de los sistemas de información y tecnologías involucradas. En este contexto la simulación, y más aún, la simulación distribuida, es una herramienta fundamental para asistir en la toma de decisiones colaborativas, las cuales se basan en la evaluación del desempeño.

La SD permite modelar el comportamiento de la CS, como submodelos ejecutados en distintas computadoras, los cuales se integran para componer el modelo de simulación de la cadena de suministro completa. Esta característica de la SD trae aparejado varios beneficios para la toma de decisiones, entre los que se pueden destacar: evaluar el desempeño integral de la CS, por medio de los modelos de simulación específicos de cada participante; compartir información entre los participantes, sin revelar detalles operativos; determinar en qué parte de la CS es necesario mejorar, para alcanzar el objetivo propuesto; establecer un vocabulario común, a efectos de definir las interacciones entre los modelos de simulación.

Si bien la SD posee numerosas ventajas, los dos inconvenientes más importantes que trae aparejado su uso son: la sincronización de los simuladores y la interoperabilidad entre ellos. El primero de ellos (sincronización), hace referencia a la necesidad de que la SD genere exactamente los mismos resultados, que si es ejecutada de forma local. El segundo inconveniente hace referencia a la interpretación que realiza un módulo, de los eventos enviados por otros módulos; de manera tal que los eventos, provoquen los efectos deseados por quien diseña el módulo de simulación que emisor del evento. Este inconveniente, se denomina interoperabilidad entre simuladores. Por un lado, con el fin de sortear el inconveniente de la sincronización (también denominado administración del tiempo), es posible utilizar un esquema conservador u optimista para la administración del tiempo. Por otro lado, para abordar la interoperabilidad entre simuladores, comúnmente, se utiliza el estándar 1516-2010 HLA de la IEEE.

Si bien el estándar HLA aborda la interoperabilidad, el mismo solo permite alcanzar la interoperabilidad sintáctica entre simuladores, por medio del uso del modelo de objetos de federación. A lo largo de esta tesis, se persiguió como meta alcanzar la interoperabilidad semántica; la cual hace referencia a si la interpretación por parte de los simuladores, de los

datos o eventos intercambiados, es válida en la composición de la SD. El objetivo de la interoperabilidad semántica es lograr que, sistemas de información autónomos comprendan el significado de la información generada y compartida, por otros sistemas.

Para alcanzar el nivel de interoperabilidad semántico, es necesario representar el conocimiento sobre los datos a intercambiar entre los simuladores, de una forma estandarizada, que sea capaz de eliminar la ambigüedad. Por este motivo, se decidió utilizar ontologías para expresar el conocimiento común, que utilizan los diversos simuladores de la federación, en la SD basada en HLA. Además, dado que los datos a intercambiar deben estar contenidos en el FOM, y en el contexto de esta tesis, representan relaciones de una CS; fue necesario modelar el conocimiento, tanto del dominio de cadenas de suministro como de federaciones HLA.

Con el objetivo de representar los dominios mencionados en el párrafo anterior, en el capítulo 3 se definió, la red de ontologías denominada *SCFHLA* (Supply Chain Federation HLA), la cual provee un marco conceptual que da soporte, a la interoperabilidad semántica entre simuladores, en una SD basada en HLA de CS. La red de ontologías conceptualiza los dominios de CS y de federaciones HLA. Las razones fundamentales por las cuales se opta por construir una red de ontologías, en lugar de utilizar una única ontología, son las siguientes:

- Permitir el desarrollo modular de cada ontología de dominio.
- Facilitar la gestión y mantenimiento de cada ontología.
- Soportar el desarrollo colaborativo concurrente, de la red de ontologías.
- Mantener cada ontología lo más simple que sea posible, a efectos de facilitar la comprensión de los conceptos involucrados en cada una de las mismas.

De esta manera, el marco conceptual provee:

- Un vocabulario común para definir el modelo de CS.
- Reglas de integridad que definen restricciones entre los conceptos, lo cual permite organizar los mismos, y vincular conceptos de manera valida.
- Reglas de mapeo entre los conceptos definidos, en las distintas ontologías de dominio.

- Reglas de derivación para vincular los conceptos necesarios de SD basada en HLA, para simular el modelo de CS.

Para desarrollar la ontología de CS se decidió incorporar los conceptos de procesos, atributos de rendimiento, y métricas, del modelo de referencia de operaciones de cadena de suministro (SCOR). El modelo SCOR es uno de los modelos con mayor aceptación a nivel mundial, por parte de quienes modelan cadenas de suministro, y además de definir los procesos de negocios clave, presenta atributos de rendimiento y métricas, para realizar una evaluación de la CS, tanto a nivel global como a nivel operativo.

Aunque el modelo SCOR posee algunas falencias, como su limitado nivel de formalidad en la descripción de la CS, la falta de una sintaxis clara, algunas incongruencias debido a la gran dimensión del modelo, y a las debilidades mencionadas; aún así, sigue siendo el modelo más completo disponible, que sintetiza la experiencia de la industria en el funcionamiento de estas redes organizacionales, y por lo tanto, es de suma utilidad para el diseño de una ontología en el dominio de cadenas de suministro.

Por medio del diseño de una ontología basada en el modelo SCOR, se pretendió lograr una especificación con un mayor grado de formalidad, una sintaxis más clara, y sin ambigüedades. De esta forma, se obtuvo una mejora sobre la definición del modelo SCOR, mitigando sus falencias. Con el desarrollo de esta ontología, se pretendió modelar la estructura y comportamiento de una CS, para que luego esta información sea vinculada, a los conceptos asociados a una federación HLA.

Para desarrollar la ontología de federaciones HLA, dado que se trabajó sobre el estándar HLA 1516-2010 de la IEEE, se incorporaron tres grupos de conceptos del estándar, los cuales son:

- Conceptos relacionados con la conformación de una federación, como son el modelo de objetos de federación, modelo de objetos de simulador, modelo de objetos, federación, federado, administrador, entre otros.
- Conceptos relacionados con la gestión de los servicios de la infraestructura de tiempo de ejecución, como son el tipo de datos, la administración del tiempo, el

modo de transporte, objetos e interacciones para la gestión de servicios, entre otros.

- Conceptos relacionados con la conformación de los modelos de objetos necesarios para la SD basada en HLA, como son clases de objeto, clases de interacción, atributos, parámetros, y propiedades.

Además de los conceptos definidos en el estándar HLA, la ontología añade nuevos conceptos al dominio (como el de administrador y objetivo), y contiene reglas de integridad para garantizar la correcta conformación de la federación HLA.

El problema de la interoperabilidad semántica entre simuladores posee distintas soluciones, como pueden ser:

- Una alternativa es que la información requerida por el FOM, sea definida por una única persona u organización, para cada escenario de CS a simular, en la SD basada en HLA. En este caso, no existen problemas de interpretación de los datos, ya que toda la información definida en el FOM, se interpreta como establece dicha persona u organización.
- Otra alternativa es, desarrollar de forma colaborativa y manual el FOM. En este caso, las organizaciones ó personas deben acordar, de forma previa al desarrollo del modelo de objetos, cual va a ser el contenido y el significado, de cada uno de los términos contenidos en el mismo.

Para ambos casos las herramientas disponibles son editores, que garantizan la consistencia sintáctica del FOM, es decir, solo verifican que la información contenida en el mismo, se encuentra en el formato correcto, y con las etiquetas definidas en el lenguaje XML. Por estos motivos, es clara la necesidad que existe de herramientas de software, que den soporte a las tareas de diseño y desarrollo colaborativo, del modelo de objetos de federación en una SD basada en HLA. Con el fin de satisfacer esta necesidad de herramientas, en el capítulo 4 de esta tesis, se presentó el diseño de una plataforma basada en la red de ontologías *SCFHLA*, para la construcción del modelo de objetos de federaciones HLA. La finalidad de esta plataforma, es brindar una herramienta de software que facilite y automatice parcialmente, las tareas relacionadas con la construcción colaborativa del FOM.

La plataforma diseñada cumple con los requerimientos necesarios, para la construcción del modelo de objetos de federación, en forma colaborativa entre los miembros de la cadena, a partir del vocabulario definido en la red de ontologías. A efectos de construir el FOM, la plataforma utiliza la red de ontologías y el algoritmo de transformación. La red de ontologías permite modelar toda la información necesaria, relativa al modelo de interoperabilidad para la SD basada en HLA de una CS; mientras que el algoritmo de transformación, utiliza como entradas un archivo de configuración y la información definida en la red de ontologías, para construir el modelo de objetos de federación.

Por un lado, el archivo de configuración contiene la información de gestión, relacionada con los servicios proporcionados por una implementación específica, de la infraestructura de tiempo de ejecución, la cual en el caso de esta tesis fue *PoRTIco*. Por otro lado, la información definida en la red de ontologías, está relacionada con los objetos e interacciones, definidos para la federación HLA. Esta información se transforma, por medio del algoritmo, en etiquetas XML con atributos y datos.

De este modo, como beneficios del uso de la plataforma, se pueden destacar:

- La disminución del conocimiento específico requerido, para la construcción del modelado de objetos de federación, en cuanto al estándar HLA.
- La disponibilidad de un mecanismo, para verificar la completitud y consistencia del modelo de objetos. Por medio de las reglas definidas, es posible verificar que el FOM posea todas las etiquetas requeridas y, además, que la información contenida en él sea la misma, que la defindia en el modelo de cadena de suministro.
- La reducción del tiempo y esfuerzo necesarios, para el diseño y construcción del modelo de objetos de federación.
- La reducción de la intervención humana necesaria, para la construcción del modelo de objetos de federación.
- La simplicidad en el uso. La plataforma presenta conceptos familiares, para los analistas de negocios o quienes modelan una cadena de suministro, por lo que de esta forma, a ambos grupos, les resulta transparente utilizar la herramienta de modelado, para obtener el modelo de objetos de federación.

Cabe recordar que el objetivo de la interoperabilidad semántica, es lograr que sistemas de información autónomos, comprendan el significado de la información generada y compartida, por otros sistemas. La plataforma diseñada en esta tesis, pretende ser un aporte para que los sistemas autónomos, logren dicha comprensión, de forma semi automática.

Como medio de validación del enfoque propuesto, en el marco de esta tesis se planteo en el capítulo 5, una prueba de concepto. En dicha prueba, se describe como utilizar los componentes de la red de ontologías SCFHLA y el algoritmo de transformación, a efectos de construir el modelo de objetos de federación. Para realizar esta tarea, se presentó la construcción del modelo de interoperabilidad, en la red de ontologías. Luego, se utilizó el algoritmo de transformación para convertir el modelo de interoperabilidad, en un documento con formato XML, el cual pueda ser utilizado como FOM en la SD basada en HLA, del escenario modelado. Estas son las actividades requeridas, para construir el modelo de objetos de federación, a partir de la descripción del escenario de CS propuesto como caso de estudio.

6.2 Trabajos Futuros

En base al desarrollo realizado durante esta tesis, a las potencialidades y debilidades de la plataforma, y a los objetivos planteados, surgieron líneas de trabajo a estudiar en un futuro cercano, para mejorar o extender la plataforma.

Como primer objetivo, a corto plazo se plantea diseñar e implementar la herramienta para la construcción del FOM, de la plataforma. Este es el primer objetivo, dado que en el marco de esta tesis, solo se ha especificado su funcionalidad. El desarrollo de esta herramienta permitiría que los participantes de una CS, la utilicen de forma colaborativa para el desarrollo del FOM, así como también, facilitaría la integración de los componentes desarrollados en esta tesis.

Como un trabajo futuro, se plantea construir un modelo de objetos de federación, por medio del enfoque propuesto en esta tesis, y desarrollar la SD basada en HLA de CS, que haga uso del FOM construido. Esta tarea implica el desarrollo de modelos de simulación, que respeten el contrato establecido en el FOM, además de la puesta en funcionamiento de una

infraestructura de tiempo de ejecución, que permita la construcción de la federación HLA. Para llevar a cabo esta tarea, se utilizará el formalismo DEVS para desarrollar los modelos de simulación de los federados; y además, se utilizará la infraestructura de tiempo de ejecución *PoRTIco*, para construir la federación HLA. Una vez construidos los federados y la federación, se deben realizar pruebas, a efectos de depurar errores de implementación y ajustar los parámetros de la simulación.

Según el estándar 1516-2010 HLA, en una SD cada federación debe tener asociado un modelo de objetos de federación, y cada federado debe tener asociado un modelo de objetos de simulador (SOM). A su vez, cada objeto o interacción definido en el FOM debe encontrarse definido también, en al menos uno de los SOM de los federados, que participan de la federación con dicho FOM. Que los federados y las federaciones sigan estas reglas, las cuales se encuentran definidas en el estándar HLA, asegura una interacción correcta entre los simuladores integrantes de una federación. Si bien estos lineamientos del estándar deben cumplirse, toda la responsabilidad de hacer cumplir estas reglas, recae sobre quienes desarrollan los componentes (federados) y componen la federación HLA, de la SD.

Hasta el momento de redacción de esta tesis, el autor no conoce de la existencia de herramientas para verificar que el contenido del FOM, se encuentre definido en los SOM de los federados, que participan de la federación. Debido a esto, resulta de interés para asegurar un correcto funcionamiento de la SD basada en HLA, desarrollar una herramienta que compruebe, la composición entre: el FOM de la federación a simular, y los SOM de cada uno de los federados participantes de la federación. Además, esta herramienta puede corroborar otro tipo de reglas, como por ejemplo: vincular a una federación recién creada un FOM; vincular a un federado recién creado un SOM; si un federado se une a una federación y no posee SOM, entonces asociarle a ese federado, el FOM de la federación como su SOM; si una federación no posee FOM, y se le une un federado que posee SOM, entonces asociarle a esa federación, el SOM del federado como su FOM.

Una de las limitaciones del trabajo presentado, es que el modelo de objetos de federación construido, a partir de los componentes de la plataforma, es solo para SD basada

en HLA del dominio de CS. Esta limitación se puede resolver, dados los beneficios de haber trabajado con una red de ontologías, en lugar de con una única ontología.

Como trabajo futuro se prevé reemplazar la ontología de CS, por una ontología de otro dominio. Luego, se deberían añadir reglas de mapeo del nuevo dominio, al dominio de federaciones HLA. De este modo, se podría trabajar de la misma manera que con la ontología de CS, para construir el modelo de objetos de federación, en otros dominios. Así, la red de ontologías podría, eventualmente, ir cubriendo distintos dominios, y brindar los mismos beneficios que se describieron en esta tesis, para la construcción de un modelo de objetos de federación del dominio de CS. Con el paso del tiempo, distintos dominios podrían ser alcanzados, y la plataforma dejaría de ser específica, de un dominio en particular. Esta característica probablemente permita probar, aún más, la aplicabilidad del enfoque de trabajo, propuesto a lo largo de esta tesis.

La tarea de construir el FOM es una de las más desafiantes, a la hora de desarrollar una SD basada en HLA debido a que este documento, es un requisito fundamental para ejecutar la simulación. Sin embargo, además de contar con este documento, es necesario que cada federado posea los modelos de simulación, con la lógica interna de los procesos que participan en la CS. Por este motivo encontrar un modo de facilitar, la construcción de los modelos de simulación necesarios, resulta de suma importancia; ya que disminuye el trabajo requerido, para construir una SD basada en HLA, de cadenas de suministro. Para ello como trabajo futuro se pretende, vincular a los procesos definidos en la ontología de CS, a una especificación de procesos de negocios para, a partir de la misma, derivar los modelos de simulación correspondientes.

Esta línea de investigación, pretende especificar con el lenguaje de modelado de procesos de negocio BPM (por sus siglas en ingles *–Business Process Modeling*-), los procesos internos de cada federado, que dan soporte a los procesos interorganizacionales especificados, en la ontología de CS. A partir del diagrama de los procesos internos, definido con la notación de BPM, junto con la información de entrada y salida, definida en los procesos interorganizacionales; se definirán reglas para transformar dicho diagrama, en los modelos de simulación necesarios, para construir la SD basada en HLA de la CS.

Referencias

Ali, Nauman Bin, Kai Petersen, y Breno Bernard Nicolau de França. 2015. "Evaluation of Simulation-Assisted Value Stream Mapping for Software Product Development: Two Industrial Cases". *Information and Software Technology* 68 (diciembre): 45–61. https://doi.org/10.1016/j.infsof.2015.08.005.

Allocca, Carlo, Mathieu D'Aquin, y Enrico Motta. 2009. "DOOR: Towards a Formalization of Ontology Relations". En , 1:13–20. Funchal - Madeira, Portugal. https://doi.org/10.5220/0002276400130020.

Álvarez, Esther, Fernando Díaz, y Miguel A. Larrinaga. 2011. "Panorama de la gestión de la cadena de suministro: retos, colaboración y gestión de excepciones". *Boletín de Estudios Económicos* 66 (204): 531.

Anagnostou, Anastasia, y Simon J.E. Taylor. 2017. "A Distributed Simulation Methodological Framework for OR/MS Applications". *Simulation Modelling Practice and Theory* 70 (enero): 101–19. https://doi.org/10.1016/j.simpat.2016.10.007.

Archer, P., S. Goedertier, y N. Loutas. 2012. "D7.1.3 – Study on persistent URIs, with identification of best practices and recommendations on the topic for the MSs and the EC." https://joinup.ec.europa.eu/sites/default/files/document/2013-02/D7.1.3%20-%20Study%20on%20persistent%20URIs.pdf.

Arrazola, Jaime. 2007. "Towards a New Model: the Globally Integrated Enterprise". *Universia Business Review* 14 (mayo): 108–18.

Aversano, Lerina, Carmine Grasso, y Maria Tortorella. 2016. "Managing the Alignment between Business Processes and Software Systems". *Information and Software Technology* 72 (abril): 171–88. https://doi.org/10.1016/j.infsof.2015.12.009.

Barnett, Michael W., y Charles J. Miller. 2000. "Analysis of the virtual enterprise using distributed supply chain modeling and simulation: an application of e-SCOR". En *Proceedings of the 32nd conference on Winter simulation*, editado por J. A. Joines, R. R. Barton, K. Kang, y P. A. Fishwick, 352–355. Society for Computer Simulation International. http://dl.acm.org/citation.cfm?id=510435.

Bell, David, Sergio de Cesare, Mark Lycett, Navonil Mustafee, y Simon J. Taylor. 2007. "Semantic Web Service Architecture for Simulation Model Reuse". En *11th IEEE Symposium on Distributed Simulation and Real Time Applications*, 129–36. IEEE. https://doi.org/10.1109/DS-RT.2007.38.

Birasnav, M., y Joshua Bienstock. 2019. "Supply Chain Integration, Advanced Manufacturing Technology, and Strategic Leadership: An Empirical Study". *Computers & Industrial Engineering*, enero. https://doi.org/10.1016/j.cie.2019.01.021.

Bocciarelli, Paolo, Andrea D'Ambrogio, Emiliano Paglia, y Andrea Giglio. 2017. "An HLA-Based BPMN Extension for the Specification of Business Process Collaborations". En , 1–8. IEEE. https://doi.org/10.1109/DISTRA.2017.8167668.

