La presente publicación corresponde a una tesis presentada para cumplir con los requisitos exigidos por la Universidad Tecnológica Nacional – Facultad Regional Santa Fe para obtener el grado Académico de Doctor en Ingeniería con Mención en Sistemas de Información

Tesis Doctoral

Método para el Modelado y Especificación de Procesos de Negocio Colaborativos

por Pablo David Villarreal

Director: Dr. Omar CHIOTTI

Co-Director: Dr. Enrique SALOMONE

Villarreal, Pablo David

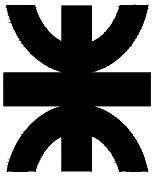
Método para el modelado y especificación de procesos de negocios colaborativos / Pablo David Villareal ; dirigido por Omar Chiotti y Enrique Salomone. - 1a ed. - Santa Fe : el autor, 2005.

ISBN 987-43-9709-8

1. Empresas de Negocios. I. Chiotti, Omar, dir. II. Salomone, Enrique, dir. III. Título

CDD 658.1

A Judith y a mis padres



Universidad Tecnológica Nacional

Facultad Regional Santa Fe

Doctorado en Ingeniería

Mención en Sistemas de Información

Método para el Modelado y Especificación de Procesos de Negocio Colaborativos

por Pablo David Villarreal

Director	Co-Director
Dr. Omar Chiotti	Dr. Enrique Salomone

Jurados de Tesis:

Dr. Horacio Leone

Dr. Luis Olsina

Dr. Roberto Uzal

- Junio 2005 -

Índice

Lista de Figuras

Lista de Tablas

Resumen

El modelado y la especificación de los procesos de negocio colaborativos constituyen un desafío importante para que las empresas puedan establecer colaboraciones Business-to-Business. El objetivo de esta tesis es definir y proponer un método robusto y sistemático que de soporte al desarrollo de procesos colaborativos, desde el modelado de los mismos, hasta la especificación de dichos procesos y de las interfaces de las empresas que conforman el sistema de información Business-to-Business, en una tecnología particular. El método está basado en la filosofía del desarrollo conducido por modelos y fundamentalmente, en los principios y guías de la Arquitectura Conducida por Modelos. Como parte de este método, se propone el lenguaje de modelado UP-ColBPIP, para dar soporte al análisis y diseño de los procesos colaborativos. El lenguaje está basado en el concepto de protocolos de interacción para definir procesos colaborativos independientes de la tecnología. Además, se propone un método y una herramienta para definir y ejecutar transformaciones de modelos de procesos colaborativos en especificaciones basadas en un estándar Business-to-Business. Finalmente, se definen las transformaciones que posibilitan generar, a partir de modelos definidos con UP-ColBPIP, soluciones tecnologías con dos tipos diferentes de estándares Business-to-Business.

Prólogo

Actualmente, las empresas se están enfocando en establecer relaciones de colaboración con sus proveedores y clientes (socios de negocio), para mejorar la competitividad de las cadenas de suministro. *Colaboración* significa que las empresas alcanzan metas comunes, coordinan sus acciones e intercambian información, a través de la definición y ejecución en forma conjunta de *procesos de negocio colaborativos*.

Existen diferentes modelos de negocio para llevar a cabo la colaboración, denominados Modelos de Colaboración. Estos modelos definen las reglas generales y parámetros a ser tenidos en cuenta en una relación entre empresas, y son soportados a través de relaciones de colaboración Business-to-Business (B2B), denominadas *colaboraciones B2B*. Estas relaciones, además del modelo, comprenden la integración de los sistemas de los socios, a través de un sistema de información B2B, del cual existe dos tipos: sistemas e-Marketplaces y sistemas de información Peer-to-Peer (P2P).

En esta tesis se discuten los costos y beneficios de diferentes tipos de colaboraciones B2B. De acuerdo a los beneficios identificados, se puede inferir que a través de modelos de colaboración descentralizados, soportados por colaboraciones B2B basadas en sistemas de información P2P, las empresas podrán alcanzar altos niveles de integración y autonomía, gestión descentralizada de los procesos colaborativos, y personalización de estos procesos con cada socio. Como contraparte de estos beneficios, actualmente el desarrollo de estas soluciones es costoso. Esto trae aparejado nuevos desafíos, tanto desde el punto de vista de la gestión de los procesos colaborativos como desde el punto de vista de la interoperabilidad de los sistemas de los socios, que hacen el desarrollo más complejo e incrementan los tiempos para implementar dichas colaboraciones B2B. Los socios necesitan reducir estos tiempos, con el objetivo de aprovechar las oportunidades de negocio que detectan.

Para abordar estas dificultades, resulta necesario considerar dos niveles de abstracción en el desarrollo de colaboraciones B2B: el nivel de negocio y el nivel tecnológico. El nivel de negocio (el dominio del problema) refiere al diseño de los procesos colaborativos y las interfaces de negocio de los socios, sin considerar la tecnología de implementación. Los procesos colaborativos son definidos de acuerdo al modelo de colaboración a utilizar. El nivel tecnológico (el dominio de la solución)

refiere a la definición de protocolos B2B basados en un estándar particular, los cuales implican la especificación de: los procesos colaborativos en un lenguaje ejecutable de procesos de negocio, y las interfaces de los socios correspondientes al sistema de información B2B. A través de estas especificaciones es posible dar soporte a la ejecución de los procesos colaborativos. En consecuencia, las definiciones en ambos niveles deben tener una mutua correspondencia.

El principal concepto en una colaboración B2B es el de proceso de negocio colaborativo independiente de la tecnología. El comportamiento de la colaboración y los roles que cada socio debe desempeñar quedan explicitados en dichos procesos. Además, la generación automática de las especificaciones de los protocolos B2B en un estándar, a partir de los modelos de procesos colaborativos independientes de la tecnología, es de suma importancia para disminuir la complejidad, el tiempo y los costos en la construcción de colaboraciones B2B.

Por lo tanto, en esta tesis se propone un método para dar soporte a las diferentes etapas del desarrollo de procesos colaborativos, en ambos niveles de una relación de colaboración B2B. El método, basado en los principios establecidos por la Arquitectura Conducida por Modelos (Model-Driven Architecture), explota los beneficios del desarrollo conducido por modelos para el dominio de procesos colaborativos y sistemas de información B2B. De esta manera se provee los lineamientos y el marco conceptual que permiten separar los modelos de procesos colaborativos de las soluciones tecnológicas, y generar estas últimas a partir de los primeros. El propósito es utilizar como principales productos de diseño, los modelos de procesos colaborativos independientes de la tecnología. En este método, dos tareas fundamentales deben ser soportadas:

- *Modelado Conceptual de Procesos de Negocio Colaborativos*.
- *Generación Automática de Especificaciones de los Procesos Colaborativos y de las Interfaces de los Componentes de los Socios en un estándar B2B*.

Las características de las colaboraciones B2B, a las que se está apuntando en esta tesis, imponen ciertos requerimientos que deben ser satisfechos en un modelo conceptual de procesos colaborativos. Actualmente existe una carencia de lenguajes que permitan modelar procesos colaborativos independientes de la tecnología que satisfagan dichos requerimientos.