Breitman, Karin Koogan, Marco Antonio Casanova, y Walter Truszkowski, eds. 2007. "Methods for Ontology Development". En *Semantic Web: Concepts, Technologies and Applications*, 155–73. NASA Monographs in Systems and Software Engineering. London: Springer London. https://doi.org/10.1007/978-1-84628-710-7_8.

Brusa, Graciela, Ma Laura Caliusco, y Omar Chiotti. 2006. "Building Ontology in Public Administration: A Case Study". En , 226:16–30. CEUR-Workshop Proceedings. http://ceur-ws.org/Vol-226/paper02.pdf.

Camarinha-Matos, Luis M., Hamideh Afsarmanesh, Nathalie Galeano, y Arturo Molina. 2009. "Collaborative Networked Organizations – Concepts and Practice in Manufacturing Enterprises". *Computers & Industrial Engineering*, Collaborative e-Work Networks in Industrial Engineering, 57 (1): 46–60. https://doi.org/10.1016/j.cie.2008.11.024.

Chan, Felix T.S., y T. Zhang. 2011. "The Impact of Collaborative Transportation Management on Supply Chain Performance: A Simulation Approach". *Expert Systems with Applications* 38 (3): 2319–29. https://doi.org/10.1016/j.eswa.2010.08.020.

Chandra, Charu, y Jānis Grabis. 2007. *Supply Chain Configuration: Concepts, Solutions, and Applications*. Springer US. //www.springer.com/gp/book/9781441937780.

Chandra, Charu, y Armen Tumanyan. 2007. "Ontology-Driven Information System for Supply Chain Management". En *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, editado por Raj Sharman, Rajiv Kishore, y Ram Ramesh, 697–726. Integrated Series in Information Systems. Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-37022-4_25.

Chavez, Jorge H. 2012. *Supply Chain Management (Gestión de la cadena de suministro)*. RIL Editores.

Cho, Dong Won, Young Hae Lee, Sung Hwa Ahn, y Min Kyu Hwang. 2012. "A framework for measuring the performance of service supply chain management". *Computers & Industrial Engineering*, Soft Computing for Management Systems, 62 (3): 801–18. https://doi.org/10.1016/j.cie.2011.11.014.

Chung, Walter W. C., Anthony Y. K. Yam, y Michael F. S. Chan. 2004. "Networked Enterprise: A New Business Model for Global Sourcing". *International Journal of Production Economics*, Supply Chain Management for the 21st Century Organizational Competitiveness, 87 (3): 267–80. https://doi.org/10.1016/S0925-5273(03)00222-6.

Cigolini, R., M. Pero, T. Rossi, y A. Sianesi. 2014. "Linking supply chain configuration to supply chain perfrmance: A discrete event simulation model". *Simulation Modelling Practice and Theory* 40: 1–11. https://doi.org/10.1016/j.simpat.2013.08.002.

Cope, Dayana, Mohamed Sam Fayez, Mansooreh Mollaghasemi, y Assem Kaylani. 2007. "Supply chain simulation modeling made easy: an innovative approach". En *Simulation Conference, 2007 Winter*, 1887–1896. IEEE. http://ieeexplore.ieee.org/abstract/document/4419816/.

D'Angelo, Gabriele. 2011. "Parallel and Distributed Simulation from Many Cores to the Public Cloud". En *High Performance Computing and Simulation (HPCS), 2011*

International Conference on, 14–23. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5999802.

D'Angelo, Gabriele, y Moreno Marzolla. 2014. "New trends in parallel and distributed simulation: From many-cores to Cloud Computing". *Simulation Modelling Practice and Theory* 49 (diciembre): 320–35. https://doi.org/10.1016/j.simpat.2014.06.007.

Delligatti, Lenny. 2013. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. 01 ed. Upper Saddle River, NJ: Addison Wesley.

Díaz, A., R. Motz, E. Rohrer, y L. Tansini. 2011. "An Ontology Network for Educational Recommender Systems". En *Educational Recommender Systems and Technologies: Practices and Challenges*, 67–93.

Diaz, Alicia, Regina Motz, y Edelweis Rohrer. 2011. "Making Ontology Relationships Explicit in a Ontology Network". En *Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management*, 13. Santiago, Chile.

DOD. 1995. "Modeling And Simulation (M&S) Master Plan." DoD-5000.59-P. OFFICE OF THE UNDER SECRETARY OF DEFENSE (ACQUISITION AND TECHNOLOGY) WASHINGTON DC. https://apps.dtic.mil/docs/citations/ADA301787.

Dragoicea, Monica, Laurentiu Bucur, Wei-Tek Tsai, y Hessam Sarjoughian. 2012. "Integrating HLA and Service-Oriented Architecture in a Simulation Framework". En *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 861–66. IEEE. https://doi.org/10.1109/CCGrid.2012.76.

Dyer, Jeffrey H. 2000. *Collaborative Advantage: Winning Through Extended Enterprise Supplier Networks*. Edición: New. New York: Oxford University Press.

Flynn, Barbara B., Baofeng Huo, y Xiande Zhao. 2010. "The impact of supply chain integration on performance: A contingency and configuration approach". *Journal of Operations Management* 28 (1): 58–71. https://doi.org/10.1016/j.jom.2009.06.001.

Friesen, Andreas, Wolfgang Theilmann, Markus Heller, Jens Lemcke, y Christof Momm. 2012. "On Some Challenges in Business Systems Management and Engineering for the Networked Enterprise of the Future". En *Business System Management and Engineering*, editado por Claudio A. Ardagna, Ernesto Damiani, Leszek A. Maciaszek, Michele Missikoff, y Michael Parkin, 7350:1–15. Berlin, Heidelberg: Springer Berlin Heidelberg. http://link.springer.com/10.1007/978-3-642-32439-0_1.

Fujimoto, Richard. 2015. "Parallel and distributed simulation". En *Proceedings of the 2015 Winter Simulation Conference*, 45–59. IEEE Press. http://dl.acm.org/citation.cfm?id=2888624.

Fujimoto, Richard M. 1990. "Parallel Discrete Event Simulation". *Communications of the ACM* 33 (10): 30–53. https://doi.org/10.1145/84537.84545.

———. 2016. "Research Challenges in Parallel and Distributed Simulation". *ACM Transactions on Modeling and Computer Simulation* 26 (4): 1–29. https://doi.org/10.1145/2866577.

Fujimoto, Richard M., Asad Waqar Malik, y A. Park. 2010. "Parallel and distributed simulation in the cloud". *SCS M&S Magazine* 3: 1–10.

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*. New York: John Wiley & Sons.

Furian, N., M. O'Sullivan, C. Walker, S. Vössner, y D. Neubacher. 2015. "A Conceptual Modeling Framework for Discrete Event Simulation Using Hierarchical Control

Structures". *Simulation Modelling Practice and Theory* 56 (agosto): 82–96. https://doi.org/10.1016/j.simpat.2015.04.004.

Gogi, Anastasia, Antuela A. Tako, y Stewart Robinson. 2016. "An experimental investigation into the role of simulation models in generating insights". *European Journal of Operational Research* 249 (3): 931–44. https://doi.org/10.1016/j.ejor.2015.09.042.

Gómez-Pérez, Asunción. 2004. "Ontology Evaluation". En *Handbook on Ontologies*, editado por Steffen Staab y Rudi Studer, 251–73. International Handbooks on Information Systems. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-24750-0_13.

Gómez-Pérez, Asunción, Mariano Fernández-López, y Oscar Corcho. 2004. *Ontological Engineering*. Advanced Information and Knowledge Processing. London: Springer-Verlag. http://link.springer.com/10.1007/b97353.

———. 2010. *Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition*. Springer.

Grüninger, Michael, y Mark S. Fox. 1995. "Methodology for the Design and Evaluation of Ontologies". En . Montreal.

Guarino, Nicola, Daniel Oberle, y Steffen Staab. 2009. "What Is an Ontology?" En *Handbook on Ontologies*, editado por Steffen Staab y Rudi Studer, 1–17. International Handbooks on Information Systems. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-92673-3_0.

Gutiérrez, M. Milagros, y Horacio P. Leone. 2012. "DE2M: An Environment for Developing Distributed and Executable Enterprise Models". *Advances in Engineering Software* 47 (1): 80–103. https://doi.org/10.1016/j.advengsoft.2011.12.002.

Hearnshaw, Edward J.S., y Mark M.J. Wilson. 2013. "A Complex Network Approach to Supply Chain Network Theory". *International Journal of Operations & Production Management* 33 (4): 442–69. https://doi.org/10.1108/01443571311307343.

Heath, Tom, y Christian Bizer. 2011. "Linked Data: Evolving the Web into a Global Data Space". *Synthesis Lectures on the Semantic Web: Theory and Technology* 1 (1): 1–136. https://doi.org/10.2200/S00334ED1V01Y201102WBE001.

Hennies, Til, Tobias Reggelin, Juri Tolujew, y Pierre-Alain Piccut. 2014. "Mesoscopic supply chain simulation". *Journal of Computational Science* 5 (3): 463–70. https://doi.org/10.1016/j.jocs.2013.08.004.

Hofmann, M., Julia Palii, y Goran Mihelcic. 2011. "Epistemic and normative aspects of ontologies in modelling and simulation". *Journal of Simulation* 5 (3): 135–146.

Hogan, Aidan, Andreas Harth, Alexandre Passant, Stefan Decker, y Axel Polleres. 2010. "Weaving the Pedantic Web". *3rd International Workshop on Linked Data on the Web (LDOW2010) in Conjunction with 19th International World Wide Web Conference*, abril. http://aran.library.nuigalway.ie/xmlui/handle/10379/1093.

IEEE. 2010a. *IEEE Standard for Modeling and Simulation (M & S) High Level Architecture (HLA) Framework and Rules*. Editado por Institute of Electrical and Electronics Engineers. New York: Institute of Electrical and Electronics Engineers. http://ieeexplore.ieee.org/servlet/opac?punumber=5553438.

———. 2010b. *IEEE Standard for Modeling and Simulation (M & S) High Level Architecture (HLA) Object Model Template (OMT) Specification*. Editado por Institute of Electrical and Electronics Engineers. New York: Institute of Electrical and Electronics Engineers. http://ieeexplore.ieee.org/servlet/opac?punumber=5557729.

Ingalls, Ricki G. 2014. “Introduction to supply chain simulation”. En *Proceedings of the 2014 Winter Simulation Conference*, 36–50. IEEE Press. http://dl.acm.org/citation.cfm?id=2693860.

Jain, Ajitesh, Richard Fujimoto, John Crittenden, Liu Mengmeng, Jongchan Kim, y Zhongming Lu. 2015. “Towards automating the development of federated distributed simulations for modeling sustainable urban infrastructures”. En *Proceedings of the 2015 Winter Simulation Conference*, 2668–2679. IEEE. http://ieeexplore.ieee.org/abstract/document/7408374/.

“JDOM”. 2017. 2017. http://www.jdom.org/index.html.

Jian, Wang, Huang Yang, y Wang ZiYang. 2017. “Research on Modeling and Simulation of Distributed Supply Chain Based on HAS”. En *Intelligent Robotics and Applications*, editado por YongAn Huang, Hao Wu, Honghai Liu, y Zhouping Yin, 10464:401–12. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-65298-6_37.

Karhela, Tommi, Antti Villberg, y Hannu Niemistö. 2012. “OPEN ONTOLOGY-BASED INTEGRATION PLATFORM FOR MODELING AND SIMULATION IN ENGINEERING”. *International Journal of Modeling, Simulation, and Scientific Computing* 03 (02): 1250004. https://doi.org/10.1142/S1793962312500043.

Knublauch, Holger, Matthew Horridge, Mark A. Musen, Alan L. Rector, Robert Stevens, Nick Drummond, Phillip W. Lord, Natalya Fridman Noy, Julian Seidenberg, y Hai Wang. 2005. “The Protege OWL Experience”. En *Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions*, editado por Bernardo Cuenca Grau, Ian Horrocks, Bijan Parsia, y Peter F. Patel-Schneider. Vol. 188. CEUR Workshop Proceedings. Galway, Ireland: CEUR-WS.org. http://ceur-ws.org/Vol-188/sub14.pdf.

Liu, Xiaocheng, Qiang He, Xiaogang Qiu, Bin Chen, y Kedi Huang. 2012. “Cloud-Based Computer Simulation: Towards Planting Existing Simulation Software into the Cloud”. *Simulation Modelling Practice and Theory* 26 (agosto): 135–50. https://doi.org/10.1016/j.simpat.2012.05.001.

Long, Qingqi. 2014. “An agent-based distributed computational experiment framework for virtual supply chain network development”. *Expert Systems with Applications* 41 (9): 4094–4112. https://doi.org/10.1016/j.eswa.2014.01.001.

Long, Qingqi, y Wenyu Zhang. 2014. “An integrated framework for agent based inventory–production–transportation modeling and distributed simulation of supply chains”. *Information Sciences* 277 (septiembre): 567–81. https://doi.org/10.1016/j.ins.2014.02.147.

Lustig, I, B Dietrich, C Johnson, y C Dziekan. 2010. “The Analytics Journey”. Analytics Magazine. 3 de noviembre de 2010. http://analytics-magazine.org/the-analytics-journey/.

Lutz, Robert. 2014. “SISO Standards in Action: A DSEEP Based Process Framework for UAS Airspace Integration Analysis”. En *Fall Simulation Interoperability Workshop*.

Maedche, Alexander. 2002. *Ontology Learning for the Semantic Web*. The Springer International Series in Engineering and Computer Science. Springer US. //www.springer.com/la/book/9780792376569.

McIntyre, Heather M., Ebb Smith, y Mary Goode. 2013. “United Kingdom Mission Training Through Distributed Simulation”. *Military Psychology* 25 (3): 280–93. https://doi.org/10.1037/h0094969.

Mejía Argueta, Christopher, y Catalina Higuita Salazar. 2015. "Cost to serve as a strategic decision variable in the design of strategies as regards emerging marketing channels". *Estudios Gerenciales* 31 (134): 50–61.

Mezgár, István, y Ursula Rauschecker. 2014. "The Challenge of Networked Enterprises for Cloud Computing Interoperability". *Computers in Industry* 65 (4): 657–74. https://doi.org/10.1016/j.compind.2014.01.017.

Mittal, Saurabh, Saikou Diallo, y Andreas Tolk. 2018. *Emergent Behavior in Complex Systems Engineering: A Modeling and Simulation Approach*. John Wiley & Sons.

Mojtahed, Vahid, Mika Cohen, Thomas Jansson, Martin Eklöf, Christian M\a artenson, y Jelena Zdravkovic. 2011. "Realising the first prototype of the semantic interoperability logical framework". En *NATO IST, Workshop (IST-097/RWS-016) November*. http://www.foi.se/global/our_knowledge/decision_support_system_and_information_fusion/foi-s--3829--se.pdf.

Montoya-Torres, Jairo R., y Diego Ortiz Vargas. 2011. "Análisis del concepto de colaboración en la cadena de suministro: Una revisión de la literatura científica". En *Ninth LACCEI Latin American and Caribbean Conference*. Medellín, Colombia. http://laccei.org/LACCEI2011-Medellin/RefereedPapers/LO075_Montoya.pdf.

Moon, Jong-Hyuk, Young-Hae Lee, y Dong-Won Cho. 2012. "Object-Oriented Simulation Modeling for Service Supply Chain". *Journal of the Korea Society for Simulation* 21 (1): 55–68. https://doi.org/10.9709/JKSS.2012.21.1.055.

Mustafee, Navonil, Korina Katsaliaki, y Simon JE Taylor. 2014. "A review of literature in distributed supply chain simulation". En *Simulation Conference (WSC), 2014 Winter*, 2872–2883. IEEE.

Negri, Elisa, Luca Fumagalli, Marco Garetti, y Letizia Tanca. 2016. "Requirements and Languages for the Semantic Representation of Manufacturing Systems". *Computers in Industry* 81 (septiembre): 55–66. https://doi.org/10.1016/j.compind.2015.10.009.

Nouman, Athar, Anastasia Anagnostou, y Simon J.E. Taylor. 2013. "Developing a Distributed Agent-Based and DES Simulation Using poRTIco and Repast". En , 97–104. Delft, Netherlands: IEEE. https://doi.org/10.1109/DS-RT.2013.18.

Noy, Natalya F, y Deborah L McGuinness. 2001. "Ontology Development 101: A Guide to Creating Your First Ontology". Reporte Tecnico KSL-01-05. Stanford, CA: Stanford Knowledge Systems Laboratory.

O'Connor, Martin, Holger Knublauch, S. W. Tu, y M. A. Musen. 2005. "Writing rules for the semantic web using SWRL and Jess". *Protégé With Rules WS, Madrid*. https://www.researchgate.net/profile/Martin_Oconnor3/publication/239616230_Writing_Rules_for_the_Semantic_Web_using_SWRL_and_Jess/links/53d08ce10cf25dc05cfe4861.pdf.

Oliveira, Josenildo Brito, Renato Silva Lima, y José Arnaldo Barra Montevechi. 2016. "Perspectives and Relationships in Supply Chain Simulation: A Systematic Literature Review". *Simulation Modelling Practice and Theory* 62 (marzo): 166–91. https://doi.org/10.1016/j.simpat.2016.02.001.

OMG. 2013. "OMG UML Specification". http://www.omg.org/spec/UML/2.5.

öZdikis, Özer, Umut Durak, y Halit OguztüZüN. 2010. "Tool Support for Transformation from an OWL Ontology to an HLA Object Model". En *3rd International ICST*

Conference On Simulation Tools and Techniques. Malaga, España. https://doi.org/10.4108/ICST.SIMUTOOLS2010.8678.

Petty, Mikel D., Eric W. Weisel, y R. Mielke. 2004. "Overview of a Theory of Composability". En . Old Dominion University.

———. 2005. "Composability Theory Overview and Update". En , 431–437. San Diego, California, USA.

Piedra, Nelson, Chicaiza, Janneth, Quichimbo, Pricila, Saquicela, Victor, Cadme, Elizabeth, López, Jorge, Espinoza, Mauricio, y Tovar, Edmundo. 2015. "Marco de Trabajo para la Integración de Recursos Digitales Basado en un Enfoque de Web Semántica". *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, nº E3 (marzo): 55–70. https://doi.org/10.17013/risti.e3.55-70.

Piedra, Nelson, y Juan Pablo Suárez. 2018. "Hacia la Interoperabilidad Semántica para el Manejo Inteligente y Sostenible de Territorios de Alta Biodiversidad usando SmartLand-LD". *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação* 26 (marzo): 104–21. https://doi.org/10.17013/risti.26.104-121.

"Pitch Technologies". 2017. 2017. http://www.pitchtechnologies.com/.

Pokorny, Tim, Michael Fraser, y Lance Burns. 2016. "Portico". 2016. http://www.porticoproject.org/comingsoon/.

Powers, Matthew J., Susan M. Sanchez, y Thomas W. Lucas. 2012. "The Exponential Expansion of Simulation in Research". En *Proceedings of the Winter Simulation Conference*, 138:1–138:12. WSC '12. Berlin, Germany: Winter Simulation Conference. http://dl.acm.org/citation.cfm?id=2429759.2429944.