Por lo tanto, en esta tesis se propone *UP-ColBPIP*, un lenguaje de modelado de procesos colaborativos independientes de la tecnología, que provee abstracciones conceptuales adecuadas para el dominio de las relaciones de colaboración B2B. La base teórica del lenguaje está sustentada por el uso del formalismo de protocolos de interacción para representar procesos colaborativos, junto con la aplicación de los principios de la teoría de actos de comunicación y la teoría LAP (Language/Action Perspective). A través de la definición de protocolos de interacción, los diseñadores se enfocan en modelar no sólo el intercambio de información, sino también los aspectos de comunicación requeridos en los procesos colaborativos. De esta manera, puede ser considerado como un tipo de lenguaje de procesos colaborativos orientado a la comunicación. El lenguaje UP-ColBPIP ha sido definido como un perfil UML, basado en UML2. Para derivar el perfil UML, se ha definido un metamodelo del lenguaje, a través del cual se describe la semántica de los elementos conceptuales del lenguaje, las restricciones de cada uno de ellos y sus relaciones.

Con respecto a la generación automática de la solución tecnológica, es necesario llevar a cabo transformaciones de modelos de procesos colaborativos definidos con UP-ColBPIP, a modelos basados en un estándar B2B, y generar a partir de estos últimos el código XML correspondiente a las especificaciones de los procesos colaborativos y las interfaces de los sistemas de los socios en el estándar B2B. Para hacer realidad dichas transformaciones, se identifican los requerimientos particulares de las mismas y se propone un *Método de Transformación de Modelos* que satisface estos requerimientos. Como parte del método se ha propuesto: el *Lenguaje de Transformación de Modelos*, el *Lenguaje de Modelado de Reglas* y las *Reglas de Producción de Código XML*. Los lenguajes y técnicas propuestas en este método de transformación de modelos están acoplados a los principios establecidos en MDA.

Por otra parte se ha trabajado en el desarrollo de una herramienta prototipo que implementa el lenguaje de transformación de modelos, y permite llevar a cabo la definición y ejecución de transformaciones de modelos.

Finalmente, para validar el lenguaje UP-ColBPIP, y como parte del método de desarrollo conducido por modelos para procesos colaborativos, se definen las transformaciones que permiten generar, a partir de modelos de procesos colaborativos definidos con UP-ColBIP, soluciones tecnológicas utilizando dos tipos diferentes de

estándares B2B: ebXML, un estándar basado en transacciones de negocio; y BPEL, un estándar basado en composición de servicios web.

Los resultados parciales del trabajo realizado en esta tesis han sido divulgados a través de las siguientes publicaciones:

- Villarreal P., Salomone E., Chiotti O. "Modeling and Specification of Collaborative Business Processes with a MDA Approach and a UML Profile". Capítulo que ha pasado la primer etapa de revisión para ser incluido en el libro *Enterprise Modeling and Computing with UML* (Editor: Peter Rittgen), a ser publicado por Idea Group, Inc.
- Villarreal P., Salomone E., Chiotti O. "Applying Model-Driven Development to Collaborative Business Processes." *Proceedings 8º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS´05)*. Valparaiso, Chile, 2005.
- Villarreal P., Salomone E., Chiotti O. "A UML Profile for Modeling Collaborative Business Processes based on Interaction Protocols". *Argentine Symposium on Information Systems (ASIS 2004), 33 JAIIO*, Argentina, 2004.
- Villarreal P., Caliusco M.L., Galli M.R., Salomone E., Chiotti O. "Decentralized Process Management for Inter-Enterprise Collaboration". Libro: *Emerging issues on Supply Chain Management*. Editor: *B.S. Sahay. MacMillan India Limited,* New Delhi, India, 2004.
- Caliusco M.L., Villarreal P., Arredondo F., Zanel C., Zucchini D., Chiotti O., Galli M.R. "Decentralized Management Model of a Partner-to-Partner Collaborative Relationship". *Proceedings of The 2nd World POM Conference on POM and 15th annual POM Conference*, Cancún, México, 2004.
- Villarreal, P., Salomone, E., Chiotti, O. "B2B Relationships: Defining Public Business Processes using Interaction Protocols". *Journal of the Chilean Society of Computer Science, Special issue on the Best Papers of the JCC 2003, Vol. 4(1)*, 2003.
- Villarreal P., Caliusco M.L., Galli M.R., Salomone E., Chiotti O. "Decentralized Process Management for Inter-Enterprise Collaboration".

Vision, Special Issue on Supply Chain Management, Vol 7, pp. 69-79. Editor: B. S. Sahay, Management Development Institute, Gurgaon, India, 2003.

- Villarreal P., Salomone E., Chiotti O. "Managing Public Business Processes in B2B Relationships using B2B Interaction Protocols". *XXIX Conferencia Latinoamérica de Informática (CLEI 2003)*, Bolivia, 2003.
- Villarreal P.D., Caliusco M.L., Zucchini D., Arredondo F., Zanel C., Galli M.R., Chiotti O. "Integrated Production Planning and Control in a Collaborative Partner-to-Partner Relationship". Libro: *Managing e-Business in the 21st Century.* Editores: *S. Sharma & J. Gupta. Heidelberg Press, pp. 91-110*, Australia, 2003.
- Villarreal P., Salomone E., Chiotti O. "B2B Relationships: Defining Public Business Processes using Interaction Protocols". *Jornadas Chilenas de Computación (JCC 2003)*, 2003.
- Villarreal P., Salomone E., Chiotti O. "Modeling Public Business Processes with Interaction Protocols in B2B Relationships". *Actas del IX Congreso Argentino de Ciencias de la Computación (CACIC 2003)*, La Plata, Argentina, 2003.
- Caliusco M.L., Villarreal P., Galli M.R., Chiotti O., Salomone E. "Model and Architecture to Production Inter-Enterprise Collaboration Management". *Proceedings de la XVII Conferencia Latinoamericana de Informática (CLEI 2001)*, Venezuela, Septiembre de 2001.
- Villarreal P., Caliusco M.L., Salomone E., Galli M.R., Chiotti O. "Support for Collaboration and Negotiation in SC Management". *IV Simposio de Administración de la Producción, Logística y Operaciones Internacionales. Conferencia de la Sociedad de Administración de Producción y Operaciones (SIMPOI 2001/POMS)*, Guarujá, Brasil, 2001.

Reconocimientos

Quiero agradecer en primer lugar a mi director de tesis, el Dr. Omar Chiotti, por sus invalorables consejos, por su guía y apoyo constante a mi tarea, y por haberme brindado todo lo que estaba a su alcance para que pueda iniciarme en el camino de la investigación.

Mi agradecimiento a mi codirector, el Dr. Enrique Salomone, por las discusiones enriquecedoras, guía y apoyo constante a mi tarea.

A la Universidad Tecnológica Nacional por haber hecho posible la realización de esta tesis a través del soporte económico facilitado con una Beca de Posgrado, y a la Facultad Regional Santa Fe por el espacio y material brindado.

A las instituciones que brindaron apoyo económico para la realización de este trabajo a través de subsidios a los siguientes proyectos:

- "Soporte para el Diseño de Sistemas de Información Business-to-Business" Ente financiador: Universidad Tecnológica Nacional – Facultad Regional Santa Fe.
- "Information System to Support Collaborative Business Processes of a P2P Model to Manage the Supply Chain". Ente financiador: Agencia Nacional de Promoción Científica y Técnica.
- "Sistema de Información Soporte de Procesos de Negocio Colaborativos" Ente financiador: Universidad Tecnológica Nacional – Facultad Regional Santa Fe.
- "Herramientas Computacionales Avanzadas para la Gestión de la Cadena de Suministro en una relación Business-to-Business". Ente financiador: Universidad Tecnológica Nacional – Facultad Regional Santa Fe.