Proudlove, N. C., S. Bisogno, B. S. S. Onggo, A. Calabrese, y N. Levialdi Ghiron. 2017. "Towards fully-facilitated discrete event simulation modelling: Addressing the model coding stage". *European Journal of Operational Research* 263 (2): 583–95. https://doi.org/10.1016/j.ejor.2017.06.002.

Rahimi, Fatemeh, Charles Møller, y Lars Hvam. 2016. "Business Process Management and IT Management: The Missing Integration". *International Journal of Information Management* 36 (1): 142–54. https://doi.org/10.1016/j.ijinfomgt.2015.10.004.

Rainey, Larry B., y Andreas Tolk. 2015. *Modeling and Simulation Support for System of Systems Engineering Applications*. John Wiley & Sons.

Ramanathan, Usha. 2014. "Performance of supply chain collaboration – A simulation study". *Expert Systems with Applications*, 21st Century Logistics and Supply Chain Management, 41 (1): 210–20. https://doi.org/10.1016/j.eswa.2013.07.022.

Raweewan, Morrakot, y William G. Ferrell. 2018. "Information sharing in supply chain collaboration". *Computers & Industrial Engineering* 126 (diciembre): 269–81. https://doi.org/10.1016/j.cie.2018.09.042.

Rector, Alan, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, y Chris Wroe. 2004. "OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns". En *Engineering Knowledge in the Age of the Semantic Web*, editado por Enrico Motta, Nigel R. Shadbolt, Arthur Stutt, y Nick Gibbins, 63–81. Lecture Notes in Computer Science 3257. Springer Berlin Heidelberg. http://link.springer.com/chapter/10.1007/978-3-540-30202-5_5.

Robinson, Stewart. 2002. “Modes of simulation practice: approaches to business and military simulation”. *Simulation Modelling Practice and Theory* 10 (8): 513–23. https://doi.org/10.1016/S1569-190X(02)00117-X.

Roussey, Catherine, Francois Pinet, Myoung Ah Kang, y Oscar Corcho. 2011. “An Introduction to Ontologies and Ontology Engineering”. En *Ontologies in Urban Development Projects*, editado por Gilles Falquet, Claudine Métral, Jacques Teller, y Christopher Tweed, 9–38. Advanced Information and Knowledge Processing. London: Springer London. https://doi.org/10.1007/978-0-85729-724-2_2.

Rycerz, Katarzyna, Marian Bubak, Eryk Ciepiela, Daniel Harężlak, Tomasz Gubała, Jan Meizner, Maciej Pawlik, y Bartosz Wilk. 2015. “Composing, Execution and Sharing of Multiscale Applications”. *Future Generation Computer Systems* 53 (diciembre): 77–87. https://doi.org/10.1016/j.future.2015.06.002.

Sabouhi, Fatemeh, Mir Saman Pishvaee, y Mohammad Saeed Jabalameli. 2018. “Resilient supply chain design under operational and disruption risks considering quantity discount: A case study of pharmaceutical supply chain”. *Computers & Industrial Engineering* 126 (diciembre): 657–72. https://doi.org/10.1016/j.cie.2018.10.001.

Sekiguchi, Atsuji, Kuniaki Shimada, Yuji Wada, Akio Ooba, Ryouji Yoshimi, y Akiko Matsumoto. 2012. “CONFIGURATION MANAGEMENT TECHNOLOGY FOR LARGE-SCALE SIMULATIONS”. *SCS M&S Magazine* 3 (julio): 12–22.

Silver, Gregory A., OA-H. Hassan, y John A. Miller. 2007. “From domain ontologies to modeling ontologies to executable simulation models”. En *Proceedings of the 2007 Winter Simulation Conference*, 1108–1117. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4419710.

Simchi-Levi, David, Philip Kaminsky, y Edith Simchi-Levi. 1999. *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies*. First Printing edition. Boston: McGraw-Hill.

Snively, Keith, Richard Leslie, y Chris Gaughan. 2013. “Runtime execution management of distributed simulations”. En *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, 2878–2888. IEEE Press. http://dl.acm.org/citation.cfm?id=2676339.

Staab, Steffen, y Rudi Studer, eds. 2009. *Handbook on Ontologies*. Berlin, Heidelberg: Springer Berlin Heidelberg. http://link.springer.com/10.1007/978-3-540-92673-3.

Stanford, University. 2016. “Protégé”. 2016. https://protege.stanford.edu/.

Suárez-Figueroa, Mari Carmen, Asunción Gómez-Pérez, y Mariano Fernández-López. 2012. “The NeOn Methodology for Ontology Engineering”. En *Ontology Engineering in a Networked World*, editado por Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, y Aldo Gangemi, 9–34. Springer Berlin Heidelberg. http://link.springer.com/chapter/10.1007/978-3-642-24794-1_2.

Sun, Hongbo, Wenhui Fan, Weiming Shen, y Tianyuan Xiao. 2012. “Ontology-Based Interoperation Model of Collaborative Product Development”. *Journal of Network and Computer Applications* 35 (1): 132–44. https://doi.org/10.1016/j.jnca.2011.02.012.

Supply Chain Council. 2012. *SCOR*. Supply Chain Council.

———. 2017. “SCOR 12.0 | New SCOR Model for the Modern Supply Chain | APICS”. 2017. http://www.apics.org/apics-for-business/frameworks/scor12.

Tako, Antuela A., y Kathy Kotiadis. 2015. "PartiSim: A multi-methodology framework to support facilitated simulation modelling in healthcare". *European Journal of Operational Research* 244 (2): 555–64. https://doi.org/10.1016/j.ejor.2015.01.046.

Taylor, Simon J. E. 2019. "Distributed simulation: state-of-the-art and potential for operational research". *European Journal of Operational Research* 273 (1): 1–19. https://doi.org/10.1016/j.ejor.2018.04.032.

Taylor, Simon J. E., Azam Khan, Katherine L. Morse, Andreas Tolk, Levent Yilmaz, y Justyna Zander. 2013. "Grand Challenges on the Theory of Modeling and Simulation". En *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium*, 34:1–34:8. DEVS 13. San Diego, CA, USA: Society for Computer Simulation International. http://dl.acm.org/citation.cfm?id=2499634.2499668.

Taylor, Simon J E, Azam Khan, Katherine L Morse, Andreas Tolk, Levent Yilmaz, Justyna Zander, y Pieter J Mosterman. 2015. "Grand Challenges for Modeling and Simulation: Simulation Everywhere—from Cyberinfrastructure to Clouds to Citizens". *SIMULATION* 91 (7): 648–65. https://doi.org/10.1177/0037549715590594.

Taylor, Simon J. E., Stephen J. Turner, Steffen Strassburger, y Navonil Mustafee. 2012. "Bridging the Gap: A Standards-based Approach to OR/MS Distributed Simulation". *ACM Trans. Model. Comput. Simul.* 22 (4): 18:1–18:23. https://doi.org/10.1145/2379810.2379811.

Taylor, Simon J.E., David Bell, Navonil Mustafee, Sergio de Cesare, Mark Lycett, y Paul A. Fishwick. 2010. "Semantic Web Services for Simulation Component Reuse and Interoperability: An Ontology Approach". En *Organizational Advancements through Enterprise Information Systems: Emerging Applications and Developments*, editado por Angappa Gunasekaran y Timothy Shea, 336–52. IGI Global. http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-968-7.

Taylor, Simon JE, Sally Brailsford, Stephen E. Chick, Pierre L'Ecuyer, Charles M. Macal, y Barry L. Nelson. 2013. "Modeling and Simulation Grand Challenges: An OR/MS Perspective". En *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, 1269–1282. IEEE Press. http://dl.acm.org/citation.cfm?id=2676144.

Tolaba, Ana Carolina, Maria Laura Caliusco, y Maria Rosa Galli. 2014. "Representación del Conocimiento de la Información Geográfica siguiendo un Enfoque basado en Ontologías". *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação* 14 (diciembre). https://doi.org/10.17013/risti.14.101-116.

Tolk, A., Lisa Jean Bair, y Saikou Y. Diallo. 2013. "Supporting Network Enabled Capability by Extending the Levels of Conceptual Interoperability Model to an Interoperability Maturity Model". *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 10 (2): 145–60. https://doi.org/10.1177/1548512911428457.

Tolk, A., Saikou Y. Diallo, y Jose J. Padilla. 2012. "Semiotics, entropy, and interoperability of simulation systems—Mathematical foundations of M&S standardization". En *Simulation Conference (WSC), Proceedings of the 2012 Winter*, 1–12. IEEE.

Topçu, Okan, y Halit Oğuztüzün. 2013. "Layered simulation architecture: A practical approach". *Simulation Modelling Practice and Theory* 32 (marzo): 1–14. https://doi.org/10.1016/j.simpat.2012.11.001.

———. 2017. “High Level Architecture”. En *Guide to Distributed Simulation with HLA*, de Okan Topçu y Halit Oğuztüzün, 29–78. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-61267-6_2.

Turnitsa, Charles, Jose J. Padilla, y Andreas Tolk. 2010. “Ontology for modeling and simulation”. En *Simulation Conference (WSC), Proceedings of the 2010 Winter*, 643–651. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5679124.

Uygun, Özer, Ercan Öztemel, y Cemalettin Kubat. 2009. “Scenario Based Distributed Manufacturing Simulation Using HLA Technologies”. *Information Sciences* 179 (10): 1533–41. https://doi.org/10.1016/j.ins.2008.10.019.

W3C. 2004. “SWRL: A Semantic Web Rule Language Combining OWL and RuleML”. mayo de 2004. https://www.w3.org/Submission/SWRL/.

———. 2012. “OWL 2 Web Ontology Language Document Overview (Second Edition)”. 11 de diciembre de 2012. https://www.w3.org/TR/owl2-overview/.

———. 2013. “SPARQL 1.1 Overview”. 21 de marzo de 2013. https://www.w3.org/TR/sparql11-overview/.

———. 2014a. “RDF - Semantic Web Standards”. 25 de febrero de 2014. https://www.w3.org/RDF/.

———. 2014b. “RDF Schema 1.1”. 25 de febrero de 2014. https://www.w3.org/TR/rdf-schema/.

———. 2015. “Extensible Markup Language (XML)”. 2015. https://www.w3.org/XML/.

———. 2017. “HTML 5.2”. 14 de diciembre de 2017. https://www.w3.org/TR/html52/.

Yoo, Taejong, Hyunbo Cho, y Enver Yücesan. 2010. “Hybrid Algorithm for Discrete Event Simulation Based Supply Chain Optimization”. *Expert Systems with Applications* 37 (3): 2354–61. https://doi.org/10.1016/j.eswa.2009.07.039.

Yunus, E.N., y S.K. Tadisina. 2016. “Drivers of supply chain integration and the role of organizational culture: Empirical evidence from Indonesia”. *Business Process Management Journal* 22 (1): 89–115. https://doi.org/10.1108/BPMJ-12-2014-0127.

Zacharewicz, G., D. Chen, y B. Vallespir. 2009. “Short-Lived Ontology Approach for Agent/HLA Federated Enterprise Interoperability”. En *2009 International Conference on Interoperability for Enterprise Software and Applications China*, 329–35. https://doi.org/10.1109/I-ESA.2009.27.

Zacharewicz, Gregory, David Chen, y Bruno Vallespir. 2008. “HLA Supported, Federation Oriented Enterprise Interoperability, Application to Aerospace Enterprises”. En *Proceedings of the 2008 Summer Computer Simulation Conference*, 55:1–55:12. SCSC ’08. Vista, CA: Society for Modeling & Simulation International. http://dl.acm.org/citation.cfm?id=2367656.2367711.

Zehe, Daniel, Alois Knoll, Wentong Cai, y Heiko Aydt. 2015. “SEMSim Cloud Service: Large-Scale Urban Systems Simulation in the Cloud”. *Simulation Modelling Practice and Theory* 58 (noviembre): 157–71. https://doi.org/10.1016/j.simpat.2015.05.005.

Zeigler, Bernard P., y Phillip E. Hammonds. 2007. *Modeling & Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange*. 1 edition. Burlington, MA: Elsevier Academic Press.

Zhang, YongPing, y JingYuan Zhang. 2013. “The study of design for torpedo homing-performance simulation system based on high level architecture”. En *2013 IEEE 4th International Conference on Software Engineering and Service Science*, 1045–48. https://doi.org/10.1109/ICSESS.2013.6615486.

Zhao, Y., J. Dong, y T. Peng. 2009. “Ontology Classification for Semantic-Web-Based Software Engineering”. *IEEE Transactions on Services Computing* 2 (4): 303–17. https://doi.org/10.1109/TSC.2009.20.

Zhu, Jinda, y Libing Liu. 2014. “A collaboration ontology modeling method for high level architecture”. *Computer Modelling and New Technologies* 18 (12): 202–8.

Anexo A Métricas del modelo SCOR

En este anexo se presenta, como se descomponen cada una de las métricas, del modelo SCOR. Las métricas que se presentan en color azul son de nivel uno, las de color violeta son de nivel dos y, por último, las de color naranja son de nivel tres.

Perfect Order Fulfillment ID: RL.1.1

Formula: *[Total Perfect Orders] / [Total Number of Orders] x 100%.* Unit of Measure: %

Una orden es considerada perfecta si se cumplen 4 requisitos: Que la entrega sea completa, entrega en la fecha solicitada por el cliente, documentación exacta y condición perfecta.
Cada requisito se puntúa con 1 si se cumple o con 0 si no.
Si "Entrega Completa + Entrega en Fecha + Documentación + Condición Perfecta = 4" entonces *la orden es perfecta*.

% of Orders Delivered In Full ID: RL.2.1

Formula: *[Total number of orders delivered in full] / [Total number of orders delivered] x 100%*

Una orden es considerada entregada completa si se cumplen 2 requisitos: Se entregan todos los ítems ordenados y en las cantidades solicitadas.

Delivery Item Accuracy ID: RL.3.33

Formula: *[Total number of orders with all items ordered are actually provided] / [Total number of orders delivered] x 100%*

Delivery Quantity Accuracy ID: RL.3.35

Formula: *[Total number of orders with all quantities ordered are actually provided] / [Total number of orders delivered] x 100%*

Delivery Performance to Customer Commit Date ID: RL.2.2

Formula: *[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%*

Una orden es considerada entregada en la fecha solicitada por el cliente si se cumplen 2 requisitos: La orden es recibida en el tiempo definido por el cliente y la entrega se hace en la localización correcta.

Customer Commit Date Achievement Time Customer Receiving ID: RL.3.32

Formula: *[Total number of orders received on time] / [Total number of orders delivered] x 100%*

Delivery Location Accuracy ID: RL.3.34

Formula: *[Total number of orders delivered on the correct location] / [Total number of orders delivered] x 100%*

Documentation Accuracy ID: RL.2.3

Formula: *[Total number of orders delivered with accurate documentation] / [Total number of orders delivered] x 100%*

Una orden es considerada entregada con documentación exacta si se cumplen 4 requisitos: Documentación de embarque (envío), documentación de pago (boleta), documentación de cumplimiento (seguridad del material) y otra documentación requerida (certificado de calidad).

Compliance Documentation Accuracy ID: RL.3.31

Formula: *[Total number of orders with complete, correct and readily available documentations] / [Total number of orders delivered] x 100%*

Other Required Documentation Accuracy ID: RL.3.43

Formula: *[Total number of orders with complete, correct and readily available other documentation] / [Total number of orders delivered] x 100%*

Payment Documentation Accuracy ID: RL.3.45

Formula: *[Total number of orders with complete, correct and readily available payment documentations] / [Total number of orders delivered] x 100%*

Shipping Documentation Accuracy ID: RL.3.50

Formula: *[Total number of orders with complete, correct and readily available shipping documentations] / [Total number of orders delivered] x 100%*

Perfect Condition ID: RL.2.4

Formula: *[Total number of orders delivered in Perfect Condition] / [Total number of orders delivered] x 100%*

Una orden es considerada entregada en condición perfecta si se cumplen 4 requisitos: Entrega sin daños, de acuerdo a la especificación y con configuración correcta (si aplica), instalado sin fallas (si aplica) y aceptado por el cliente, sin devolución para reparar o reemplazar (dentro del periodo de garantía).

% Of Faultless Installations ID: RL.3.12

Formula: *[Total number of faultless installations] / [Total number of units installed] x 100%*

% Orders/Lines Received Damage Free ID: RL.3.24

Formula: *[Total number of orders/lines processed damage free] / [Total number of orders processed] x 100%*

Orders Delivered Damage Free Conformance ID: RL.3.41

Formula: *[Total number of orders delivered without damage] / [Total number of orders delivered] x 100%*

Orders Delivered Defect Free Conformance ID: RL.3.42

Formula: *[Total number of orders delivered without defect] / [Total number of orders delivered] x 100%*

Warranty and Returns ID: RL.3.55

Valor: *[Total number of returns within the warranty period].*

La Tabla A.1 resume la descomposición de las métricas descriptas anteriormente, las cuales forman parte del atributo *Reliability*.

Tabla A.1 – Métricas del atributo confiabilidad

Atributo	Métrica Nivel 1	Formula Métrica Nivel 1	Unidad	Métrica Nivel 2	Formula Métrica Nivel 2	Métrica Nivel 3	Formula Métrica Nivel 3
Reliability (RL)	Perfect Order Fulfillment	[Total Perfect Orders] / [Total Number of Orders] x 100%	% (Porcentaje)	% of Orders Delivered In Full	[Total number of orders delivered in full] / [Total number of orders delivered] x 100%	Delivery Item Accuracy	[Total number of orders with all items ordered are actually provided] / [Total number of orders delivered] x 100%
						Delivery Quantity Accuracy	[Total number of orders with all quantities ordered are actually provided] / [Total number of orders delivered] x 100%
				Delivery Performance to Customer Commit Date	[Total number of orders delivered on the original commitment date] / [Total number of orders delivered] x 100%	Customer Commit Date Achievement Time Customer Receiving	[Total number of orders received on time] / [Total number of orders delivered] x 100%
						Delivery Location Accuracy	[Total number of orders delivered on the correct location] / [Total number of orders delivered] x 100%

				Documentation Accuracy	[Total number of orders delivered with accurate documentation] / [Total number of orders delivered] x 100%	Compliance Documentation Accuracy	[Total number of orders with complete, correct and readily available documentations] / [Total number of orders delivered] x 100%
						Other Required Documentation Accuracy	[Total number of orders with complete, correct and readily available other documentation] / [Total number of orders delivered] x 100%
						Payment Documentation Accuracy	[Total number of orders with complete, correct and readily available payment documentations] / [Total number of orders delivered] x 100%
						Shipping Documentation Accuracy	[Total number of orders with complete, correct and readily available shipping documentations] / [Total number of orders delivered] x 100%

				Perfect Condition	[Total number of orders delivered in Perfect Condition] / [Total number of orders delivered] x 100%	% Of Faultless Installations	[Total number of faultless installations] / [Total number of units installed] x 100%
						% Orders/Lines Received Damage Free	[Total number of orders/lines processed damage free] / [Total number of orders processed] x 100%
						Orders Delivered Damage Free Conformance	[Total number of orders delivered without damage] / [Total number of orders delivered] x 100%
						Orders Delivered Defect Free Conformance	[Total number of orders delivered without defect] / [Total number of orders delivered] x 100%
						Warranty and Returns	Value

Order Fulfillment Cycle Time ID: RS.1.1

Esta es la formula original: *[Sum Actual Cycle Times For All Orders Delivered] / [Total Number Of Orders Delivered]* in days. La misma se puede desglosar en:

Formula: *[(Source CT + Make CT + Deliver CT + Deliver Retail CT) For All Orders Delivered] / [Total Number Of Orders Delivered]* in days. Unit of Measure: Days.