A cada uno de los integrantes del grupo de investigación GIDSATD, por su constante aliento y por haberme brindado un cálido ambiente de trabajo para el desarrollo de esta tesis. A Georgina, por alentarme a finalizar esta tesis. A Lucas, que con su esfuerzo y dedicación hizo posible que pudiéramos contar con una herramienta para transformaciones de modelos; y a Mariela, por haberme ayudado en el tedioso trabajo de corrección.

A Laura, por compartir discusiones que fueron la fuente de inspiración de varios trabajos en conjunto y principalmente de nuestros trabajos de tesis.

Quiero agradecer, muy especialmente, a mis padres por su apoyo constante y porque gracias a sus esfuerzos de años de trabajo, han posibilitado que halla llegado a esta instancia.

Por último y muy especialmente, quiero agradecer a mi esposa Judith, por acompañar mi esfuerzo con cariño, comprensión y apoyo, y gracias a su aliento constante me ha brindado el soporte fundamental para la realización de este trabajo.

Capítulo

1

INTRODUCCIÓN

Este capítulo describe brevemente el contexto en el que se enmarca esta tesis, los problemas a resolver y los objetivos planteados en este trabajo de investigación, una breve mención de las principales contribuciones, y la organización general de la tesis.

1.1. El Contexto

Durante las últimas décadas, las empresas se han enfrentado a un mercado global y altamente competitivo. Esto ha llevado a las mismas a adoptar nuevas filosofías de gestión basadas en el establecimiento de relaciones más estrechas con sus clientes y proveedores, desde filosofías como JIT (Just-In-Time), hasta el concepto de integración de la cadena de suministro (Sahay y Maini, 2004).

Existen diferentes niveles de integración en la cadena de suministro (Grieger, 2003a). En el nivel más básico, las empresas sólo intercambian información hacia un solo sentido (generalmente desde los clientes a los proveedores) con el objetivo de disminuir algunas incertidumbres y mejorar sus procesos de negocio internos. Este tipo de integración es el más primitivo e implica una simple cooperación entre las empresas.

El nivel más alto de integración, conduce al concepto de *colaboración* entre empresas. Dicho concepto implica una integración mayor, que va más allá del intercambio de información, en donde las empresas persiguen metas comunes. *Colaboración* significa que las empresas establecen relaciones de negocio, en donde los socios de negocio pueden mejorar sus beneficios y reducir sus costos a través de la definición y ejecución en forma conjunta de *procesos de negocio colaborativos* (Liu y Kumar, 2003) (Mentzer y otros, 2000). Estos también se conocen como *procesos de negocio públicos, compartidos, entre empresas o entre organizaciones*. Los mismos se extienden a través de dos o más empresas y permiten coordinar las actividades de cada una, para alcanzar una meta común sin dejar de lado sus intereses particulares.

De esta manera, las empresas se están enfocando en establecer relaciones de colaboración con sus proveedores y clientes, con el objetivo de disminuir sus costos y mejorar sus rendimientos internos, permitiendo de este modo mejorar el rendimiento de la cadena de suministro global.

Diferentes modelos de negocio han sido propuestos para llevar a cabo la colaboración entre empresas. Dichos modelos se conocen como *Modelos de Colaboración*. Los mismos establecen las reglas generales y los parámetros a ser tenidos en cuenta para la gestión integrada de la cadena de suministro, como así también definen los lineamientos de los principales procesos colaborativos a ser realizados. Los diferentes tipos de modelos de colaboración son discutidos en el capítulo 2.

Los modelos de colaboración han surgido debido a que las nuevas Tecnologías de Información y Comunicación (TICs), principalmente aquellas basadas en Internet, han posibilitado a las empresas conducir sus relaciones de negocio en forma electrónica, originando lo que se conoce como *relaciones Business-to-Business (Negocio-a-Negocio)*. Las empresas entablan relaciones Business-to-Business (B2B) para llevar a cabo transacciones comerciales, intercambio de información y ejecución de procesos colaborativos a través de TICs.

El concepto de *relación B2B* abarca tanto el modelo de negocio, como la tecnología de información utilizada. Con respecto a esto último, las relaciones B2B comprenden la integración de los sistemas de información de una empresa con los de sus socios de negocio (clientes, proveedores, bancos, etc.) (Markus y otros, 2003). Esta integración es realizada a través de los sistemas de información B2B, los cuales abarcan a más de una empresa y están constituidos por componentes pertenecientes a cada socio. Estos sistemas son los encargados de dar soporte a la ejecución de los procesos colaborativos. Por lo tanto, el objetivo de las relaciones B2B es integrar a las empresas desde el punto de vista de los sistemas de información. Cuando se enfocan en dar soporte a los modelos de colaboración para la gestión de la cadena de suministro se las conoce como *relaciones de colaboración B2B* o simplemente *colaboraciones B2B* (Carol, 2001) (Anderson, 2000) (Turban y otros, 2003).

En primer lugar, existen dos tipos de sistemas de información B2B para llevar a cabo colaboraciones B2B: *sistemas e-Marketplaces* y *sistemas de información Peer-to-Peer (P2P)*. Los sistemas e-marketplaces ofrecen una solución centralizada para dar

soporte a los modelos de colaboración. La principal característica de un e-marketplace es que actúa de intermediario, operado por una tercer parte independiente, el cual vincula múltiples empresas (Grieger, 2003b). El control de los procesos colaborativos es realizado completamente por el e-marketplace. Por otra parte, un sistema de información P2P tiene como principal característica que la ejecución de los procesos colaborativos se realiza a través de interacciones directas entre los sistemas de los socios, sin el uso de un servidor centralizado o una parte intermedia (Småros y Främling, 2001) (Chen y otros, 2001). Este tipo de sistema está constituido por componentes autónomos, heterogéneos y distribuidos, a través de los cuales las empresas dan soporte a la ejecución en forma conjunta de los procesos colaborativos. En esta tesis, estos componentes se denominan *Sistemas de Administración de Procesos Colaborativos (SAPC)*. Cada empresa, para llevar a cabo una colaboración B2B, debe implementar un SAPC.

En resumen, los sistemas e-markeplaces son más apropiados para dar soporte a modelos de colaboración que proponen una gestión centralizada. En cambio, los sistemas de información P2P son más apropiados para dar soporte a modelos de colaboración que proponen una gestión descentralizada.

1.2. Problemas y Objetivos

Como se discute en el capítulo 2, sólo a través de modelos de colaboración descentralizados soportados por colaboraciones B2B basadas en sistemas de información peer-to-peer, las empresas podrán alcanzar altos niveles de integración y autonomía, gestión descentralizada de los procesos colaborativos y personalización de estos procesos con cada socio. Como contraparte de estos beneficios, la desventaja radica en los altos costos para el desarrollo e implementación de estas soluciones, debido principalmente a que cada empresa debe generar soluciones específicas para cada uno de los socios con los cuales desea colaborar. Esto trae aparejado nuevos desafíos, tanto desde el punto de vista de la gestión de los procesos colaborativos como desde el punto de vista de la interoperabilidad de los SAPC de cada socio, lo cual hace que el desarrollo sea más complejo y el tiempo para implementar dichas relaciones sea mayor. Aunque las colaboraciones B2B son de larga duración y estables, los tiempos para implementar las mismas son críticos. Los socios necesitan reducir estos tiempos, con el objetivo de aprovechar las oportunidades de negocio que detectan.

Por otra parte, la creciente cantidad de nuevas tecnologías y estándares para el desarrollo de sistemas de información B2B está causando confusión a las empresas (Baghdadi, 2004). Actualmente, no existe un método que determine el proceso ha seguir en el desarrollo de estas relaciones.

Para abordar estas dificultades, es necesario separar la vista que los analistas de negocio y diseñadores de sistemas tienen de la solución, de la vista que tienen los arquitectos y desarrolladores del sistema de información P2P. La primera se refiere al nivel de negocio de una colaboración B2B, esto es el dominio del problema, mientras que la segunda se refiere al nivel tecnológico, esto es el dominio de la solución (Figura 1-1).

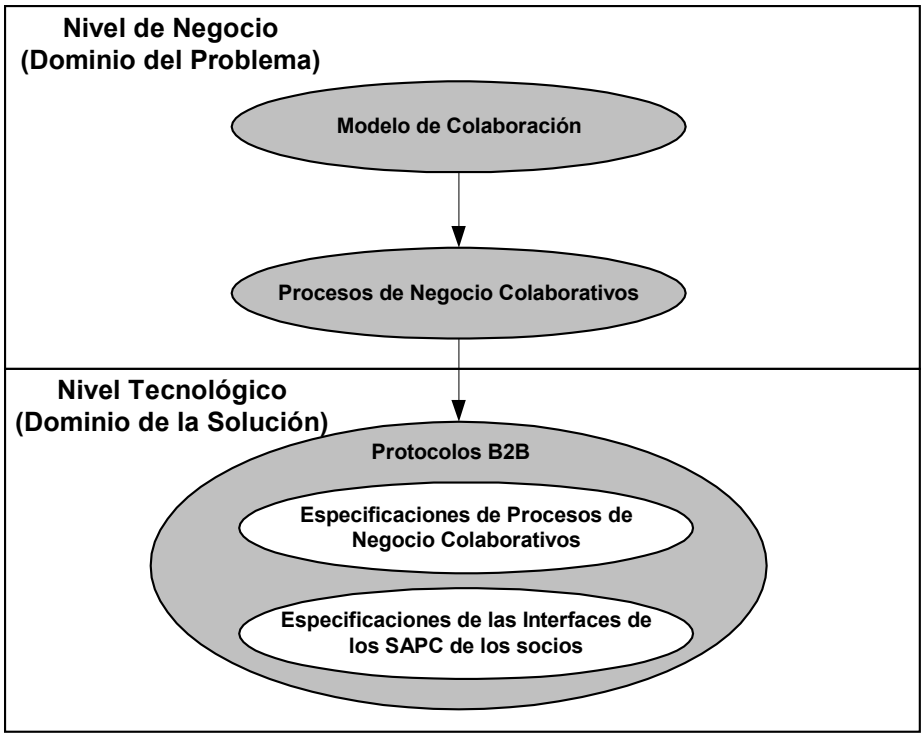


Figura 1-1. Niveles de abstracción de una colaboración B2B.

En el nivel de negocio, las empresas acuerdan el modelo de colaboración a utilizar para llevar a cabo una gestión integrada de la cadena de suministro. Aunque el modelo explica la manera en que las empresas llevarán a cabo la colaboración, su definición explícita y detallada es formalizada en los procesos de negocio colaborativos. Uno o más procesos serán derivados y definidos respetando los lineamientos generales del modelo de colaboración. Dichos procesos deben definirse y acordarse en forma conjunta por los socios. Un proceso colaborativo no sólo determina la información a ser

intercambiada, sino también el comportamiento que tendrán las interacciones entre los socios y las responsabilidades de cada uno en dicho proceso. Mediante la ejecución de estos procesos las empresas podrán tomar decisiones en forma conjunta para alcanzar metas comunes, coordinar sus acciones e intercambiar información. Por lo tanto, el diseño de los procesos colaborativos es un objetivo clave para las empresas que participan en una colaboración B2B.

Debido a que en este nivel, los analistas de negocio y diseñadores de sistemas no desean tratar con los detalles técnicos de la implementación, los procesos colaborativos deben ser definidos en forma completamente independiente de la tecnología de implementación. Por lo tanto, un importante requerimiento es el uso de modelos conceptuales que representen procesos colaborativos independientes de la tecnología.

En el nivel tecnológico, las empresas deben enfocarse en la tecnología de información a utilizar, para dar soporte a la ejecución de los procesos colaborativos. En este nivel, el mayor desafío es posibilitar las interacciones entre los SAPC que forman parte del sistema de información P2P. La interoperabilidad entre los SAPC puede ser alcanzada a través del uso de estándares B2B, los cuales proveen un conjunto de lenguajes para la especificación de los denominados *Protocolos B2B* (Bussler, 2002) (Bernauer y otros, 2003a).

Un protocolo B2B define, por un lado, varios aspectos de las interfaces de los SAPC, tales como los servicios que soportan a los procesos, el protocolo de transporte a utilizar, los tipos de documentos, los aspectos de seguridad, etc. (Bussler, 2002); mientras que, por otro lado, define la lógica de los procesos colaborativos a través de la especificación de los mismos en un *Lenguaje Ejecutable de Procesos de Negocio*, el cual es provisto por un estándar B2B. Estos lenguajes están basados en XML (W3C, 1997) y permiten especificar un proceso en un formato que puede ser procesado e interpretado por software. Los SAPC deben poder interpretar estas especificaciones con el objetivo de ejecutar, controlar, y monitorear los procesos colaborativos, intercambiando mensajes basados en XML. Además, los SAPC deben implementar las interfaces de los socios definidas en el protocolo B2B. De esta forma, se cumple con un importante requisito de las colaboraciones B2B: llevar a cabo una gestión descentralizada de los procesos colaborativos a través de interacciones peer-to-peer entre SAPC autónomos, distribuidos y heterogéneos (Villarreal y otros, 2003c). La

implementación de cada uno de los SAPC puede ser realizada en forma independiente (Villarreal y otros, 2003b), usando diferentes tecnologías y plataformas. La única restricción es que deben interoperar en base a lo definido en los protocolos B2B. Esta tesis se enfoca en la generación de los protocolos B2B, pero no en el desarrollo de los SAPC.

De esta manera, en el nivel tecnológico, un proceso colaborativo es implementado a través de un protocolo B2B, que contiene la especificación del proceso en un lenguaje ejecutable y la especificación de las interfaces correspondientes de los SAPC de los socios. En este nivel, los socios deben seleccionar y acordar el estándar B2B que consideren más adecuado para la implementación de los mismos.

En resumen, para el desarrollo de colaboraciones B2B, es necesario primero dar soporte al diseño de procesos colaborativos en el nivel de negocio, esto es el dominio del problema, y luego dar soporte a la definición de los mismos en el nivel tecnológico, esto es el dominio de la solución, a través de la especificación de estos procesos y la especificación de las interfaces de los socios, utilizando un estándar B2B.