Donde CT = Cycle Time.
Esta métrica mide el ciclo promedio de tiempo para cumplir con las ordenes de los clientes. Para cada orden, este ciclo comienza con la recepción del pedido y finaliza con la aceptación del cliente de la orden.

Source CT ID: RS.2.1

Formula: *(Identify Sources of Supply CT + Select Supplier and Negotiate CT) + Schedule Product Deliveries CT + Receiving Product CT + Verify Product CT + Transfer Product CT + Authorize Supplier Payment CT*

Cuando la estrategia To Stock ó (exclusivo) To Order es utilizada, los conceptos "identify sources of supply CT y select supplier and negotiate CT" no se usan/calculan.

Authorize Supplier Payment CT ID: RS.3.8

Valor: *Average time associated with authorize payment to suppliers*

Identify Sources of Supply CT ID: RS.3.35

Valor: *Average time associated with identify sources of suppliers*

Receiving Product CT ID: RS.3.113

Valor: *Total elapsed time from product is received to when it is passed to next process*

Schedule Product Deliveries CT ID: RS.3.122

Valor: *Average time associated with scheduling shipment of return of MRO product*

Select Supplier and Negotiate CT ID: RS.3.125

Valor: *Average time associated with select and negotiates with supplier*

Transfer Product CT ID: RS.3.139

Valor: *Average time associated with transfer until product is moved to the next process*

Verify Product CT ID: RS.3.140

Valor: *Average time associated with verifying raw material product*

Make CT ID: RS.2.2

Formula: *(Finalize Production Engineering CT) + Schedule Production Activities CT + Issue Material/Product CT + Produce Test CT + Package CT + Stage Finished Product CT + Release Finished Product Deliver CT*

Cuando la estrategia To Stock ó (exclusivo) To Order es utilizada, el concepto "Finalize Production Engineering CT" no se usa/calcula. Cuando se usa la estrategia To Stock la métrica "Issue Material/Product CT" estima el tiempo para proveer el material, sino en el caso de las otras estrategias mide el ciclo de tiempo para proveer los insumos o productos internos del proceso.

Finalize Production Engineering CT ID: RS.3.33

Valor: *Average time associated with finalization of production engineering*

Issue Material CT ID: RS.3.49

Valor: *Average time associated with issuance of material to production*

Produce and Test CT ID: RS.3.101

Valor: *Average time associated with production and test*

Release Finished Product to Deliver CT ID: RS.3.114

Valor: *Average time associated with releasing finished product to deliver*

Schedule Production Activities CT ID: RS.3.123

Valor: *Average time associated with scheduling production activities*

Stage Finished Product CT ID: RS.3.128

Valor: *Average time associated with staging finished product*

Package CT ID: RS.3.142

Valor: *Average time associated with packaging the product in make process*

Deliver CT ID: RS.2.3

Formula: *MAX {[Resources & Determine Delivery Date CT + (Consolidate Orders CT + Schedule Installation CT) + Build Loads CT + Route Shipments CT + Select Carriers and Rate Shipments CT], Receive Product from Make/Source CT} + Pick Product CT + Pack Product CT + Load Vehicle & Generate Shipping Documentation CT + Ship Product CT + (Receive & Verify Product CT) + (Install Product CT)*

La función MAX es para indicar que las tareas sDx.3 – sDx.7 pueden estar en paralelo con la sDx.8 y la que lleve más tiempo determina el ciclo de tiempo.

Build Loads CT ID: RS.3.16

Valor: *Average time associated with building shipments loads*

Consolidate Orders CT ID: RS.3.18

Valor: *Average time required for customer order consolidation*

Install Product CT ID: RS.3.46

Valor: *Average time associated with product installation*

Load Product & Generate Shipping Documentation CT ID: RS.3.51

Valor: *Average time associated with product loading and generation of shipping documentation*

Pack Product CT ID: RS.3.95

Valor: *Average time associated with packing a product for shipment*

Pick Product CT ID: RS.3.96

Valor: *Average time associated with product pick*

Receive & Verify Product by Customer CT ID: RS.3.102

Valor: *Average time associated with receiving and verifying an order at customer site*

Receive Product from Source/Make CT ID: RS.3.110

Valor: *Average time associated with receiving a transfer of product to deliver processes from source or make*

Receive, Configure, Enter & Validate Order CT ID: RS.3.111

Valor: *Average time associated with receiving and verifying an order at customer site*

Reserve Resources and Determine Delivery Date CT ID: RS.3.116

Valor: *Average time associated with reserving resources and determining a delivery date*

Route Shipments CT ID: RS.3.117

Valor: *Average time associated with routing shipments*

Schedule Installation CT ID: RS.3.120

Valor: *Average time associated with scheduling installation of product*

Select Carriers & Rate Shipments CT ID: RS.3.124

Valor: *Average time associated with selecting carriers and rating shipments*

Ship Product CT ID: RS.3.126

Valor: *Average time associated with shipping product*

Deliver Retail CT ID: RS.2.4

Formula: *Generate Stocking Schedule CT + Receive Product CT + Pick Product CT + Stock Shelf CT + Fill Shopping Cart CT + Checkout CT + Install CT*

Checkout CT ID: RS.3.17

Valor: *Average time required for customer checkout*

Fill Shopping Cart CT ID: RS.3.32

Valor: *Average time associated with "filling the shopping cart"*

Generate Stocking Schedule CT ID: RS.3.34

Valor: *Average time associated with generating a stocking schedule*

Pick Product from Backroom CT ID: RS.3.97

Valor: *Average time associated with product pick from backroom*

Receive Product at Store CT ID: RS.3.109

Valor: *Average time associated with receiving product at customer store*

Stock Shelf CT ID: RS.3.129

Valor: *Average time associated with stocking shelves*

La Tabla A.2 resume la descomposición de las métricas descriptas anteriormente, las cuales forman parte del atributo *Responsiveness*.

Tabla A.2 – Métricas del atributo capacidad de respuesta

Atributo	Métrica Nivel 1	Formula Métrica Nivel 1	Unidad	Métrica Nivel 2	Formula Métrica Nivel 2	Métrica Nivel 3	Formula Métrica Nivel 3
Responsiveness (RS)	Order Fulfillment Cycle Time	[(Source CT + Make CT + Deliver CT + Deliver Retail CT) For All Orders Delivered] / [Total Number Of Orders Delivered]	Days	Source Cycle Time	(Identify Sources of Supply CT + Select Supplier and Negotiate CT) + Schedule Product Deliveries CT + Receiving Product CT + Verify Product CT + Transfer Product CT + Authorize Supplier Payment CT	Authorize Supplier Payment CT	Value
						Identify Sources of Supply CT	Value
						Receiving Product CT	Value
						Schedule Product Deliveries CT	Value
						Select Supplier and Negotiate CT	Value
						Transfer Product CT	Value
						Verify Product CT	Value
				Make Cycle Time	(Finalize Production Engineering CT) + Schedule Production Activities CT + Issue Material/Product CT + Produce & Test CT + Package CT + Stage Finished Product CT + Release Finished Product Deliver CT	Finalize Production Engineering CT	Value
						Schedule Production Activities CT	Value
						Issue Material/Product CT	Value
						Produce & Test CT	Value
						Package CT	Value
						Stage Finished Product CT	Value
						Release Finished Product Deliver CT	Value
				Deliver Cycle Time	MAX {[Resources & Determine Delivery Date CT + (Consolidate	Build Loads CT	Value
						Consolidate Orders CT	Value
						Install Product CT	Value

					Orders CT + Schedule Installation CT) + Build Loads CT + Route Shipments CT + Select Carriers and Rate Shipments CT], Receive Product from Make/Source CT} + Pick Product CT + Pack Product CT + Load Vehicle & Generate Shipping Documentation CT + Ship Product CT + (Receive & Verify Product CT) + (Install Product CT)	Load Product & Generate Shipping Documentation CT	Value
						Pack Product CT	Value
						Pick Product CT	Value
						Receive & Verify Product by Customer CT	Value
						Receive Product from Source/Make CT	Value
						Receive, Configure, Enter & Validate Order CT	Value
						Reserve Resources and Determine Delivery Date CT	Value
						Route Shipments CT	Value
						Schedule Installation CT	Value
						Select Carriers & Rate Shipments CT	Value
						Ship Product CT	Value
				Deliver Retail Cycle Time	Generate Stocking Schedule CT + Receive Product CT + Pick Product CT + Stock Shelf CT +	Checkout CT	Value
						Fill Shopping Cart CT	Value
						Generate Stocking Schedule CT	Value
						Pick Product from	Value

					Fill Shopping Cart CT + Checkout CT + Install CT	Backroom CT Receive Product at Store CT Stock Shelf CT	Value Value

Upside SC Flexibility ID: AG.1.1

Esta métrica calcula el número de días que son necesarios para cubrir un incremento sostenido no planeado del 20% en las cantidades que deben entregarse.

Formula: *P * MAX {Upside Source Flexibility, Upside Make Flexibility, Upside Deliver Flexibility, Upside Source Return Flexibility, Upside Deliver Return Flexibility} + (1-P) * [Upside Source Flexibility + Upside Make Flexibility + Upside Deliver Flexibility + Upside Source Return Flexibility + Upside Deliver Return Flexibility]* Unit of Measure: Days.

Donde P = 1 Si las métricas se pueden calcular en paralelo.
P = 0 En caso contrario, es decir, las métricas se calculan secuencialmente.
Unplanned Event = Fecha (Tiempo) en el que ocurre el evento no planeado que provoca el incremento sostenido del 20% en las cantidades que deben entregarse.

Upside Source Flexibility ID: AG.2.1

Esta métrica calcula el número de días que son necesarios para cubrir un incremento sostenido no planeado del 20% en las cantidades de materias primas.

Valor: *Days Required Achieve 20% Increase Raw Materials*

Como esta es una actividad de planificación, generalmente el resultado de la misma es una estimación. Se puede estimar a partir de la salida de las métricas de los otros atributos.

Upside Make Flexibility ID: AG.2.2

Esta métrica calcula el número de días que son necesarios para cubrir un incremento sostenido no planeado del 20% en las cantidades a producir. No se consideran restricciones de materias primas.

Valor: *Days Required Achieve 20% Increase Production*

Upside Deliver Flexibility ID: AG.2.3

Esta métrica calcula el número de días que son necesarios para cubrir un incremento sostenido no planeado del 20% en las cantidades que deben entregarse. No se consideran otras restricciones.

Valor: *Days Required Achieve 20% Increase Quantity Delivered*

Upside Source Return Flexibility ID: AG.2.4

Esta métrica calcula el número de días que son necesarios para cubrir un incremento sostenido no planeado del 20% en las materias primas que deben devolverse a los proveedores.

Valor: *Days Required Achieve 20% Increase Return Raw Materials*

Upside Deliver Return Flexibility ID: AG.2.5

Esta métrica calcula el número de días que son necesarios para cubrir un incremento sostenido no planeado del 20% en los productos finales que devuelven los clientes.

Valor: *Days Required Achieve 20% Increase Return Finished Goods*

Upside SC Adaptability ID: AG.1.2

Esta métrica calcula el máximo porcentaje de incremento sostenible en las cantidades que deben entregarse en un plazo de 30 días. El plazo de días se puede cambiar y es útil solo a propósitos de benchmarking.

Formula: *MIN {Upside Source Adaptability, Upside Make Adaptability, Upside Deliver Adaptability, Upside Source Return Adaptability, Upside Deliver Return Adaptability}* Unit of Measure: %.

Upside Source Adaptability ID: AG.2.6

Esta métrica calcula el máximo porcentaje de incremento sostenible en las cantidades de materias primas que pueden ser adquiridas/recibidas en un plazo de 30 días.

Valor: *Max % Increase Raw Materials Quantities*

Upside Make Adaptability ID: AG.2.7

Esta métrica calcula el máximo porcentaje de incremento sostenible en las cantidades a producir en un plazo de 30 días, sin considerar restricciones de materias primas.

Valor: *Max % Increase Production*

Upside Deliver Adaptability ID: AG.2.8

Esta métrica calcula el máximo porcentaje de incremento sostenible en las cantidades que deben entregar en un plazo de 30 días, sin considerar restricciones de disponibilidad de bienes terminados.

Valor: *Max % Increase Quantities Delivered*

Upside Source Return Adaptability ID: AG.2.9

Esta métrica calcula el máximo porcentaje de incremento sostenible en las cantidades de materias primas que deben devolverse a los proveedores en un plazo de 30 días.

Valor: *Max % Increase Return Raw Materials*

Upside Deliver Return Adaptability ID: AG.2.10

Esta métrica calcula el máximo porcentaje de incremento sostenible en las cantidades de productos finales que devuelven los clientes en un plazo de 30 días.

Valor: *Max % Increase Return Finished Goods*

Downside SC Adaptability ID: AG.1.3

Esta métrica calcula el porcentaje de reducción sostenible en las cantidades ordenadas 30 días antes de su entrega sin restricciones de inventario o costo. El plazo de días se puede cambiar y es útil solo a propósitos de benchmarking. La formula presentada por SCOR es: *Downside*

Source Adaptability + Downside Make Adaptability + Downside Deliver Adaptability pero en realidad según las notas sobre cómo se calcula, la fórmula es la siguiente:

Formula: *MIN {Downside Source Adaptability, Downside Make Adaptability, Downside Deliver Adaptability}* Unit of Measure: %.

Según las notas de la métrica, la formula se calcula como el mínimo porcentaje entre las tres reducciones presentadas. Esto se debe la mínima reducción es la que más limita y a su vez el máximo margen de reducción que pueden soportar.

Downside Source Adaptability ID: AG.2.11

Esta métrica calcula el máximo porcentaje de reducción sostenible en las cantidades de materias primas 30 días antes de la entrega sin restricciones de inventario o costo.

Valor: *Raw Material Quantity Reduction*

Downside Make Adaptability ID: AG.2.12

Esta métrica calcula el máximo porcentaje de reducción sostenible en las cantidades a producir 30 días antes de la entrega sin considerar restricciones de inventario o costo.

Valor: *Production Quantities Reduction*

Downside Deliver Adaptability ID: AG.2.13

Esta métrica calcula el máximo porcentaje de reducción sostenible en las cantidades que deben entregarse 30 días antes de la entrega sin considerar restricciones de inventario o costo.

Valor: *Delivered Quantities Reduction*

Overall Value at Risk ID: AG.1.4

Es una métrica financiera para entender la exposición al riesgo de una cartera de comercio basado en la volatilidad histórica.

Formula: *Plan VAR + Source VAR + Make VAR + Deliver VAR + Return VAR*

Unit of Measure: Monetary Units.

Supplier's/Customer's/Products's Risk Rating ID: AG.2.14

Valor: *Numerical Risk Rating*

Plan Value at Risk ID: AG.2.15

Valor: *Plan Value at Risk*

Source Value at Risk ID: AG.2.16

Valor: *Source Value at Risk*

Make Value at Risk ID: AG.2.17

Valor: *Make Value at Risk*

Deliver Value at Risk ID: AG.2.18

Valor: *Deliver Value at Risk*

Return Value at Risk ID: AG.2.19

Valor: *Return Value at Risk*

La Tabla A.3 resume la descomposición de las métricas descriptas anteriormente, las cuales forman parte del atributo *Agility*.

Tabla A.3 – Métricas del atributo agilidad

Atributo	Métrica Nivel 1	Formula Métrica Nivel 1	Unidad	Métrica Nivel 2	Formula Métrica Nivel 2	Métrica Nivel 3
Agility (AG)	Upside SC Flexibility	P * MAX {Upside Source Flexibility, Upside Make Flexibility, Upside Deliver Flexibility, Upside Source Return Flexibility, Upside Deliver Return Flexibility} + (1-P) * [Upside Source Flexibility + Upside Make Flexibility + Upside Deliver Flexibility + Upside Source Return Flexibility + Upside Deliver Return Flexibility]	Days	Upside Source Flexibility	Days Required Achieve 20% Increase Raw Materials	-
				Upside Make Flexibility	Days Required Achieve 20% Increase Production	-
				Upside Deliver Flexibility	Days Required Achieve 20% Increase Quantity Delivered	-
				Upside Source Return Flexibility	Days Required Achieve 20% Increase Return Raw Materials	-
				Upside Deliver Return Flexibility	Days Required Achieve 20% Increase Return Finished Goods	-
	Upside SC Adaptability	MIN {Upside Source Adaptability, Upside Make Adaptability, Upside Deliver Adaptability, Upside Source Return Adaptability, Upside Deliver Return Adaptability}	%	Upside Source Adaptability	Max % Increase Raw Materials Quantities	-
				Upside Make Adaptability	Max % Increase Production	-
				Upside Deliver Adaptability	Max % Increase Quantities Delivered	-
				Upside Source Return Adaptability	Max % Increase Return Raw Materials	-
				Upside Deliver Return Adaptability	Max % Increase Return Finished Goods	-
	Downside SC Adaptability	MIN {Downside Source Adaptability, Downside Make Adaptability, Downside Deliver Adaptability}	%	Downside Source Adaptability	Raw Material Quantity Reduction	-
				Downside Make Adaptability	Production Quantities Reduction	-

				Downside Deliver Adaptability	Delivered Quantities Reduction	-
	Overall Value at Risk (VAR)	Plan VAR + Source VAR + Make VAR + Deliver VAR + Return VAR	Monetary Units	Supplier's/Customer's/Products's Risk Rating Plan Value at Risk Source Value at Risk Make Value at Risk Deliver Value at Risk Return Value at Risk	Numerical Risk Rating Plan Value at Risk Source Value at Risk Make Value at Risk Deliver Value at Risk Return Value at Risk	- - - - - -

Total Cost to Serve ID: CO.1.001

Esta métrica calcula la suma de los costos de la CS para entregar productos o servicios a los clientes.

Formula: *Planning Cost + Sourcing Cost + Material Landed Cost + Production Cost + Order Management Cost + Fulfillment Cost + Returns Cost.*

Unit of Measure: Monetary Units.

Planning Cost ID: CO.2.001

Esta métrica calcula el costo del personal, automatización, bienes y gastos generales asociados a los procesos de planificación.

Formula: *Planning Labor Cost + Planning Automation Cost + Planning Property, Plant, Equipment Cost + Planning Governance, Risk, Compliance (GRC), Overhead Cost*

Planning Labor Cost ID: CO.3.001

Valor: *Planning Labor Cost*

Planning Automation Cost ID: CO.3.002

Valor: *Planning Automation Cost*

Planning Property, Plant, Equipment Cost ID: CO.3.003

Valor: *Planning Property, Plant, Equipment Cost*

Planning GRC, Overhead Cost ID: CO.3.004

Valor: *Planning GRC, Overhead Cost*

Sourcing Cost ID: CO.2.002

Esta métrica calcula el costo de gestionar la orden, recibir, inspeccionar, almacenar los materiales y productos.

Formula: *Sourcing Labor Cost + Sourcing Automation Cost + Sourcing Property, Plant, Equipment Cost + Sourcing Governance, Risk, Compliance (GRC), Inventory, Overhead Cost*

Sourcing Labor Cost ID: CO.3.005

Valor: *Sourcing Labor Cost*

Sourcing Automation Cost ID: CO.3.006

Valor: *Sourcing Automation Cost*

Sourcing Property, Plant, Equipment Cost ID: CO.3.007

Valor: *Sourcing Property, Plant, Equipment Cost*

Sourcing GRC, Inventory, Overhead Cost ID: CO.3.008

Valor: *Sourcing GRC, Inventory, Overhead Cost*

Material Landed Cost ID: CO.2.003

Esta métrica calcula el costo de comprar o fabricar materiales, productos o mercaderías disponibles en el lugar de uso.

Formula: *Purchased Material Cost + Material Transportation Cost + Material Customs, Duties, Taxes, Tariffs Cost + Material Risk, Compliance Cost*

Purchased Material Cost ID: CO.3.009

Valor: *Purchased Material Cost*

Material Transportation Cost ID: CO.3.010

Valor: *Material Transportation Cost*

Material Customs, Duties, Taxes, Tariffs Cost ID: CO.3.011

Valor: *Material Customs, Duties, Taxes, Tariffs Cost*

Material Risk, Compliance Cost ID: CO.3.012

Valor: *Material Risk, Compliance Cost*

Production Cost ID: CO.2.004

Esta métrica calcula el costo asociado con gestionar y realizar la producción.

Formula: *Production Labor Cost + Production Automation Cost + Production Property, Plant, Equipment Cost + Production Governance, Risk, Compliance (GRC), Inventory, Overhead Cost*

Production Labor Cost ID: CO.3.014

Valor: *Production Labor Cost*

Production Automation Cost ID: CO.3.015

Valor: *Production Automation Cost*

Production Property, Plant, Equipment Cost ID: CO.3.016

Valor: *Production Property, Plant, Equipment Cost*

Production GRC, Inventory, Overhead Cost ID: CO.3.017

Valor: *Production GRC, Inventory, Overhead Cost*

Order Management Cost ID: CO.2.005

Esta métrica calcula el costo de personal, automatización y bienes asociados con la gestión de órdenes, tanto en sus consultas como en las entregas y cobros.

Formula: *Order Management Labor Cost + Order Management Automation Cost + Order Management Property, Plant, Equipment Cost + Order Management Governance, Risk, Compliance (GRC), Overhead Cost*

Order Management Labor Cost ID: CO.3.018

Valor: *Order Management Labor Cost*

Order Management Automation Cost ID: CO.3.019

Valor: *Order Management Automation Cost*

Order Management Property, Plant, Equipment Cost ID: CO.3.020

Valor: *Order Management Property, Plant, Equipment Cost*

Order Management GRC, Overhead Cost ID: CO.3.021

Valor: *Order Management GRC, Inventory, Overhead Cost*

Fulfillment Cost ID: CO.2.006

Esta métrica calcula el costo total de personal, automatización, bienes y costo generales asociados con el cumplimiento de órdenes.

Formula: *Transportation Cost + Fulfillment Customs, Duties, Taxes, Tariffs Cost + Fulfillment Labor Cost + Fulfillment Automation Cost + Fulfillment Property, Plant, Equipment Cost + Fulfillment Governance, Risk, Compliance (GRC), Inventory, Overhead Cost*

Transportation Cost ID: CO.3.022

Valor: *Transportation Labor Cost*

Fulfillment Customs, Duties, Taxes, Tariffs Cost ID: CO.3.023

Valor: *Fulfillment Customs, Duties, Taxes, Tariffs Cost*

Fulfillment Labor Cost ID: CO.3.024

Valor: *Fulfillment Labor Cost*

Fulfillment Automation Cost ID: CO.3.025

Valor: *Fulfillment Automation Cost*

Fulfillment Property, Plant, Equipment Cost ID: CO.3.026

Valor: *Fulfillment Property, Plant, Equipment Cost*

Fulfillment GRC, Inventory, Overhead Cost ID: CO.3.027

Valor: *Fulfillment GRC, Inventory, Overhead Cost*

Returns Cost ID: CO.2.007

Esta métrica calcula el costo de disposición de los materiales devueltos debido a errores de planificación, calidad de producción del proveedor, gestión de la orden y errores de entrega. Estos costos se pueden describir como los costos de "retrabajo" debido a una entrega imperfecta al cliente.

Formula: *Discounts, Refunds Cost + Disposition Cost + Return Governance, Risk, Compliance (GRC), Inventory, Overhead Cost*

Discounts, Refunds Cost ID: CO.3.028

Valor: *Discounts, Refund Cost*

Disposition Cost ID: CO.3.029

Valor: *Disposition Cost*

Return GRC, Inventory, Overhead Cost ID: CO.3.030

Valor: *Return GRC, Inventory, Overhead Cost*

Cost of Goods Sold ID: CO.2.008

Esta métrica calcula el costo de los materiales directos, del trabajo directo y los gastos generales asociados con la adquisición o producción de bienes terminados.

Formula: *Purchased Material Cost + Direct Labor Cost + Production Labor Cost + Production Automation Cost + Production Property, Plant, Equipment Cost + Production GRC, Inventory, Overhead Cost*

La Tabla A.4 resume la descomposición de las métricas descriptas anteriormente, las cuales forman parte del atributo *Cost*.

Tabla A.4 – Métricas del atributo costo

Atributo	Métrica Nivel 1	Formula Métrica Nivel 1	Unidad	Métrica Nivel 2	Formula Métrica Nivel 2	Métrica Nivel 3	Formula Métrica Nivel 3
Cost (CO)	Total Cost to Serve	Planning Cost + Sourcing Cost + Material Landed Cost + Production Cost + Order Management Cost + Fulfillment Cost + Returns Cost	Monetary Units	Planning Cost	Planning Labor Cost + Planning Automation Cost + Planning Property, Plant, Equipment Cost + Planning Governance, Risk, Compliance (GRC), Overhead Cost	Planning Labor Cost	Value
						Planning Automation Cost	Value
						Planning Property, Plant, Equipment Cost	Value
						Planning GRC, Overhead Cost	Value
				Sourcing Cost	Sourcing Labor Cost + Sourcing Automation Cost + Sourcing Property, Plant, Equipment Cost + Sourcing Governance, Risk, Compliance (GRC), Inventory, Overhead Cost	Sourcing Labor Cost	Value
						Sourcing Automation Cost	Value
						Sourcing Property, Plant, Equipment Cost	Value
						Sourcing GRC, Inventory, Overhead Cost	Value
				Material Landed Cost	Purchased Material Cost + Material Transportation Cost + Material Customs, Duties, Taxes, Tariffs Cost + Material Risk, Compliance Cost	Purchased Material Cost	Value
						Material Transportation Cost	Value
						Material Customs, Duties, Taxes, Tariffs Cost	Value
						Material Risk, Compliance Cost	Value
				Production Cost	Production Labor Cost + Production Automation Cost + Production Property, Plant, Equipment Cost + Production Governance,	Production Labor Cost	Value
						Production Automation Cost	Value
						Production Property, Plant, Equipment Cost	Value

					Risk, Compliance (GRC), Inventory, Overhead Cost	Production GRC, Inventory, Overhead Cost	Value
				Order Management Cost	Order Management Labor Cost + Order Management Automation Cost + Order Management Property, Plant, Equipment Cost + Order Management Governance, Risk, Compliance (GRC), Overhead Cost	Order Management Labor Cost Order Management Automation Cost Order Management Property, Plant, Equipment Cost Order Management GRC, Inventory, Overhead Cost	Value Value Value Value
				Fulfillment Cost	Transportation Cost + Fulfillment Customs, Duties, Taxes, Tariffs Cost + Fulfillment Labor Cost + Fulfillment Automation Cost + Fulfillment Property, Plant, Equipment Cost + Fulfillment Governance, Risk, Compliance (GRC), Inventory, Overhead Cost	Transportation Labor Cost Fulfillment Customs, Duties, Taxes, Tariffs Cost Fulfillment Labor Cost Fulfillment Automation Cost Fulfillment Property, Plant, Equipment Cost Fulfillment GRC, Inventory, Overhead Cost	Value Value Value Value Value Value
				Returns Cost	Discounts, Refunds Cost + Disposition Cost + Return Governance, Risk, Compliance (GRC),	Discounts, Refund Cost Disposition Cost Return GRC, Inventory,	Value Value Value

					Inventory, Overhead Cost	Overhead Cost	
				Cost of Goods Sold	Purchased Material Cost + Direct Labor Cost + Production Labor Cost + Production Automation Cost + Production Property, Plant, Equipment Cost + Production GRC, Inventory, Overhead Cost	Purchased Material Cost Direct Labor Cost Production Labor Cost Production Automation Cost Production Property, Plant, Equipment Cost Production GRC, Inventory, Overhead Cost	Value Value Value Value Value Value

Cash To Cash Cycle Time ID: AM.1.1

Esta métrica calcula el tiempo que demora una inversión en retornar ganancias para la compañía luego de la compra de materias primas.

Formula: *Inventory Days Supply + Days Sales Outstanding – Days Payable Outstanding.*

Unit of Measure: Days.

Aclaración: Para las siguientes métricas, el "5 point rolling average" se calcula como:

$$\frac{\sum_{t=1}^{t=n-1} X_t + X_n}{n}$$

Donde las X son los valores para cada trimestre y n = 5 (los períodos a contar).

Es decir que es [(Suma de los 4 trimestres anteriores + Trimestre actual) / 5]

Days Sales Outstanding ID: AM.2.1

Esta métrica calcula el periodo de tiempo desde que una venta es hecha hasta que el dinero de los clientes es recibido.

Formula: *5 Point Annual Average Gross Accounts Receivable / (Total Gross Annual Sales / 365).*

Unit of Measure: Days. En donde "5 Point Annual Average" se considera como "5 Point Rolling Average Annual"

Inventory Days of Supply ID: AM.2.2

Esta métrica calcula la cantidad de inventario (stock) expresado en días de venta.

Formula: *5 Point Rolling Average Gross Value Inventory At Standard Cost / (Annual Cost Goods Sold / 365)*

Days Payable Outstanding ID: AM.2.3

Esta métrica calcula el periodo de tiempo desde la compra de materiales, trabajo, y/o conversión de recursos hasta que los pagos son efectuados.

Formula: *5 Point Rolling Average Gross Accounts Payable / (Total Gross Annual Material Purchases / 365)*

Return on Supply Chain Fixed Assets ID: AM.1.2

Esta métrica calcula el retorno de la inversión de capital en los bienes (activos) de la CS.

Formula: (*Supply Chain Revenue – Total Cost Serve) / Supply Chain Fixed Assets.*

Unit of Measure: Monetary Units.

Supply Chain Revenue ID: AM.2.4

Esta métrica calcula el ingreso operativo de la CS.

Valor: *Supply Chain Revenue.*

Supply Chain Fixed Assets ID: AM.2.5

Esta métrica calcula los costos asociados con los bienes (activos) fijos de los procesos Plan, Source, Make, Deliver y Return.

Formula: *Plan Fixed Asset Value + Source Fixed Asset Value + Make Fixed Asset Value + Deliver Fixed Asset Value + Return Fixed Asset Value*

Nota: Los términos que se encuentran en esta fórmula, no se presentan en SCOR. Esto se debe a que los términos de la formula son solo valores atómicos (mediciones puntuales sobre un objeto).

Return on Working Capital ID: AM.1.3

Esta métrica calcula la magnitud de la inversión relativa al capital versus los ingresos de la CS.

Formula: (*Supply Chain Revenue – Total Cost Serve) / (Inventory + Accounts Receivable – Accounts Payable).* Unit of Measure: Monetary Units.

Accounts Payable ID: AM.2.6

Esta métrica calcula la cantidad de materiales comprados, trabajo, y/o conversión de recursos que se deben pagar, expresados en unidades monetarias.

Valor: *5 Point Rolling Average Gross Accounts Payable*

Accounts Receivable ID: AM.2.7

Esta métrica calcula la cantidad de cuentas por cobrar pendientes, expresada en unidades monetarias.

Valor: *5 Point Rolling Average Gross Accounts Receivable*

Inventory ID: AM.2.8

Esta métrica calcula la cantidad de inventario expresada en unidades monetarias.

Valor: *5 Point Rolling Average Gross Value Inventory At Standard Cost*

La Tabla A.5 resume la descomposición de las métricas descriptas anteriormente, las cuales forman parte del atributo *Assests*.

Tabla A.5 – Métricas del atributo activos

Atributo	Métrica Nivel 1	Formula Métrica Nivel 1	Unidad	Métrica Nivel 2	Formula Métrica Nivel 2	Métrica Nivel 3	Formula Métrica Nivel 3
Assets (AM)	Cash-to-Cash Cycle Time	Inventory Days Supply + Days Sales Outstanding – Days Payable Outstanding	Days	Days Sales Outstanding	5 Point Annual Average Gross Accounts Receivable / (Total Gross Annual Sales / 365)	-	-
				Inventory Days of Supply	5 Point Rolling Average Gross Value Inventory At Standard Cost / (Annual Cost Goods Sold / 365)	-	-
				Days Payable Outstanding	5 Point Rolling Average Gross Accounts Payable / (Total Gross Annual Material Purchases / 365)	-	-
	Return on SC Fixed Assets	(Supply Chain Revenue – Total Cost Serve) / Supply Chain Fixed Assets	Monetary Units	Supply Chain Revenue	Supply Chain Revenue	-	-
				Supply Chain Fixed Assets	Plan Fixed Asset Value + Source Fixed Asset Value + Make Fixed Asset Value + Deliver Fixed Asset Value + Return Fixed Asset Value	Plan Fixed Asset Value	Value
						Source Fixed Asset Value	Value
						Make Fixed Asset Value	Value
						Deliver Fixed Asset Value	Value
						Return Fixed Asset Value	Value
	Return on Working Capital		Monetary Units	Accounts Payable	5 Point Rolling Average Gross Accounts Payable	-	-
				Accounts Receivable	5 Point Rolling Average Gross Accounts Receivable	-	-
						-	-

		(Supply Chain Revenue – Total Cost Serve) / (Inventory + Accounts Receivable – Accounts Payable)		Inventory	5 Point Rolling Average Gross Value Inventory At Standard Cost		

Anexo B FOM

En este anexo se presenta el modelo de objetos de federación, obtenido en la prueba de concepto del capítulo 5. Este documento se encuentra en formato XML, y se puede descargar del siguiente link: https://drive.google.com/file/d/1QRd-D4UuGM3Zp6ZByMyd67P5oXb-iszV/view?usp=sharing.

```
<?xml version="1.0"?>
<!DOCTYPE objectModel SYSTEM "HLA.dtd">
<objectModel
        DTDversion="1516.2"
        name="Example"
        type="FOM"
        version="1.0"
        date="2000-04-01"
        purpose="Provide an example of an HLA FOM"
        sponsor="DMSO">
        <objects>
                <objectClass name="HLAobjectRoot"
                        sharing="Neither">
                <attribute name="HLAprivilegeToDeleteObject"
                        dataType="NA"
                        updateType="NA"
                        updateCondition="NA"
```

```
            ownership="NoTransfer"
            sharing="Neither"
            dimensions="NA"
            transportation="HLAreliable"
            order="TimeStamp"/>
        <objectClass name="UserBaseClass"
            sharing="Neither"
            semantics="This object class is the base of all user-defined object
            classes">
            <objectClass name="Total Delivered"
                sharing="PublishSubscribe">
                <attribute name="Unit"
                    dataType="HLAunicodeString"/>
                <attribute name="Value"
                    dataType="HLAfloat64LE"/>
            </objectClass>
            <objectClass name="Received On Time"
                sharing="PublishSubscribe">
                <attribute name="Unit"
                    dataType="HLAunicodeString"/>
                <attribute name="Value"
                    dataType="HLAfloat64LE"/>
            </objectClass>
            <objectClass name="Delivered On Correct Location"
                sharing="PublishSubscribe">
```

```
                    <attribute name="Unit"
                            dataType="HLAunicodeString"/>
                    <attribute name="Value"
                            dataType="HLAfloat64LE"/>
                </objectClass>
            </objectClass>
        <objectClass name="HLAmanager"
        sharing="Neither"
        semantics="This object class is the root class of all MOM object classes">
            <objectClass name="HLAfederate"
                    sharing="Publish"
                    semantics="This object class shall contain RTI state variables
                    relating to a joined federate. The RTI shall publish it
                    and shall register one object instance for each joined
                    federate in a federation. Dynamic attributes that shall
                    be contained in an object instance shall be updated
                    periodically, where the period should be determined by
                    an interaction of the class HLAmanager.
                    HLAfederate.HLAadjust.HLAsetTiming. If this value is
                    never set or is set to zero, no periodic up-date shall
                    be performed by the RTI.">
                    <attribute name="HLAfederateHandle"
                            dataType="HLAhandle"
                            updateType="Static"
                            updateCondition="NA"
```

```
            ownership="NoTransfer"
            sharing="Publish"
            dimensions="Federate"
            transportation="HLAreliable"
            order="Receive"
            semantics="Handle of the joined federate returned by a Join
            Federation Execution service invocation"/>
        <attribute name="HLAfederateType"
            dataType="HLAunicodeString"
            updateType="Static"
            updateCondition="NA"
            ownership="NoTransfer"
            sharing="Publish"
            dimensions="Federate"
            transportation="HLAreliable"
            order="Receive"
            semantics="Type of the joined federate specified by the joined
            federate when it joined the federation"/>
        <attribute name="HLAfederateHost"
            dataType="HLAunicodeString"
            updateType="Static"
            updateCondition="NA"
            ownership="NoTransfer"
            sharing="Publish"
            dimensions="Federate"
```

transportation="HLAreliable"

order="Receive"

semantics="Host name of the computer on which the joined federate

is executing"/>

<attribute name="HLARTIversion"

dataType="HLAunicodeString"

updateType="Static"

updateCondition="NA"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Version of the RTI software being used"/>

<attribute name="HLAFDDID"

dataType="HLAunicodeString"

updateType="Static"

updateCondition="NA"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Identifier associated with the FDD data used by the

```
            joined federate"/>
        <attribute name="HLAtimeConstrained"
            dataType="HLAboolean"
            updateType="Conditional"
            updateCondition="Service invocation"
            ownership="NoTransfer"
            sharing="Publish"
            dimensions="Federate"
            transportation="HLAreliable"
            order="Receive"
            semantics="Whether the time advance of the joined federate is
            constrained by other joined federates"/>
        <attribute name="HLAtimeRegulating"
            dataType="HLAboolean"
            updateType="Conditional"
            updateCondition="Service invocation"
            ownership="NoTransfer"
            sharing="Publish"
            dimensions="Federate"
            transportation="HLAreliable"
            order="Receive"
            semantics="Whether the joined federate influences the time
            advancement of other joined federates"/>
        <attribute name="HLAasynchronousDelivery"
            dataType="HLAboolean"
```

updateType="Conditional"

updateCondition="Service invocation"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Whether the RTI shall deliver receive-order messages

to the joined federate while the joined federate's

time manager state is not Time Advancing (only matters

if the joined federate is time-constrained)"/>

<attribute name="HLAfederateState"

dataType="HLAfederateState"

updateType="Conditional"

updateCondition="Service invocation"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="State of the joined federate"/>

<attribute name="HLAtimeManagerState"

dataType="HLAtimeState"

updateType="Conditional"

updateCondition="Service invocation"
ownership="NoTransfer"
sharing="Publish"
dimensions="Federate"
transportation="HLAreliable"
order="Receive"
semantics="State of the joined federate's time manager" />
<attribute name="HLAlogicalTime"
dataType="HLAlogicalTime"
updateType="Periodic"
updateCondition="HLAsetTiming.HLAreportPeriod"
ownership="NoTransfer"
sharing="Publish"
dimensions="Federate"
transportation="HLAreliable"
order="Receive"
semantics="Joined federate's logical time. Initial value of this information is initial value of time of the Time Representation Abstract datatype (TRADT)."/>
<attribute name="HLAlookahead"
dataType="HLAtimeInterval"
updateType="Periodic"
updateCondition="HLAsetTiming.HLAreportPeriod"
ownership="NoTransfer"
sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Minimum duration into the future that a TSO message

will be scheduled. The value shall not be defined if

the joined federate is not time-regulating)"/>

<attribute name="HLAGALT"

dataType="HLAlogicalTime"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Joined federate's Greatest Available Logical Time (GALT). The value shall not be defined if GALT is not

defined for the joined federate."/>

<attribute name="HLALITS"

dataType="HLAlogicalTime"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Joined federate's Least Incoming Time Stamp (LITS).