De acuerdo a lo expresado, se puede inferir que el principal concepto en una colaboración B2B es el de proceso de negocio colaborativo independiente de la tecnología. En primer lugar, el comportamiento de la colaboración y los roles que cada socio debe desempeñar quedan explicitados en dichos procesos. Además, las especificaciones de los procesos colaborativos y las interfaces de los SAPC en un estándar B2B, deben tener una correspondencia mutua con los procesos y las interfaces definidas en el nivel de negocio. Para ello, es necesario que estas especificaciones sean derivadas en forma automática, a partir de modelos conceptuales definidos con independencia de la tecnología. Varios trabajos (Baghdadi, 2004) (Bernauer y otros, 2003a) han reconocido la necesidad de proveer métodos o guías tanto para el diseño conceptual de los procesos colaborativos, como para la derivación automática de dichas especificaciones.

Por lo tanto, con el propósito de que las empresas puedan implementar, de manera rápida y con el mínimo esfuerzo posible, colaboraciones B2B para dar soporte a modelos de colaboración descentralizados, en este punto, surge el primer objetivo de esta tesis:

Proveer un método sistemático para el desarrollo de procesos de negocio colaborativos.

Dicho método debe tratar con estos dos niveles de abstracción y proveer los lineamientos a seguir para dar soporte a las tres etapas del desarrollo de procesos colaborativos: análisis, diseño y especificación. La etapa de *análisis* trata con la captura de requerimientos de negocio (detalles del acuerdo de colaboración, metas de negocio comunes, etc.) y la identificación de los procesos a partir del modelo de colaboración seleccionado. La etapa de *diseño* trata con la definición detallada del comportamiento de los procesos colaborativos. Estas dos etapas comprenden la modelación de dichos procesos en forma independiente de la tecnología. Además, la etapa de diseño, también trata con las interfaces independientes de la tecnología de los SAPC de los socios. La última etapa, la de *especificación*, comprende la especificación de dichos procesos y de las interfaces de los socios en un estándar B2B. Luego, el propósito de tal método debería ser disminuir el tiempo y la complejidad en el desarrollo de procesos colaborativos, a través de la provisión de lenguajes, técnicas y herramientas que permitan llevar a cabo dos tareas fundamentales:

- *Modelado Conceptual de Procesos Colaborativos Independientes de la Tecnología*, para dar soporte a las etapas de análisis y diseño.
- *Generación Automática de Especificaciones de los Procesos Colaborativos y de las Interfaces de los SAPC en un estándar B2B*, para dar soporte a la etapa de especificación.

El objetivo de esta última tarea es integrar el nivel de negocio y el nivel tecnológico, de tal manera que lo definido en ambos niveles sea consistente.

Como se discute en el capítulo 3, sección 3.4, actualmente no existe un método robusto y bien definido que provea los lenguajes, técnicas y herramientas adecuadas para realizar cada una de estas tareas.

Para dar soporte al modelado conceptual, es necesario la utilización de un lenguaje de modelado de procesos colaborativos adecuado. Las características de las colaboraciones B2B, a las que se está apuntando en esta tesis, imponen ciertos requerimientos que deben ser satisfechos en un modelo de procesos colaborativos. Dichos requerimientos son discutidos en forma detallada en el capítulo 3, sección 3.2.1.

Actualmente, como se discute en el capítulo 3, sección 3.4, existe una carencia de lenguajes de modelado independientes de la tecnología que soporten el requerimiento de modelar procesos colaborativos que representen: la vista global de las interacciones peer-to-peer, y las responsabilidades de los roles que desempeñan los socios en los procesos. Por otra parte, como se discute en el capítulo 4, sección 4.1, los actuales lenguajes de modelado de procesos de negocio basados en workflows no son adecuados para procesos colaborativos, debido a que no soportan los requerimientos a ser satisfechos en el modelado de los mismos. Además, en colaboraciones B2B, las interacciones entre empresas no pueden estar restringidas a la mera transferencia de información. Estas interacciones apuntan principalmente a la realización de acciones por parte de cada empresa y la comunicación de esas acciones hacia sus otros socios, con el objetivo de llevar a cabo una gestión coordinada de los procesos. Estos aspectos comunicativos tampoco están contemplados en muchos de los lenguajes de modelado existentes. Finalmente, como se discute en el capítulo 4, sección 4.5, las propuestas existentes para el diseño de procesos colaborativos no dan un soporte adecuado para definir negociaciones complejas en los procesos colaborativos.

Por lo tanto, esto da lugar a la definición del segundo objetivo de esta tesis:

Proveer un lenguaje de modelado para procesos colaborativos, que proporcione abstracciones conceptuales adecuadas en el dominio de las colaboraciones B2B.

A través de este lenguaje las empresas podrán definir los procesos colaborativos usando una semántica común, que les permita entender de la misma manera estos procesos. Además, el propósito es proveer un lenguaje que sea fácil de usar y comprender por los analistas de negocio y los diseñadores de sistemas que no conocen, o no están interesados, en los aspectos tecnológicos de la solución.

Con respecto a la generación automática de la solución tecnológica, actualmente existen dos tipos de estándares que pueden ser utilizados para la definición de los protocolos B2B (Bernauer y otros, 2003a): estándares basados en Composición de Servicios Web, tales como *Business Process Execution Language for Web Services (BPEL)* (BEA y otros, 2003) y *Web Service Choreography Interface (WSCI)* (W3C, 2002); y estándares basados en Transacciones de Negocio, tales como *Electronic*

Business using eXchange Markup Language (ebXML) (OASIS, 1999) y *RosettaNET* (RosettaNet, 1999).

Como consecuencia de estas distintas soluciones tecnológicas posibles, un socio podría establecer colaboraciones B2B con varios socios usando estándares diferentes. Más aún, una empresa podría utilizar una misma solución en el nivel de negocio para con dos socios, y utilizar con cada socio estándares diferentes. Esto significa que una solución definida en el nivel de negocio podría ser materializada en el nivel tecnológico a través de diferentes tecnologías.

Este problema debería ser resuelto en tiempo de diseño, definiendo procesos colaborativos independientes de la tecnología y reutilizándolos para generar soluciones tecnológicas basadas en diferentes estándares.

Por lo tanto, esto da lugar a la definición del tercer objetivo de esta tesis:

Proveer un método y una herramienta para generar en forma automática especificaciones, basadas en un estándar B2B específico, de los procesos colaborativos y de las interfaces de los SAPC de los socios, a partir de modelos conceptuales de procesos colaborativos.

Debido a que los actuales estándares B2B están aún en su etapa de evolución, existen varias versiones de los mismos, y nuevos estándares están próximos a aparecer, dicho método no debería ser específico para cada solución tecnológica a generar. Por el contrario, debería permitir a las empresas definir y generar en forma automática diferentes soluciones tecnológicas.

Finalmente, para demostrar la robustez de un lenguaje de modelado de procesos colaborativos, es necesario validar que a partir de modelos definidos con dicho lenguaje es posible derivar los principales conceptos requeridos en el nivel tecnológico, según el estándar B2B a utilizar. Esto da lugar al cuarto objetivo de esta tesis:

Generar, a partir de modelos conceptuales de procesos colaborativos, soluciones tecnológicas, tanto en estándares basados en transacción de negocio como en estándares basados en composición de servicios web.

Esto permitirá a las empresas contar con una guía para la generación de soluciones en estos dos tipos de tecnologías, como así también conocer los aspectos que deben ser considerados en cada una.