The value shall not be defined if LITS is not defined

for the joined federate."/>

<attribute name="HLAROlength"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Number of RO messages queued for delivery to the joined federate."/>

<attribute name="HLATSOlength"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Number of TSO messages queued for delivery to the

joined federate"/>

<attribute name="HLAreflectionsReceived"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Total number of reflections received by the joined federate."/>

<attribute name="HLAupdatesSent"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Total number of updates sent by the joined federate"/>

<attribute name="HLAinteractionsReceived"
dataType="HLAcount"
updateType="Periodic"
updateCondition="HLAsetTiming.HLAreportPeriod"
ownership="NoTransfer"
sharing="Publish"
dimensions="Federate"
transportation="HLAreliable"
order="Receive"
semantics="Total number of interactions received by the joined federate."/>
<attribute name="HLAinteractionsSent"
dataType="HLAcount"
updateType="Periodic"
updateCondition="HLAsetTiming.HLAreportPeriod"
ownership="NoTransfer"
sharing="Publish"
dimensions="Federate"
transportation="HLAreliable"
order="Receive"
semantics="Total number of interactions sent by the joined federate. This information shall reflect related DDM usage."/>
<attribute name="HLAobjectsInstancesThatCanBeDeleted"
dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Total number of object instances whose HLAprivilegeToDeleteObject attribute is owned by the joined federate"/>

<attribute name="HLAobjectInstancesUpdated"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Total number of object instances for which the joined federate has invoked the Update Attribute Values service."/>

<attribute name="HLAobjectInstancesReflected"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"
ownership="NoTransfer"
sharing="Publish"
dimensions="Federate"
transportation="HLAreliable"
order="Receive"
semantics="Total number of object instances for which the joined federate has had a Reflect Attribute Values service invocation."/>
<attribute name="HLAobjectInstancesDeleted"
dataType="HLAcount"
updateType="Periodic"
updateCondition="HLAsetTiming.HLAreportPeriod"
ownership="NoTransfer"
sharing="Publish"
dimensions="Federate"
transportation="HLAreliable"
order="Receive"
semantics="Total number of times the Delete Object Instance service was invoked by the joined federate since the federate joined the federation"/>
<attribute name="HLAobjectInstancesRemoved"
dataType="HLAcount"
updateType="Periodic"
updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Total number of times the Remove Object Instance service was invoked for the joined federate since the federate joined the federation."/>

<attribute name="HLAobjectInstancesRegistered"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Total number of times the Register Object Instance or Register Object Instance with Region service was invoked by the joined federate since the federate joined the federation."/>

<attribute name="HLAobjectInstancesDiscovered"

dataType="HLAcount"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Total number of times the Discover Object Instance service was invoked for the joined federate since the federate joined the federation"/>

<attribute name="HLAtimeGrantedTime"

dataType="HLAmsec"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="Wall clock time duration that the federate has spent in the Time Granted state since the last update of this attribute."/>

<attribute name="HLAtimeAdvancingTime"

dataType="HLAmsec"

updateType="Periodic"

updateCondition="HLAsetTiming.HLAreportPeriod"

ownership="NoTransfer"

```
                sharing="Publish"
                dimensions="Federate"
                transportation="HLAreliable"
                order="Receive"
                semantics="Wall clock time duration that the federate has spent
            in the Time Advancing state since the last update of
            this attribute."/>
        </objectClass>
        <objectClass name="HLAfederation"
        sharing="Publish"
        semantics="This object class shall contain RTI state variablesrelating to a federation execution. The RTI shall
publish it and shall register one object instance for
the federation execution. It shall not automatically
update the values of the instance attributes; a joined
federate shall use a Request Attribute Value Update
service to obtain values for the instance attributes.">
        <attribute name="HLAfederationName"
                dataType="HLAunicodeString"
                updateType="Static"
                updateCondition="NA"
                ownership="NoTransfer"
                sharing="Publish"
                dimensions="NA"
                transportation="HLAreliable"
```

order="Receive"

semantics="Name of the federation to which the joined federate belongs"/>

<attribute name="HLAfederatesinFederation"

dataType="HLAhandleList"

updateType="Conditional"

updateCondition="Federate joins or resigns"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Identifiers of joined federates that are joined to the federation"/>

<attribute name="HLARTIversion"

dataType="HLAunicodeString"

updateType="Static"

updateCondition="NA"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Version of the RTI software"/>

<attribute name="HLAFDDID"

dataType="HLAunicodeString"

updateType="Static"

updateCondition="NA"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Identifier associated with the FDD used in the relevant Create Federation Execution service invocation."/>

<attribute name="HLAlastSaveName"

dataType="HLAunicodeString"

updateType="Conditional"

updateCondition="Service invocation"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Name associated with the last federation state save (null if no saves have occurred)"/>

<attribute name="HLAlastSaveTime"

dataType="HLAlogicalTime"

updateType="Conditional"

updateCondition="Service invocation"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Logical time at which the last federation state timed save occurred. The value shall not be defined if no timed saves have occurred."/>

<attribute name="HLAnextSaveName"

dataType="HLAunicodeString"

updateType="Conditional"

updateCondition="Service invocation"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Name associated with the next federation state save (null if no saves are scheduled)"/>

<attribute name="HLAnextSaveTime"

dataType="HLAlogicalTime"

updateType="Conditional"

updateCondition="Service invocation"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Logical time at which the next federation state timed save is scheduled. The value shall not be defined if no timed saves are scheduled."/>

<attribute name="HLAautoProvide"

dataType="HLAswitch"

updateType="Conditional"

updateCondition="MOM interaction"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Value of federation-wide Auto Provide Switch. Updated when value of switch changes"/>

<attribute name="HLAconveyRegionDesignatorSets"

dataType="HLAswitch"

updateType="Conditional"

updateCondition="MOM interaction"

ownership="NoTransfer"

sharing="Publish"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

```
            semantics="Value of federation-wide Convey Region Designator
Sets Switch. Updated when value of switch changes"/></objectClass>
        </objectClass>
</objectClass>
</objects>
<interactions>
        <interactionClass name="HLAinteractionRoot"
            sharing="Neither"
            dimensions="NA"
            transportation="HLAreliable"
            order="Receive">
            <interactionClass name="UserInteractionBase"
                sharing="Neither"
                dimensions="NA"
                transportation="HLAreliable"
                order="TimeStamp"
                semantics="Base classs of user-defined interactions">
                <interactionClass name="Pedido Suministro"
                    sharing="PublishSubscribe"
                    transportation="HLAreliable"
                    order="TimeStamp">
                    <parameter name="Fecha Solicitada"
                        dataType="date"/>
                    <parameter name="Cantidad"
                        dataType="HLAfloat64LE"/>
```

```
            <parameter name="Item"
                  dataType="HLAfloat64LE"/>
      </interactionClass>
      <interactionClass name="Entrega Suministro"
            sharing="PublishSubscribe"
            transportation="HLAreliable"
            order="TimeStamp">
            <parameter name="Fecha Entrega"
                  dataType="date"/>
            <parameter name="Cantidad"
                  dataType="HLAfloat64LE"/>
            <parameter name="Item"
                  dataType="HLAfloat64LE"/>
      </interactionClass>
      <interactionClass name="Pedido Mercaderia"
            sharing="PublishSubscribe"
            transportation="HLAreliable"
            order="TimeStamp">
            <parameter name="Fecha Solicitada"
                  dataType="date"/>
            <parameter name="Cantidad"
                  dataType="HLAfloat64LE"/>
            <parameter name="Item"
                  dataType="HLAfloat64LE"/>
      </interactionClass>
```

```
                    <interactionClass name="Entrega Mercaderia"
                            sharing="PublishSubscribe"
                            transportation="HLAreliable"
                            order="TimeStamp">
                            <parameter name="Fecha Entrega"
                                    dataType="date"/>
                            <parameter name="Cantidad"
                                    dataType="HLAfloat64LE"/>
                            <parameter name="Item"
                                    dataType="HLAfloat64LE"/>
                    </interactionClass>
            </interactionClass>
        <interactionClass name="HLAmanager"
                sharing="Neither"
                dimensions="NA"
                transportation="HLAreliable"
                order="Receive"
                semantics="Root class of MOM interactions">
                <interactionClass name="HLAfederate"
                        sharing="Neither"
                        dimensions="NA"
                        transportation="HLAreliable"
                        order="Receive"
                        semantics="Root class of MOM interactions that deal with a specific
joined federate">
```

<parameter name="HLAfederate"
dataType="HLAhandle"
semantics="Handle of the joined federate that was provided when
joining."/>
<interactionClass name="HLAadjust"
sharing="Neither"
dimensions="NA"
transportation="HLAreliable"
order="Receive"
semantics="Permit a joined federate to adjust the RTI
statevariables associated with another joined federate">
<interactionClass name="HLAsetTiming"
sharing="Subscribe"
dimensions="NA"
transportation="HLAreliable"
order="Receive"
semantics="Adjust the time period between updates of the
HLAmanager.HLAfederate object instance for thespecified joined federate. If this interaction is
never sent, the RTI shall not perform periodic updates">
<parameter name="HLAreportPeriod"
dataType="HLAseconds"
semantics="Number of seconds between updates of instance

attribute values of the HLAfederate object instance (A zero value causes periodic updates to cease)"/>

</interactionClass>

<interactionClass name="HLAmodifyAttributeState"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Modify the ownership state of an attribute of an

object instance for the specified joined federate. If the interaction is used to give ownership of the

instance attribute to the specified joined federate and another joined federate currently owns the

instance attribute, the owning joined federate shall be divested of ownership of the instance

attribute before ownership is granted to the specified joined federate. No notification of

change of ownership of the instance attribute shall be provided to either joined federate. In order for

ownership of the instance attribute to be granted to the specified joined federate, the following

conditions shall be true: - the specified joined federate knows about the object

instance - the specified joined federate is publishing the corresponding class attribute at the known class

of the specified object instance at that joined federate - the specified instance attribute is not owned

by the RTI (i.e., it's not a predefined attribute of a MOM object class) If one or more of the above conditions are not

met, then the interaction shall have no effect and an error shall be reported via an interaction of

class HLAmanager. HLAfederate.HLAreport. HLAreportMOMexception.">

<parameter name="HLAobjectInstance"

dataType="HLAhandle"

semantics="Handle of the object instance whose attribute

state is being changed"/>

<parameter name="HLAattribute"

dataType="HLAhandle"

semantics="Handle of the instance attribute whose stateis being changed"/>

<parameter name="HLAattributeState"

dataType="HLAownership"

semantics="New state for the attribute of the object Instance"/>

</interactionClass>

<interactionClass name="HLAsetServiceReporting"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Specify whether to report service invocations to0 or from the specified joined federate via HLAmanager.HLAfederate. HLAreport.HLAreportServiceInvocation interactions

(enable or disable servicereporting). If the specified joined federate is subscribed to the HLAmanager.HLAfederate.HLAreport. HLAreportServiceInvocation interaction, all attempts to enable service reporting for that joined federate by sending an

HLAmanager.HLAfederate.HLAadjust. HLAsetServiceReporting interaction with an

HLAreportingState parameter value of HLAtrue shall fail and be reported via the normal MOM interaction

failure means">

<parameter name="HLAreportingState"

dataType="HLAboolean"

semantics="Whether the RTI should report service invocations (default = HLAfalse)"/>

</interactionClass>

<interactionClass name="HLAsetExceptionReporting"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Specify whether the RTI shall report service invocation exceptions via

HLAmanager.HLAfederate.HLAreport. HLAreportException interactions">

<parameter name="HLAreportingState"

dataType="HLAboolean"

semantics="Whether the RTI should report exceptions (default = HLAfalse)"/>

</interactionClass>

</interactionClass>

<interactionClass name="HLArequest"

sharing="Neither"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Permit a federate to request RTI data about another federate">

<interactionClass name="HLArequestPublications"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send report interactions that contain the publication data of a joined federate. It shall result in one interaction of class HLAmanager. HLAfederate.HLAreport. HLAreportInteractionPublication and one interaction of class HLAmanager.HLAfederate. HLAreport.HLAreportObjectClassPublication for each object class published">

</interactionClass>

<interactionClass name="HLArequestSubscriptions"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send report interactions that contain the subscription data of a joined federate. It shall result in one interaction of class HLAmanager.

HLAfederate.HLAreport. HLAreportInteractionSubscription and one interaction of class HLAmanager.HLAfederate.

HLAreport.HLAreportObjectClassSubscription for each object class published">

</interactionClass>

<interactionClass
name="HLArequestObjectInstancesThatCanBeDeleted"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that contains the object instances that can be deleted at the joined federate. It shall result in one interaction of class HLAmanager.HLAfederate.HLAreport. HLAreportObjectInstancesThatCanBeDeleted">

</interactionClass>

<interactionClass name="HLArequestObjectInstancesUpdated"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that contains the object instance updating responsibility of a joined federate. It shall result in one interaction of class HLAmanager.HLAfederate.HLAreport. HLAreportObjectInstancesUpdated">

</interactionClass>

<interactionClass name="HLArequestObjectInstancesReflected"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that contains the objects instances for which a joined federate has had a Reflect Attribute Values service

invocation. It shall result in one interaction ofclass HLAmanager.HLAfederate.HLAreport.

HLAreportObjectInstancesReflected">

</interactionClass>

<interactionClass name="HLArequestUpdatesSent"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that contains the number of updates that a joined federate has generated. It shall result in one

interaction of classHLAmanager.HLAfederate.HLAreport.HLAreportUpdatesSent for each transportation type

that is used to send updates">

</interactionClass>

<interactionClass name="HLArequestInteractionsSent"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that

contains the number of interactions that a joined federate has generated. This count shall include

interactions sent with region. It shall result in one interaction of class

HLAmanager.HLAfederate.HLAreport.HLAreportInteractionsSent for each transportation

type that is used to send interactions">

</interactionClass>

<interactionClass name="HLArequestReflectionsReceived"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that

contains the number of reflections that a joined federate has received. It shall result in one

interaction of class HLAmanager.HLAfederate.HLAreport.HLAreportReflectionsReceived for each

transportation type used in receiving reflections">

</interactionClass>

<interactionClass name="HLArequestInteractionsReceived"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that

contains the number of interactions that a joined federate has received. It shall result in one

interaction of class HLAmanager.HLAfederate.HLAreport.HLAreportInteractionsReceived for each

transportation typeused in receiving interactions">

</interactionClass>

<interactionClass name="HLArequestObjectInstanceInformation"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that contains the information that a joined federate maintains on a single object instance. It shall result in one interaction of class HLAmanager.HLAfederate.HLAreport.HLAreportObjectInstanceInformation">

<parameter name="HLAobjectInstance"

dataType="HLAhandle"

semantics="Handle of the object instance for which information is being requested"/>

</interactionClass>

<interactionClass name="HLArequestSynchronizationPoints"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that contains a list of all in-progress federation synchronization points. It shall result in one interaction class HLAmanager.HLAfederate.HLAreport.HLAreportSynchronizationPoints">

</interactionClass>

<interactionClass name="HLArequestSynchronizationPointStatus"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request that the RTI send a report interaction that contains a list that includes each federate (and its

synchronization point status) that is associated with a particular synchronization point. It shall

result in one interaction of class HLAmanager.HLAfederate.HLAreport.HLAreportSynchronizationPointStaus">

</interactionClass>

</interactionClass>

<interactionClass name="HLAreport"

sharing="Neither"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Report RTI data about a joined federate. The RTI

shall send these interactions in response to interactions of class HLAmanager.HLAfederate.HLArequest.">

<interactionClass name="HLAreportObjectClassPublication"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in

response to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestPublications. It shall report

the attributes of one object class published by the joined federate. One of these interactions shall be

sent for each object class containing attributes that are published by the joined federate">

<parameter name="HLAnumberOfClasses"

dataType="HLAcount"

semantics="The number of object classes for which the joined

federate publishes attributes"/>

<parameter name="HLAobjectClass"

dataType="HLAhandle"

semantics="The object class whose publication is being reported"/>

<parameter name="HLAattributeList"

dataType="HLAhandleList"

semantics="List of handles of HLAobjectClass attributes

that the joined federate is publishing"/>

</interactionClass>

<interactionClass name="HLAreportObjectClassSubscription"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in

response to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestSubscriptions. It shall

report the attributes of one object class subscribed to by the joined federate. One of these

interactions shall be sent for each object class that is subscribed to by the joined federate. This

information shall reflect related DDM usage.">

<parameter name="HLAnumberOfClasses"

dataType="HLAcount"

semantics="The number of object classes for which the joined

federate subscribes to attributes. This information shall reflect related DDM usage."/>

<parameter name="HLAobjectClass"

dataType="HLAhandle"

semantics="The object class whose subscription is being reported"/>

<parameter name="HLAactive"

dataType="HLAboolean"

semantics="Whether the subscription is active"/>

<parameter name="HLAattributeList"

dataType="HLAhandleList"

semantics="List of handles of class attributes to which the joined federate is subscribing."/>

</interactionClass>

<interactionClass name="HLAreportInteractionPublication"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response to an interaction of class HLAmanager.HLAfederate.