1.3. Principales Contribuciones

Para alcanzar el primer objetivo, siguiendo los principios establecidos por la Arquitectura Conducida por Modelos (Model-Driven Architecture, MDA) (OMG, 2003a), en esta tesis se propone **un método, basado en la filosofía del desarrollo conducido por modelos, para procesos colaborativos**. Dicho método provee los lineamientos y el marco conceptual para separar los modelos de procesos colaborativos de las soluciones tecnológicas.

Para alcanzar el segundo objetivo, en esta tesis se propone **un lenguaje de modelado de procesos colaborativos denominado UP-ColBPIP (UML Profile for Collaborative Business Proceses based on Interaction Protocols)**. Con el propósito de satisfacer los requerimientos identificados para el modelado de procesos colaborativos, la base teórica del lenguaje está sustentada en el uso del formalismo de protocolos de interacción, junto con la aplicación de los principios de la teoría de actos de comunicación (Austin, 1962) y la teoría LAP (Languaje/Action Perspective) (Flores y Ludlow, 1980). Siguiendo con los lineamientos de un método basado en MDA, el lenguaje ha sido definido como un Perfil UML, basado en la nueva versión del Lenguaje de Modelado Unificado (Unified Modeling Language, UML) (OMG, 2003b).

Para alcanzar el tercer objetivo, se propone **un método y una herramienta para transformaciones de modelos**. Estos son utilizados para transformar modelos de procesos colaborativos definidos con el lenguaje UP-ColBPIP, a modelos basados en un estándar B2B, como así también son utilizados para generar el código XML correspondiente a las especificaciones de dichos procesos y de las interfaces de los SAPC.

Para alcanzar el cuarto objetivo, se definen **las transformaciones de modelos definidos con el lenguaje UP-ColBPIP a especificaciones basadas en el estándar ebXML, y a especificaciones basadas en los estándares BPEL y WSDL**.

1.4. Organización de la Tesis

El capítulo 2 tiene como objetivo establecer en forma precisa el contexto en el que se enmarcan las soluciones propuestas en esta tesis. Este capítulo describe el concepto de colaboración entre empresas y sus principales beneficios para la gestión integrada de la cadena de suministro. Además se discuten y analizan los principales modelos de colaboración que establecen la forma en que dicha gestión puede ser realizada. Posteriormente, se examinan y clasifican los tipos de relaciones B2B que dan soporte a estos modelos y se discuten los beneficios y costos de los mismos. Finalmente, se destacan los modelos de colaboración y los tipos de relaciones B2B sobre los cuales se enfocará esta tesis.

En el capítulo 3, en primer lugar se describen los conceptos involucrados en el nivel de negocio y el nivel tecnológico de una colaboración B2B. Además, se identifican los requerimientos para dar soporte al diseño conceptual de procesos de negocio colaborativos, como así también los requerimientos para la generación de especificaciones ejecutables de estos procesos y de las interfaces de los socios en un estándar B2B. Luego se presenta el método basado en el desarrollo conducido por modelos, propuesto para el desarrollo de procesos colaborativos. Dicho método sigue los principios establecidos por la Arquitectura Conducida por Modelos (MDA) para separar los modelos de procesos colaborativos, de las soluciones tecnológicas. Los principales componentes y técnicas de este método son introducidos. El capítulo finaliza con una comparación y discusión de los trabajos relacionados.

El capítulo 4 presenta el lenguaje de modelado conceptual UP-ColBPIP, propuesto para diseñar procesos colaborativos independientes de la tecnología. Se describen los objetivos, los principios de diseño y las bases teóricas del lenguaje. Además, se presenta el metamodelo de UP-ColBPIP, en donde se describe la semántica de los conceptos provistos por el lenguaje y las relaciones entre los mismos. Luego, se presenta la definición de UP-ColBPIP como un perfil UML que implementa dicho metamodelo. La aplicación del lenguaje se describe a través de un caso de estudio. Finalmente, se discuten trabajos relacionados y se comparan los mismos con el lenguaje propuesto.

El capítulo 5 presenta el método y la herramienta de transformación de modelos propuestos para llevar a cabo transformaciones de modelos de procesos colaborativos. En este capítulo, se abordan los conceptos y las principales características que debe

tener un método de transformaciones de modelos, y se examinan los diferentes enfoques existentes. Luego se discuten los requerimientos particulares y se establecen los objetivos para llevar a cabo transformaciones de modelos de procesos colaborativos. Posteriormente, se describen los componentes del método: el lenguaje de transformación de modelos, el lenguaje de modelado de reglas y las reglas de producción de código XML. Finalmente se presenta la arquitectura y los principales componentes de la herramienta de transformación de modelos que soporta al método definido.

El capítulo 6 presenta la generación de soluciones tecnológicas basadas en el estándar ebXML a partir de modelos definidos con UP-ColBPIP. Dos lenguajes relevantes provistos por este estándar son descritos: el lenguaje BPSS, que permite especificar procesos colaborativos, y el lenguaje CPPA, que permite especificar las interfaces de los socios. Como parte del método de desarrollo conducido por modelos para procesos colaborativos se describen: la transformación de modelos UP-ColBPIP a BPSS y la transformación de modelos UP-ColBPIP a CPPA. Las mismas son definidas usando el método y la herramienta de transformación de modelos propuestos en el capítulo 5. Para ejemplificar la salida final de dichas transformaciones, correspondiente al código XML que generan, se utiliza el modelo UP-ColBPIP definido en el caso de estudio del capítulo 4.

El capítulo 7 presenta la generación de soluciones tecnológicas basadas en composición de servicios web a partir de modelos UP-ColBPIP. En particular se generan soluciones basadas en los lenguajes BPEL y WSDL. Cada uno de estos lenguajes es descrito brevemente. Como parte del método de desarrollo conducido por modelos de procesos colaborativos se describen: la transformación de modelos UP-ColBPIP a BPEL y la transformación de modelos UP-ColBPIP a WSDL. Para ejemplificar la salida final de dichas transformaciones, correspondiente al código XML que generan, se utiliza el modelo UP-ColBPIP definido en el caso de estudio del capítulo 4.

El capítulo 8 tiene como objetivo destacar las principales contribuciones de esta tesis, y describir los aspectos de las mismas que constituyen el punto de partida para el desarrollo de trabajos futuros.

Capítulo

2

MODELOS DE COLABORACIÓN Y RELACIONES BUSINESS-TO-BUSINESS PARA LA GESTIÓN DE LA CADENA DE SUMINISTRO

El objetivo de este capítulo es establecer en forma precisa el contexto en el que se enmarcan las soluciones propuestas en esta tesis. En primer lugar se describen los conceptos de colaboración entre empresas y sus principales beneficios para la gestión integrada de la cadena de suministros. Luego se discuten y analizan algunos modelos de colaboración propuestos para llevar a cabo dicha gestión. Posteriormente, se examina la tecnología de información que posibilita la implementación de estos modelos. Se discuten las características de los diferentes tipos de relaciones Business-to-Business (B2B) y se provee una clasificación de las mismas. Luego, se detallan los beneficios y costos de las colaboraciones B2B entre empresas. Finalmente se destacan los modelos de colaboración y los tipos de relaciones B2B sobre los cuales se enfocará esta tesis. El objetivo de este capítulo es establecer en forma precisa el contexto en el que se enmarcan las soluciones propuestas en esta tesis.