HLArequest.HLArequestPublications. It shall report the interaction classes published by the joined

federate">

<parameter name="HLAinteractionClassList"

dataType="HLAhandleList"

semantics="List of interaction classes that the joined

federate is publishing"/>

</interactionClass>

<interactionClass name="HLAreportInteractionSubscription"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response

to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestSubscriptions. It shallreport the interaction classes subscribed to by the

joined federate. This information shall reflect related DDM usage.">

<parameter name="HLAinteractionClassList"

dataType="HLAinteractionSubList"

semantics="List of interaction class/subscription type

pairs. Each pair consists of the handle of an interaction class that the joined federate is

subscribed to and whether the joined federate is actively subscribing. This information shall

reflect related DDM usage."/>

</interactionClass>

<interactionClass
name="HLAreportObjectInstancesThatCanBeDeleted"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in

response to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestObject

InstancesThatCanBeDeleted. It shall report the number of object instances (by registered class of

the object instances) whose HLAprivilegeToDeleteObject attribute areowned by

the joined federate.">

<parameter name="HLAobjectInstanceCounts"

dataType="HLAobjectClassBasedCounts"

semantics="A list of object instance counts. Each object

instance count consists of an object class handle and the number of object instances of

that class"/>

</interactionClass>

<interactionClass name="HLAreportObjectInstancesUpdated"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in

response to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestObjectInstancessUpdated. It

shall report the number of object instances (by registered class of the object instances) for which

the joined federate has invoked the Update Attribute Values service.">

<parameter name="HLAobjectInstanceCounts"

dataType="HLAobjectClassBasedCounts"

semantics="List of object instance counts. Each object

instance count consists of an object class handle and the number of object instances of

that class."/>

</interactionClass>

<interactionClass name="HLAreportObjectInstancesReflected"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI inresponse to an interaction of class

HLAmanager.HLAfederate.HLArequest.HLArequestObjectInstancessReflected. It shall report the number of object instances

(by registered class of the object instances) for which the joined federate has had a Reflect

Attribute Values service invocation.">

<parameter name="HLAobjectInstanceCounts"

dataType="HLAobjectClassBasedCounts"

semantics="List of object instance counts. Each object

instance count consists of an object class handle and the number of object instances of that class."/>

</interactionClass>

<interactionClass name="HLAreportUpdatesSent"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response to an interaction of class HLAmanager.HLAfederate. HLArequest.HLArequestUpdatesSent. It shall report the number of updates sent (by registered class of the object instances of the updates) by the joined federate since the beginning of the federation execution. One interaction of this class shall be sent by the RTI for each transportation type used.">

<parameter name="HLAtransportation"

dataType="HLAtransportationName"

semantics="Transportation type used in sending updates"/>

<parameter name="HLAupdateCounts"

dataType="HLAobjectClassBasedCounts"

semantics="List of update counts. Each update count consists of an object class handle and the number of updates sent of that class"/>

</interactionClass>

<interactionClass name="HLAreportReflectionsReceived"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestReflectionsReceived. It shall report the number of reflections received (by registered class of the object instances of the reflects) by the joined federate since the beginning of the federation execution. One interaction of this class shall be sent by the RTI for each transportation type used.">

<parameter name="HLAtransportation"

dataType="HLAtransportationName"

semantics="Transportation type used in receiving reflections"/>

<parameter name="HLAreflectCounts"

dataType="HLAobjectClassBasedCounts"

semantics="List of reflection counts. Each reflection count consists of an object class handle and the number of reflections received of that class."/>

</interactionClass>

<interactionClass name="HLAreportInteractionsSent"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response to an interaction of class

HLAmanager.HLAfederate.HLArequest.HLArequestInteractionsSent. It shall report the number of interactions sent (by sent

class of the interactions) by the joined federate since the beginning of the federation execution.

This count shall include interactions sent with region. One interaction of this class shall be sent

by the RTI for each transportation type used.">

<parameter name="HLAtransportation"

dataType="HLAtransportationName"

semantics="Transportation type used in sending interactions"/>

<parameter name="HLAinteractionCounts"

dataType="HLAinteractionCounts"

semantics="List of interaction counts. Each interaction

count consists of an interaction class handle and the number of interactions of that class.

This information shall reflect related DDM usage."/>

</interactionClass>

<interactionClass name="HLAreportInteractionsReceived"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in

response to an interaction of class HLAmanager.HLAfederate.HLArequest. HLArequestInteractionsReceived. It

shall report the number of interactions received (by sent class of the interactions) by the joined

federate since the beginning of the federation execution. One interaction of this class shall be

sent by the RTI for each transportation type used.">

<parameter name="HLAtransportation"

dataType="HLAtransportationName"

semantics="Transportation type used in receiving interactions"/>

<parameter name="HLAinteractionCounts"

dataType="HLAinteractionCounts"

semantics="List of interaction counts. Each interaction count consists of an interaction class handle and the number of interactions of that class."/>

</interactionClass>

<interactionClass name="HLAreportObjectInstanceInformation"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestObjectInstance Information.

It shall report on a single object instance and portray the attributes of that object instance that

are owned by the joined federate, the registered class of the object instance, and the known class

of the object instance at the joined federate.">

<parameter name="HLAobjectInstance"

dataType="HLAhandle"

semantics="Handle of the object instance for which the interaction was sent"/>

<parameter name="HLAownedInstanceAttributeList"

dataType="HLAhandleList"

semantics="List of the handles of all instance attributes, of the object instance, owned by the joined federate"/>

<parameter name="HLAregisteredClass"

dataType="HLAhandle"

semantics="Handle of the registered class of the object instance"/>

<parameter name="HLAknownClass"

dataType="HLAhandle"

semantics="Handle of the known class of the object instance at the joined federate"/>

</interactionClass>

<interactionClass name="HLAreportException"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI when an exception occurs as the result of a service

invocation at the indicated joined federate. This interaction shall be sent only if the last

HLAmanager.HLAfederate.HLAadjust.HLAsetExceptionReporting interaction changing the HLAreportingState parameter sets the parameter to

HLAtrue, for the indicated joined federate.">

<parameter name="HLAservice"

dataType="HLAunicodeString"

semantics="Name of the service that raised the exception"/>

<parameter name="HLAexception"

dataType="HLAunicodeString"

semantics="Textual depiction of the exception"/>

</interactionClass>

<interactionClass name="HLAreportServiceInvocation"

sharing="Publish"

dimensions="Federate ServiceGroup"

transportation="HLAreliable"

order="Receive"

semantics="This interaction shall be sent by the RTI whenever an HLA service is invoked, either by the indicated joined federate or by the RTI at the indicated

joined federate, and Service Reporting is Enabled for the indicated joined federate. This interactionshall always contain the arguments supplied by the

service invoker. If the service invocation was successful, the interaction also shall contain the

value returned to the invoker (if the service returns a value); otherwise, the interaction also

shall contain an indication of the exception that shall be raised to the invoker.">

<parameter name="HLAservice"

dataType="HLAunicodeString"

semantics="Textual name of the service"/>

<parameter name="HLAsuccessIndicator"

dataType="HLAboolean"

semantics="Whether the service invocation was successful.

Exception values are returned along with HLAfalse value"/>

<parameter name="HLAsuppliedArguments"

dataType="HLAargumentList"

semantics="Textual depiction of the arguments supplied in

the service invocation"/>

<parameter name="HLAreturnedArguments"

dataType="HLAargumentList"

semantics="Textual depiction of the argument returned by the service invocation (null if the service does

not normally return a value or if HLAsuccessIndicator is HLAfalse)"/>

<parameter name="HLAexception"

dataType="HLAunicodeString"

semantics="Textual depiction of the exception raised by

this service invocation (null if HLAsuccessIndicator is HLAtrue)"/>

<parameter name="HLAserialNumber"

dataType="HLAcount"

semantics="This is a per-joined federate serial number

that shall start at zero and shall increment by 1 for each HLAmanager.HLAfederate.HLAreport

HLAreportServiceInvocation interaction that represents service invocations to or from the

respective joined federate."/>

</interactionClass>

<interactionClass name="HLAreportMOMexception"
sharing="Publish"
dimensions="Federate"
transportation="HLAreliable"
order="Receive"
semantics="The interaction shall be sent by the RTI when one the following occurs:- a MOM interaction without all the necessary parameters is sent or
- an interaction that imitates a federate's invocation of an RTI service is sent and not all of the service's pre-conditions are met.">
<parameter name="HLAservice"
dataType="HLAunicodeString"
semantics="Name of the service interaction that had a problem or raised the exception"/>
<parameter name="HLAexception"
dataType="HLAunicodeString"
semantics="Textual depiction of the problem or exception"/>
<parameter name="HLAparameterError"
dataType="HLAboolean"
semantics="HLAtrue if there was an incorrect number of
interaction parameters, HLAfalse otherwise"/>
</interactionClass>
<interactionClass name="HLAreportSynchronizationPoints"
sharing="Publish"
dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response to an interaction of class HLAmanager.HLAfederate.HLArequest.HLArequestSynchronizationPoints. It shall report the list of active synchronization points in the federation execution.">

<parameter name="HLAsynchPoints"

dataType="HLAsynchPointList"

semantics="List of the in progress federation execution synchronization points"/>

</interactionClass>

<interactionClass name="HLAreportSynchronizationPointStatus"

sharing="Publish"

dimensions="Federate"

transportation="HLAreliable"

order="Receive"

semantics="The interaction shall be sent by the RTI in response to an interaction of class HLAmanager.HLAfederate. HLArequest.HLArequestSynchronizationPointStatus.It shall report the status of a particular synchronization point. This shall be a list that includes each federate (and its synchronization status) that is associated with a particular synchronization point.">

<parameter name="HLAsynchPointName"

dataType="HLAunicodeString"

semantics="Name of a particular synchronization point"/>

<parameter name="HLAsynchPointFederates"

dataType="HLAsynchPointFederateList"

semantics="List of each federate (and its synchronization

status) associated with the particular synchronization point"/>

</interactionClass>

</interactionClass>

<interactionClass name="HLAservice"

sharing="Neither"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="The interaction class shall be acted upon by the RTI.

These interactions shall invoke HLA services on behalf of another joined federate. They shall cause

the RTI to react as if the service has invoked by that other joined federate. If exceptions arise as a result

of the use of these interactions, they shall be reported via the HLAmanager.

HLAfederate.HLAreport.HLAreportMOMexception interaction to all joined federates that subscribe to

this interaction. There are two ways an error canoccur: the sending federate does not provide all the

required arguments as parameters or the preconditions of the spoofed service are not met. Each type of error

is reported via the HLAMOMreportMOMexception. NOTE&#65533;These interactions shall have the potential to

disrupt normal federation execution and should be used with great care.">

<interactionClass name="HLAresignFederationExecution"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Cause the joined federate to resign from the federation execution. A joined federate shall be

able to send this interaction anytime.">

<parameter name="HLAresignAction"

dataType="HLAresignAction"

semantics="Action that the RTI is to take in conjunction with the resignation"/>

</interactionClass>

<interactionClass name="HLAsynchronizationPointAchieved"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Mimic the federate's report of achieving a synchronization point.">

<parameter name="HLAlabel"

dataType="HLAunicodeString"

semantics="Label associated with the synchronization point"/>

</interactionClass>

<interactionClass name="HLAfederateSaveBegun"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Mimic the federate's report of starting a save">

</interactionClass>

<interactionClass name="HLAfederateSaveComplete"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Mimic the joined federate's report of completion of a save. A joined federate shall be able to send

this interaction during a federation save.">

<parameter name="HLAsuccessIndicator"

dataType="HLAboolean"

semantics="Whether the save was successful"/>

</interactionClass>

<interactionClass name="HLAfederateRestoreComplete"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Mimic the joined federate's report of completion

of a restore. A joined federate shall be able to send this interaction during a federationrestore.">

```
                    <parameter name="HLAsuccessIndicator"
                        dataType="HLAboolean"
                        semantics="Whether the restore was successful"/>
                </interactionClass>
                <interactionClass name="HLApublishObjectClassAttributes"
                    sharing="Subscribe"
                    dimensions="NA"
                    transportation="HLAreliable"
                    order="Receive"
                    semantics="Set the joined federate's publication status of
attributes belonging to an object class">
                    <parameter name="HLAobjectClass"
                        dataType="HLAhandle"
                        semantics="Object class for which the joined federate's
publication shall change"/>
                    <parameter name="HLAattributeList"
                        dataType="HLAhandleList"
                        semantics="List of handles of attributes of
HLAobjectClass, which the federate shall now
publish"/>
                </interactionClass>
                <interactionClass name="HLAunpublishObjectClassAttributes"
                    sharing="Subscribe"
                    dimensions="NA"
                    transportation="HLAreliable"
```

order="Receive"

semantics="Cause the joined federate no longer to publish attributes of an object class">

<parameter name="HLAobjectClass"

dataType="HLAhandle"

semantics="Object class for which the joined federate's unpublication shall change"/>

<parameter name="HLAattributeList"

dataType="HLAhandleList"

semantics="List of handles of attributes of HLAobjectClass, which the joined federate shall

now unpublish"/>

</interactionClass>

<interactionClass name="HLApublishInteractionClass"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Set the joined federate's publication status of an interaction class">

<parameter name="HLAinteractionClass"

dataType="HLAhandle"

semantics="Interaction class that the joined federate shall publish"/>

</interactionClass>

<interactionClass name="HLAunpublishInteractionClass"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Cause the joined federate no longer to publish an interaction class">

<parameter name="HLAinteractionClass"

dataType="HLAhandle"

semantics="Interaction class that the joined federate shall no longer publish"/>

</interactionClass>

<interactionClass name="HLAsubscribeObjectClassAttributes"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Set the joined federate's subscription status of attributes belonging to an object class">

<parameter name="HLAobjectClass"

dataType="HLAhandle"

semantics="Object class for which the joined federate's subscription shall change"/>

<parameter name="HLAattributeList"

dataType="HLAhandleList"

semantics="List of handles of attributes of HLAobjectClass

to which the joined federate shall now subscribe"/>

```
                    <parameter name="HLAactive"
                        dataType="HLAboolean"
                        semantics="Whether the subscription is active"/>
                </interactionClass>
                <interactionClass name="HLAunsubscribeObjectClassAttributes"
                    sharing="Subscribe"
                    dimensions="NA"
                    transportation="HLAreliable"
                    order="Receive"
                    semantics="Cause the joined federate no longer to subscribe to
attributes of an object class">
                    <parameter name="HLAobjectClass"
                        dataType="HLAhandle"
                        semantics="Object class for which the joined federate's
subscription shall change"/>
                    <parameter name="HLAattributeList"
                        dataType="HLAhandleList"
                        semantics="List of handles of attributes of
HLAobjectClass
to which the joined federate shall now unsubscribe"/>
                </interactionClass>
                <interactionClass name="HLAsubscribeInteractionClass"
                    sharing="Subscribe"
                    dimensions="NA"
                    transportation="HLAreliable"
```

order="Receive"

semantics="Set the joined federate's subscription status to an interaction class.">

<parameter name="HLAinteractionClass"

dataType="HLAhandle"

semantics="Interaction class to which the federate shall subscribe"/>

<parameter name="HLAactive"

dataType="HLAboolean"

semantics="Whether the subscription is active"/>

</interactionClass>

<interactionClass name="HLAunsubscribeInteractionClass"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Cause the joined federate no longer to subscribe to an interaction class">

<parameter name="HLAinteractionClass"

dataType="HLAhandle"

semantics="Interaction class to which the joined federate

will no longer be subscribed"/>

</interactionClass>

<interactionClass name="HLAdeleteObjectInstance"

```
                    sharing="Subscribe"
                    dimensions="NA"
                    transportation="HLAreliable"
                    order="Receive"
                    semantics="Cause an object instance to be deleted from the
federation.">
                    <parameter name="HLAobjectInstance"
                        dataType="HLAhandle"
                        semantics="Handle of the object instance that is to be
deleted"/>
                    <parameter name="HLAtag"
                        dataType="HLAopaqueData"
                        semantics="Tag associated with the deletion"/>
                    <parameter name="HLAtimeStamp"
                        dataType="HLAlogicalTime"
                        semantics="Time stamp of the deletion (optional)"/>
                </interactionClass>
                <interactionClass name="HLAlocalDeleteObjectInstance"
                    sharing="Subscribe"
                    dimensions="NA"
                    transportation="HLAreliable"
                    order="Receive"
                    semantics="Inform the RTI that it shall treat the specified
object instance as if thejoined federate did not know about the object instance.">
                    <parameter name="HLAobjectInstance"
                        dataType="HLAhandle"
```

semantics="Handle of the object instance that is to be deleted"/>

</interactionClass>

<interactionClass name="HLAchangeAttributeTransportationType"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Change the transportation type used by the joined federate when sending attributes belonging to the object instance">

<parameter name="HLAobjectInstance"

dataType="HLAhandle"

semantics="Handle of the object instance whose attribute

transportation type is to be changed"/>

<parameter name="HLAattributeList"

dataType="HLAhandleList"

semantics="List of the handles of instance attributes whose transportation type is to be changed"/>

<parameter name="HLAtransportation"

dataType="HLAtransportationName"

semantics="Transportation type to be used for updatinginstance attributes in the list"/>

</interactionClass>

<interactionClass name="HLAchangeInteractionTransportationType"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Change the transportation type used by the joined federate when sending a class of interaction">

<parameter name="HLAinteractionClass"

dataType="HLAhandle"

semantics="Interaction class whose transportation type is

changed by this service invocation"/>

<parameter name="HLAtransportation"

dataType="HLAtransportationName"

semantics="Transportation type to be used for sending the interaction class"/>

</interactionClass>

<interactionClass name="HLAunconditionalAttributeOwnershipDivestiture"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Cause the ownership of attributes of an object instance to be unconditionally divested by the

joined federate">

<parameter name="HLAobjectInstance"

dataType="HLAhandle"

```
                    semantics="Handle of the object instance whose attributes'
ownership is to be divested"/>
                <parameter name="HLAattributeList"
                    dataType="HLAhandleList"
                    semantics="List of handles of instance attributes
belonging to HLAobjectInstance whose ownership is to be divested by the joined federate"/>
            </interactionClass>
            <interactionClass name="HLAenableTimeRegulation"
                sharing="Subscribe"
                dimensions="NA"
                transportation="HLAreliable"
                order="Receive"
                semantics="Cause the joined federate to begin regulating the
logical time of other joined federates">
                <parameter name="HLAlookahead"
                    dataType="HLAtimeInterval"
                    semantics="Lookahead to be used by the joined federate
        while regulating other joined federates"/>
            </interactionClass>
            <interactionClass name="HLAdisableTimeRegulation"
                sharing="Subscribe"
                dimensions="NA"
                transportation="HLAreliable"
                order="Receive"
```

semantics="Cause the joined federate to cease regulating the logical time of other joined federates">

</interactionClass>

<interactionClass name="HLAenableTimeConstrained"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Cause the logical time of the joined federate to begin being constrained by the logical times of other joined federates">

</interactionClass>

<interactionClass name="HLAdisableTimeConstrained"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Cause the logical time of the joined federate to cease being constrained by the logical times of other joined federates">

</interactionClass>

<interactionClass name="HLAtimeAdvanceRequest"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request an advance of the joined federate's logical time on behalf of the joined federate, and

release zero or more messages for delivery to the joined federate">

<parameter name="HLAtimeStamp"

dataType="HLAlogicalTime"

semantics="Time stamp requested"/>

</interactionClass>

<interactionClass name="HLAtimeAdvanceRequestAvailable"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request an advance of the joined federate's logical time, on behalf of the joined federate, and

release zero or more messages for delivery to the joined federate">

<parameter name="HLAtimeStamp"

dataType="HLAlogicalTime"

semantics="Time stamp requested"/>

</interactionClass>

<interactionClass name="HLAnextMessageRequest"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Request the logical time of the joined federate to be advanced to the time stamp of the next TSO message that shall be delivered to the joined

federate, provided that the message shall have a time stamp no greater than the logical time specified in the request, and release zero or more messages for delivery to the joined federate.">