2.1. Colaboración entre Empresas para la Gestión Integrada de la Cadena de Suministro

Históricamente, el modelo de negocio convencional utilizado por cada empresa de la cadena de suministro consiste en administrar en forma independiente los activos de su propia cadena de suministro interna (Sahay y Mohan, 2004). Tanto la producción en masa como el precio de los productos son considerados los criterios más importantes en las interacciones entre empresas. Por lo tanto, cada empresa mantiene una relación informal con las restantes empresas de la cadena, en donde los resultados de las operaciones y decisiones de las demás empresas sólo se consideran parámetros sujetos a incertidumbres.

Sin embargo, en las últimas décadas las empresas se han enfrentado a un mercado cuya estructura ha ido cambiando. De ser un mercado estable, con demanda quasi-

infinita, se ha llegado a un mercado globalizado, sumamente competitivo. Las empresas, para ser competitivas en este mercado, se ven en la necesidad de: mejorar la calidad de sus productos, proveer un alto nivel de servicio a los clientes, manejar expectativas de precios bajos, disminuir el tiempo de elaboración de productos, manejar sistemas productivos flexibles y confiables, y reducir los costos totales (Mulani y Lee, 2002). Esto ha llevado a las empresas a adoptar nuevas filosofías de gestión basadas en el establecimiento de relaciones más estrechas con sus clientes y proveedores, desde filosofías como JIT (Just-In-Time) en la década de los ochenta, hasta el concepto de integración de la cadena de suministro a partir de los noventa (Sahay y Maini, 2004). Con el objetivo de ser competitivas, esto último ha conducido a las empresas, aún a las pequeñas y medianas, a formar parte de cadenas de suministro que compiten entre sí en un mercado globalizado.

Como respuesta a esta nueva situación, nuevos modelos de negocio sustentados en la gestión integrada de la cadena de suministro han sido propuestos. Los mismos se basan en el análisis del conjunto de empresas involucradas en la tarea de dar valor a los productos finales, desde el abastecimiento de materias primas y la producción, hasta la distribución y venta de los mismos. De esta manera, se considera a la *gestión integrada de la cadena de suministro* como el 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 y otros, 1999). También puede definirse a la gestión integrada de la cadena de suministro como la integración de los procesos de negocio claves que van desde el usuario final hasta los proveedores originales que suministran los productos, servicios e información, que agregan valor a los clientes (Lambert y Cooper, 1998). La gestión de los flujos de materiales, de información y financiero es el principal interés de la gestión integrada de la cadena de suministro.

En la gestión integrada de la cadena de suministro, las empresas comparten información con el objetivo de disminuir las incertidumbres asociadas al flujo de materiales. Diferentes tipos de información pueden ser compartidos: información de inventario de los socios, información de ventas, estado de las órdenes, pronósticos de ventas, planes de producción y de entrega, etc. (Lee y Wang, 2000). Los beneficios de compartir información entre los socios de la cadena han sido demostrados con diversos

estudios analíticos (Lee y otros, 1997), (Chen y otros, 2000) y empíricos (Xu y otros, 2001). Los principales beneficios son niveles de inventario más bajos a través de la cadena de suministro, disminución de costos, mejoras en el flujo de caja, tiempos de ciclos más pequeños y predecibles, y más flexibilidad a cambios en la demanda.

Varios autores coinciden en la existencia de diferentes niveles de integración de la cadena de suministro (Grieger, 2003a). Spekman y otros (1998) definen tres niveles de integración o niveles de intensidad entre los socios de negocio: cooperación, coordinación y colaboración. Cooperación es cuando las empresas interactúan con el nivel más bajo de intensidad a través del intercambio de información. El próximo nivel de intensidad es coordinación, en donde las empresas además de intercambiar información, coordinan sus actividades a través de procesos de negocio. No obstante, cooperación y coordinación no significa que las empresas persigan metas de negocio comunes. Colaboración tiene el nivel más alto de intensidad y requiere los niveles más altos de confianza, de compromiso y de intercambio de información en la cadena de suministro y una visión común del futuro. A través de relaciones de negocio colaborativas las empresas entablan procesos y planificaciones en forma conjunta. El objetivo de la colaboración entre empresas es disminuir los costos e incrementar los beneficios para cada empresa, alcanzando un beneficio mutuo para todos los socios.

Por otro lado, Lee y Wang (2001) distinguen cuatro niveles de integración: el primer nivel se refiere a la integración de la información y el último establece que la integración implica la generación de un nuevo modelo de negocio.

Sahay y Maini (2004) también distinguen cuatro niveles, donde el primer nivel se refiere a cómo las empresas se relacionan con el modelo tradicional de negocio, hasta un último nivel en donde las empresas colaboran entre sí para ejecutar conjuntamente procesos de negocio compartidos.

Sin tener en cuenta el nivel de integración básico del modelo de negocio tradicional, todas las anteriores definiciones de integración concuerdan en que el nivel más bajo de integración se refiere al intercambio de información entre empresas, mientras que el nivel más alto de integración conduce al concepto de colaboración entre empresas. Por lo tanto, se pueden definir dos tipos extremos de integración de la cadena de suministro. En el primer tipo las empresas sólo intercambian información hacia un solo sentido (generalmente desde los clientes a los proveedores) con el objetivo de

disminuir las incertidumbres y mejorar sus procesos de negocio internos. Cada empresa opera y toma sus propias decisiones ha partir de la información que es intercambiada. Un simple ejemplo es una relación entre dos empresas, en la cual el cliente envía al proveedor sus datos de ventas y/o pronósticos de ventas, para que el proveedor los utilice y mejore el rendimiento de sus procesos de negocio internos responsables de la generación de pronósticos de demanda y de la planificación de la producción. Este tipo de integración es el más primitivo e implica una simple cooperación entre las empresas. Aunque se reduce la incertidumbre asociada a ciertos parámetros, existen otros problemas derivados del desconocimiento que tiene el proveedor de las decisiones que toma el cliente y de las limitaciones de capacidad que éste posee (Lee y otros, 1997).

El segundo tipo de integración, conduce al concepto de *colaboración* entre empresas. Dicho concepto implica una integración mucho más fuerte que va más allá del intercambio de información. *Colaboración* significa que las empresas establecen relaciones de negocio, en donde los socios pueden mejorar sus beneficios y reducir sus costos a través de la ejecución en forma conjunta de procesos de negocio entre empresas (Mentzer y otros, 2000). Por lo tanto, la principal característica de la colaboración entre empresas es una integración a nivel estratégico y operacional de los procesos de negocio de los socios (Sahay y Gupta, 2004). Los procesos de negocio internos o privados de una empresa son integrados con los de sus socios en la cadena a través de la definición y ejecución en forma conjunta de los *procesos de negocio colaborativos*. Estos últimos también se conocen como *procesos de negocio públicos, compartidos, entre empresas o entre organizaciones*. Los mismos se extienden a través de dos o más empresas y permiten coordinar las actividades y procesos de cada empresa. Un simple ejemplo de colaboración entre dos empresas es cuando las mismas, previo a la definición de un acuerdo de colaboración, definen y ejecutan un proceso de negocio colaborativo para acordar en forma conjunta un pronóstico de demanda o un plan de órdenes.

De esta manera, la colaboración conduce a una comunicación bidireccional, en donde los socios llevan a cabo una toma de decisiones en forma conjunta mientras preservan sus propios objetivos (Liu y Kumar, 2003). Generalmente, una relación de colaboración se establece para un período de mediano o largo plazo a partir de la definición de un acuerdo de colaboración entre las partes, el cual puede tener validez legal o no.