<parameter name="HLAtimeStamp"
dataType="HLAlogicalTime"
semantics="Time stamp requested"/>
</interactionClass>
<interactionClass name="HLAnextMessageRequestAvailable"
sharing="Subscribe"
dimensions="NA"
transportation="HLAreliable"
order="Receive"
semantics="Request the logical time of the joined federate to be advanced to the time stamp of the next TSO
message that shall be delivered to the joined federate, provided that the message shall have a time stamp no greater than the logical time specified in the request, and release zero or more messages for delivery to the joined federate.">
<parameter name="HLAtimeStamp"
dataType="HLAlogicalTime"
semantics="Time stamp requested"/>
</interactionClass>
<interactionClass name="HLAflushQueueRequest"
sharing="Subscribe"
dimensions="NA"
transportation="HLAreliable"

order="Receive"

semantics="Request the logical time of the joined federate to be advanced as far as possible, provided that the

time stamp is less than or equal to the logical time specified in the request. All TSO and RO

messages shall be delivered to the joined federate.">

<parameter name="HLAtimeStamp"

dataType="HLAlogicalTime"

semantics="Time stamp requested"/>

</interactionClass>

<interactionClass name="HLAenableAsynchronousDelivery"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Cause the RTI to deliver receive-order messages to the joined federate at any time, even if the joined

federate is time-constrained.">

</interactionClass>

<interactionClass name="HLAdisableAsyncrhonousDelivery"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="When the joined federate is time-constrained,

cause the RTI to deliver receive-order messages to

the joined federate only when its time manager

```
                state is Time Advancing.">
            </interactionClass>
            <interactionClass name="HLAmodifyLookahead"
                sharing="Subscribe"
                dimensions="NA"
                transportation="HLAreliable"
                order="Receive"
                semantics="Change the lookahead value used by the joined
federate">
                <parameter name="HLAlookahead"
                    dataType="HLAtimeInterval"
                    semantics="New value for lookahead"/>
            </interactionClass>
            <interactionClass name="HLAchangeAttributeOrderType"
                sharing="Subscribe"
                dimensions="NA"
                transportation="HLAreliable"
                order="Receive"
                semantics="Change the ordering type used by the joined
                federate when sending attributes belonging to the
                object instance">
                <parameter name="HLAobjectInstance"
                    dataType="HLAhandle"
                    semantics="Handle of the object instance whose
attribute order type is to be changed"/>
                <parameter name="HLAattributeList"
```

dataType="HLAhandleList"

semantics="List of the handles of instance attributes whose order type is to be changed"/>

<parameter name="HLAsendOrder"

dataType="HLAorderType"

semantics="Order type to be used for sending the instance attribute list"/>

</interactionClass>

<interactionClass name="HLAchangeInteractionOrderType"

sharing="Subscribe"

dimensions="NA"

transportation="HLAreliable"

order="Receive"

semantics="Change the order type used by the joined

federate when sending a class of interaction">

<parameter name="HLAinteractionClass"

dataType="HLAhandle"

semantics="Interaction class whose order type is changed by

this service invocation"/>

<parameter name="HLAsendOrder"

dataType="HLAorderType"

semantics="Order type to be used for sending the interaction class"/>

</interactionClass>

</interactionClass>

```
</interactionClass>
<interactionClass name="HLAfederation"
        sharing="Neither"
        dimensions="NA"
        transportation="HLAreliable"
        order="Receive"
        semantics="Root class of MOM interactions that deal with a specific
        federation execution.">
        <interactionClass name="HLAadjust"
                sharing="Neither"
                dimensions="NA"
                transportation="HLAreliable"
                order="Receive"
                semantics="Permit a federate to adjust the RTI state variables
                associated with a federation execution.">
                        <interactionClass name="HLAsetSwitches"
                                sharing="Subscribe"
                                dimensions="NA"
                                transportation="HLAreliable"
                                order="Receive"
                                semantics="Set the values of several HLA switches.">
                                <parameter name="HLAautoProvide"
                                        dataType="HLAswitch"
                                        semantics="Set the federation-wide Auto
Provide Switch to
```

```
                                    the provided value (this parameter is required
                                    only when a change of this switch value is
                                    needed)."/>
                            <parameter name="HLAconveyRegionDesignatorSets"
                                    dataType="HLAswitch"
                                    semantics="Set the federation-wide Convey
Region Designator Sets Switch to the provided value
                                    (this parameter is required only when a change
of this switch value is needed)."/>
                        </interactionClass>
                    </interactionClass>
                </interactionClass>
            </interactionClass>
        </interactionClass>
    </interactions>
    <dimensions>
        <dimension name="Federate"
                dataType="HLAfederateHandle"
                upperBoundNotes="MOM1"
                normalization="Normalize Federate Handle service"
                value="Excluded"/>
        <dimension name="ServiceGroup"
                dataType="HLAserviceGroupName"
                upperBound="7"
                normalization="Normalize Service Group service"
                value="Excluded"/>
```

```
</dimensions>
<time>
        <timeStamp
                dataType="NA"/>
        <lookahead
                dataType="NA"/>
</time>
<transportations>
        <transportation
                name="HLAreliable"
                description="Provide reliable delivery of data in the sense that TCP/IP
                delivers its data reliably"/>
        <transportation
                name="HLAbestEffort"
                description="Make an effort to deliver data in the sense that UDP
                provides best-effort delivery"/>
</transportations>
<switches
        autoProvide="Enabled"
        conveyRegionDesignatorSets="Enabled"
        attributeScopeAdvisory="Enabled"
        attributeRelevanceAdvisory="Enabled"
        objectClassRelevanceAdvisory="Enabled"
        interactionRelevanceAdvisory="Enabled"
        serviceReporting="Disabled"/>
```

<dataTypes>
<basicDataRepresentations>
<basicData name="HLAinteger16BE"
size="16"
interpretation="Integer in the range [-2^15, 2^15 – 1]"
endian="Big"
encoding="16-bit two's complement signed integer. The most
significant bit contains the sign."/>
<basicData name="HLAinteger32BE"
size="32"
interpretation="Integer in the range [-2^31, 2^31 – 1]"
endian="Big"
encoding="32-bit two's complement signed integer. The most
significant bit contains the sign."/>
<basicData name="HLAinteger64BE"
size="64"
interpretation="Integer in the range [-2^63, 2^63 – 1]"
endian="Big"
encoding="64-bit two's complement signed integer first. The most
significant bit contains the sign."/>
<basicData name="HLAfloat32BE"
size="32"
interpretation="Single-precision floating point number"
endian="Big"
encoding="32-bit IEEE normalized single-precision format. See IEEE

```
        Std 754-1985"/>
    <basicData name="HLAfloat64BE"
        size="64"
        interpretation="Double-precision floating point number"
        endian="Big"
        encoding="64-bit IEEE normalized double-precision format. See IEEE
        Std 754-1985"/>
    <basicData name="HLAoctetPairBE"
        size="16"
        interpretation="16-bit value"
        endian="Big"
        encoding="Assumed to be portable among hardware devices."/>
    <basicData name="HLAinteger16LE"
        size="16"
        interpretation="Integer in the range [-2^15, 2^15 – 1]"
        endian="Little"
        encoding="16-bit two's complement signed integer. The most
        significant bit contains the sign."/>
    <basicData name="HLAinteger32LE"
        size="32"
        interpretation="Integer in the range [-2^31, 2^31 – 1]"
        endian="Little"
        encoding="32-bit two's complement signed integer. The most
        significant bit contains the sign."/>
    <basicData name="HLAinteger64LE"
```

size="64"

interpretation="Integer in the range [-2^63, 2^63 – 1]"

endian="Little"

encoding="64-bit two's complement signed integer first. The most

significant bit contains the sign."/>

<basicData name="HLAfloat32LE"

size="32"

interpretation="Single-precision floating point number"

endian="Little"

encoding="32-bit IEEE normalized single-precision format. See IEEE

Std 754-1985"/>

<basicData name="HLAfloat64LE"

size="64"

interpretation="Double-precision floating point number"

endian="Little"

encoding="64-bit IEEE normalized double-precision format. See IEEE

Std 754-1985"/>

<basicData name="HLAoctetPairLE"

size="16"

interpretation="16-bit value"

endian="Little"

encoding="Assumed to be portable among hardware devices."/>

<basicData name="HLAOctet"

size="8"

interpretation="8-bit value"

```
                endian="Big"
                encoding="Assumed to be portable among hardware devices."/>
    </basicDataRepresentations>
    <simpleDataTypes>
        <simpleData name="HLAASCIIchar"
                representation="HLAoctet"
                semantics="Standard ASCII character (see ANSI Std X3.4-1986)"/>
        <simpleData name="HLAunicodeChar"
                representation="HLAoctetPairBE"
                units="NA"
                resolution="NA"
                accuracy="NA"
                semantics="Unicode UTF-16 character (see The Unicode Standard,
                Version 3.0)"/>
        <simpleData name="HLAbyte"
                representation="HLAoctet"
                semantics="Uninterpreted 8-bit byte"/>
        <simpleData name="HLAcount"
                representation="HLAinteger32BE"/>
        <simpleData name="HLAseconds"
                representation="HLAinteger32BE"
                units="seconds"/>
        <simpleData name="HLAmsec"
                representation="HLAinteger32BE"
                units="milliseconds"/>
```

```
            <simpleData name="HLAfederateHandle"
                representation="HLAinteger32BE"
                semantics="The type of the argument to Normalize Federate Handle
                service. This is a pointer to an RTI-defined programming
                language object, not an integer 32"/>
            <simpleData name="UserDataType"
                representation="HLAinteger32BE"
                units="furlongs"/>
        </simpleDataTypes>
        <enumeratedDataTypes>
            <enumeratedData name="HLAboolean"
                representation="HLAinteger32BE"
                semantics="Standard boolean type">
            <enumerator name="HLAfalse" values="0"/>
            <enumerator name="HLAtrue" values="1"/>
        </enumeratedData>
        <enumeratedData name="HLAfederateState"
            representation="HLAinteger32BE"
            semantics="State of the federate">
            <enumerator name="ActiveFederate" values="1"/>
            <enumerator name="FederateSaveInProgress" values="3"/><enumerator
name="FederateRestoreInProgress" values="5"/>
        </enumeratedData>
        <enumeratedData name="HLAtimeState"
            representation="HLAinteger32BE"
```

```
            semantics="State of time advancement">
            <enumerator name="TimeGranted" values="0"/>
            <enumerator name="TimeAdvancing" values="1"/>
        </enumeratedData>
        <enumeratedData name="HLAownership"
            representation="HLAinteger32BE">
            <enumerator name="Unowned" values="0"/>
            <enumerator name="Owned" values="1"/>
        </enumeratedData>
        <enumeratedData name="HLAresignAction"
            representation="HLAinteger32BE"
            semantics="Action to be performed by RTI in conjunction with
            resignation">
            <enumerator name="DivestOwnership" values="1"/>
            <enumerator name="DeleteObjectInstances" values="2"/>
            <enumerator name="CancelPendingAcquisitions" values="3"/>
            <enumerator name="DeleteObjectInstancesThenDivestOwnership"
            values="4"/>
            <enumerator
name="CancelPendingAcquisitionsThenDeleteObjectInstancesThenDivestOwnership"
values="5"/>
            <enumerator name="NoAction" values="6"/>
        </enumeratedData>
        <enumeratedData name="HLAorderType"
            representation="HLAinteger32BE"
            semantics="Order type to be used for sending attributes or
```

```
        interactions">
        <enumerator name="Receive" values="0"/>
        <enumerator name="TimeStamp" values="1"/>
    </enumeratedData >
    <enumeratedData name="HLAswitch"
        representation="HLAinteger32BE">
        <enumerator name="Enabled" values="1"/>
        <enumerator name="Disabled" values="0"/>
    </enumeratedData>
    <enumeratedData name="HLAsynchPointStatus"
        representation="HLAinteger32BE"
        semantics="Joined federate synchronization point status">
        <enumerator name="NoActivity" values="0"/>
        <enumerator name="AttemptingToRegisterSynchPoint" values="1"/>
        <enumerator name="MovingToSynchPoint" values="2"/>
        <enumerator name="WaitingForRestOfFederation" values="3"/>
    </enumeratedData>
    <enumeratedData name="HLAserviceGroupName"
        representation="HLAinteger32BE"
        semantics="Service group identifier">
        <enumerator name="FederationManagement" values="0"/>
        <enumerator name="DeclarationManagement" values="1"/>
        <enumerator name="ObjectManagement" values="2"/>
        <enumerator name="OwnershipManagement" values="3"/>
        <enumerator name="TimeManagement" values="4"/>
```

```
            <enumerator name="DataDistributionManagement" values="5"/>
            <enumerator name="SupportServices" values="6"/>
        </enumeratedData>
</enumeratedDataTypes>
<arrayDataTypes>
        <arrayData name="HLAASCIIstring"
            dataType="HLAASCIIchar"
            cardinality="Dynamic"
            encoding="HLAvariableArray"
            semantics="ASCII String representation"/>
        <arrayData name="HLAunicodeString"
            dataType="HLAunicodeChar"
            cardinality="Dynamic"
            encoding="HLAvariableArray"
            semantics="Unicode string representation"/>
        <arrayData name="HLAopaqueData"
            dataType="HLAbyte"
            cardinality="Dynamic"
            encoding="HLAvariableArray"
            semantics="Uninterpreted sequence of bytes"/>
        <arrayData name="HLAhandle"
            dataType="HLAbyte"
            cardinality="Dynamic"
            encoding="HLAvariableArray"
            semantics="Encoded value of a handle. The encoding is based on the
```

type of handle"/>

<arrayData name="HLAtransportationName"

dataType="HLAunicodeChar"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="String whose legal value shall be a name from any row in the OMT transportation table (IEEE Std 1516.2-2000"/>

<arrayData name="HLAlogicalTime"

dataType="HLAbyte"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="An encoded logical time. An empty array shall indicate that the values is not defined"/>

<arrayData name="HLAtimeInterval"

dataType="HLAbyte"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="An encoded logical time interval. An empty array shall indicate that the values is not defined"/>

<arrayData name="HLAhandleList"

dataType="HLAhandle"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="List of encoded handles"/>

<arrayData name="HLAinteractionSubList"

dataType="HLAinteractionSubscription"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="List of interaction subscription indicators"/>

<arrayData name="HLAargumentList"

dataType="HLAunicodeString"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="List of arguments"/>

<arrayData name="HLAobjectClassBasedCounts"

dataType="HLAobjectClassBasedCount"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="Counts of various items based on object class"/><arrayData name="HLAinteractionCounts"

dataType="HLAinteractionCount"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="List of interaction counts"/>

<arrayData name="HLAsynchPointList"

dataType="HLAunicodeString"

cardinality="Dynamic"

encoding="HLAvariableArray"

semantics="List of names of synchronization points"/>

<arrayData name="HLAsynchPointFederateList"

```
                dataType="HLAsynchPointFederate"
                cardinality="Dynamic"
                encoding="HLAvariableArray"
                semantics="List of joined federates and the synchronization status of
                each"/>
</arrayDataTypes>
<fixedRecordDataTypes>
        <fixedRecordData name="HLAinteractionSubscription"
                encoding="HLAfixedRecord"
                semantics="Interaction subscription information">
                <field name="HLAinteractionClass"
                        dataType="HLAhandle"
                        semantics="Encoded interaction class handle"/>
                <field name="HLAactive"
                        dataType="HLAboolean"
                        semantics="Whether subscription is active (HLAtrue=active)"/>
        </fixedRecordData>
        <fixedRecordData name="HLAobjectClassBasedCount"
                encoding="HLAfixedRecord"
                semantics="Object class and count of associated items">
                <field name="HLAobjectClass"
                        dataType="HLAhandle"
                        semantics="Encoded object class handle"/>
                <field name="HLAcount"
                        dataType="HLAcount"
```

```
                semantics="Number of items"/>
        </fixedRecordData>
        <fixedRecordData name="HLAinteractionCount"
                encoding="HLAfixedRecord"
                semantics="Count of interactions of a class">
                <field name="HLAinteractionClass"
                        dataType="HLAhandle"
                        semantics="Encoded interaction class handle"/>
                <field name="HLAinteractionCount"
                        dataType="HLAcount"
                        semantics="Number of interactions"/>
        </fixedRecordData>
        <fixedRecordData name="HLAsynchPointFederate"
                encoding="HLAfixedRecord"
                semantics="A particular joined federate and its synchronization point
                status">
                <field name="HLAfederate"
                        dataType="HLAhandle"
                        semantics="Encoded joined federate handle"/>
                <field name="HLAfederateSynchStatus"
                        dataType="HLAsynchPointStatus"
                        semantics="Synchronization status of the particular joined\
                        federate"/>
        </fixedRecordData>
</fixedRecordDataTypes>
```

```
</dataTypes>
<notes>
        <note
                name="MOM1"
                semantics="The value of the Dimension Upper Bound entry for the Federate
                dimension is RTI implementation dependent."/>
</notes>
</objectModel>
```

La simulación distribuida de cadenas de suministro tiene la gran ventaja de preservar la independencia de los miembros de la cadena, pudiendo reutilizar simuladores existentes sin necesidad de crear uno nuevo. Sin embargo, el problema que emerge en este tipo de simulación es la necesidad de acordar el conjunto de objetos, eventos, interacciones y métricas, que deben ser entendidas por todos los participantes para lograr con éxito un resultado valioso para los mismos. En este trabajo se presenta un marco conceptual basado en una red de ontologías, que da soporte a las tareas de modelado y composición de la simulación distribuida de cadenas de suministro para garantizar la interoperabilidad semántica de sus miembros. Se utiliza el estándar HLA (High Level Architecture) como herramienta de construcción de una simulación distribuida.

Juan Leonardo Sarli

Ingeniero en Sistemas de Información graduado en 2014.
Becario Interno Doctoral del Conicet 04/2014 a 03/2019.
Auxiliar de Trabajos Practicos de Primera en UTN FRSF desde 09/2016 hasta 03/2018.
Jefe de Trabajos Prácticos en UTN FRSF desde 04/2018 hasta la actualidad.
Docente Investigador UTN FRSF Categoría E desde 04/2018 hasta la actualidad.
Doctor en Ingeniería mención Sistemas de Información graduado en 03/2019.
Becario Interno Posdoctoral del Conicet 04/2019 a 03/2021.
Participación en numerosos proyectos de investigación de la UTN FRSF y del Conicet.
Una publicación en revista internacional indexada:
2018 DOI: 10.17013/risti.30.34-50.
Varias publicaciones en congresos internacionales:
2016 DOI: 10.1109/WSC.2016.7822175
2017 DOI: 10.1109/WSC.2017.8248242
2018 "An Interoperability Model for Collaborative Development of Distributed Supply Chain Simulations". En Proceedings of the 2018 Winter Simulation Conference.
Varias publicaciones en congresos nacionales.

Tesis Doctoral

UTN SANTA FE

edUTecNe
L U A Libro Universitario Argentino
CiN REUN Red de Editoriales de Universidades Nacionales de la Argentina