La integración de proveedores y compradores en una cadena de suministro a través de relaciones de colaboración no sólo permite disminuir las incertidumbres sino también permite mejorar su eficiencia y rendimiento. La colaboración entre empresas incrementa: las respuestas de las mismas hacia el mercado, la satisfacción al cliente final y la competitividad de todos los socios. Además, la planificación conjunta permite reducir distorsiones y manipulaciones unilaterales en los pronósticos de demanda y de órdenes, estableciendo un mecanismo de confianza entre los socios. Algunos de estos beneficios de la colaboración también han sido demostrados por diversos estudios analíticos (Aviv, 2001) (Fu y Piplani, 2004) y estudios empíricos (Kaipia y Saarinen, 2001).

Por lo tanto, actualmente las empresas están enfocadas en establecer relaciones de colaboración con sus proveedores y clientes, no sólo para mejorar el rendimiento de la cadena de suministro global sino también con el objetivo de disminuir sus costos y mejorar sus rendimientos internos. Diferentes modelos de negocio han sido propuestos para llevar a cabo la colaboración entre empresas. Dichos modelos se conocen como *modelos de colaboración*. A continuación se discuten los principales modelos.

2.2. Modelos de Colaboración

En esta sección se describen y analizan los siguientes modelos de colaboración para la gestión integrada de la cadena de suministro: VMI (Vendor Managed Inventory), CPFR (Collaborative Planning, Forecasting and Replenishment), DAMA (Demand Activated Manufacturing Architecture) y el Modelo de Colaboración Socio a Socio (Partner-to-Partner Collaborative Model).

2.2.1. Modelo de Colaboración VMI

El modelo VMI (Inventario Gestionado por el Proveedor) es quizás el tipo de modelo de colaboración más difundido, el cual tiene como propósito gestionar los procesos de aprovisionamiento en forma colaborativa. Este tipo de modelo también es conocido como SMI (Supplier Managed Inventory). La principal característica de un modelo VMI es que el proveedor es el responsable de la tarea de planificar el aprovisionamiento y controlar los niveles de inventario del cliente (Xu y otros, 2001). El principal objetivo es reducir los niveles de inventarios tanto del lado del vendedor como del lado del cliente.

En un modelo VMI, los clientes comparten con su proveedor los datos actuales de sus inventarios, como así también los datos de la demanda. Basado en esta información, el proveedor genera un pronóstico de demanda y luego un plan de órdenes de aprovisionamiento para cada cliente. A partir de este plan, el proveedor realiza los envíos y genera una notificación de envíos. Cada notificación será seguida por el transporte físico de los productos. Luego el cliente genera un acuse de recibo por cada envío. En un modelo VMI, los socios establecen algunas métricas para evaluar el rendimiento de la colaboración, como por ejemplo la tasa de cumplimientos, el nivel máximo esperado en el inventario del cliente, etc. Por lo tanto, es necesario definir algunos procesos de negocio colaborativos que manejen las excepciones en el caso que el rendimiento esperado por los socios no sea alcanzado (Liu y Kumar, 2003).

Además de los beneficios principales de VMI hacia ambas partes, este tipo de modelo provee un incentivo más fuerte para los clientes, dado que el cliente no incurre en los costos relacionados con la generación de los pronósticos de demanda, la colocación de órdenes en el proveedor y el control del inventario (Xu y otros, 2001). No obstante, los proveedores también disminuyen sus costos debido a que cuentan con información más certera de la demanda de sus clientes, lo cual permite gestionar de mejor manera sus ciclos de producción.

El modelo VMI ha sido utilizado generalmente en la industria de bienes empaquetados al consumidor, en donde la relación se establece entre empresas de producción que cumplen el rol de proveedores y centros de distribución o centros de ventas al consumidor (hipermercados, supermercados, etc.) que cumplen el rol de clientes.

Existen diferentes variaciones de este modelo. Una variación es la consignación del inventario, cuando el proveedor, además de controlar el inventario en el almacén del cliente, retiene la posesión del mismo. El cliente no realiza el pago hasta que el producto sea vendido. No obstante, desde el punto de vista de la colaboración, esto no implica demasiados cambios. Otro ejemplo de un modelo VMI en el cual las partes apuntan a establecer una colaboración más estrecha, es el modelo VMI basado en pronóstico (CompTIA EIDX, 2004) propuesto para la industria electrónica. En éste, antes de comenzar el proceso de aprovisionamiento, los socios definen en forma conjunta una única orden para todo el período que dura el acuerdo de colaboración.

Luego, el proveedor realizará el aprovisionamiento siguiendo el mecanismo descripto, tomando como marco de referencia esta orden.

2.2.2. Modelo de colaboración CPFR)

El modelo de colaboración CPFR (Collaborative Planning, Forecasting and Replenishment) ha sido desarrollado por el Voluntary Inter-Industry Standards Association (VICS) y adoptado principalmente por las compañías de bienes empaquetados al consumidor (VICS, 1998). CPFR define un conjunto de procesos de negocio colaborativos en los cuales los socios, en forma conjunta, acuerdan metas y métricas de negocio comunes, desarrollan planes operacionales y de ventas, y generan y actualizan los pronósticos de ventas y los planes de aprovisionamiento.

El objetivo principal de CPFR es alinear de la mejor manera el suministro y la demanda a través del intercambio de datos, la gestión basada en excepciones, y la colaboración estructurada entre los socios de negocio para eliminar los problemas y restricciones en el cumplimiento de las expectativas de los consumidores finales (Liu y Kumar, 2003).

En CPFR los socios deben definir en forma conjunta no sólo pronósticos de ventas, sino también pronósticos de órdenes (Holmström y otros, 2002). Además, la utilización de los procesos de manejo de excepciones permite que ante la ocurrencia de cambios bruscos en la demanda, eventos inesperados o la incorporación de nuevas políticas de promociones, los planes y pronósticos gestionados en forma conjunta puedan ser ajustados inmediatamente, minimizando o eliminando los costos de las correcciones para ambas partes.

CPFR ha sido utilizado principalmente entre empresas de producción que cumplen el rol de proveedores y centros de distribución o centros de ventas al consumidor (hipermercados, supermercados) que cumplen el rol de clientes. Experiencias pilotos entre empresas tales como Nabisco y Wall-Mart, han demostrado resultados alentadores en cuanto al mejoramiento de la disponibilidad de sus productos, incrementos en las ventas y reducción de inventarios (Småros, 2002).

El modelo CPFR original define un modelo de proceso genérico, el cual está dividido en 9 etapas que representan las principales actividades de colaboración, de acuerdo al vocabulario de CPFR (VICS, 2002). Estas etapas pueden ser vistas como

cada uno de los procesos de negocio colaborativos a ser llevados a cabo. Para cada etapa se definen los pasos del proceso, la información de entrada y la de salida. El modelo CPFR actualmente ha sido redefinido en ocho etapas de colaboración que son más fáciles de entender que el modelo original (VICS, 2004). Este modelo representa la colaboración entre dos empresas, un comprador y un vendedor, que trabajan en forma conjunta para satisfacer la demanda del cliente final, el cual es el centro del modelo. La Figura 2-1 ilustra el modelo de referencia de CPFR.

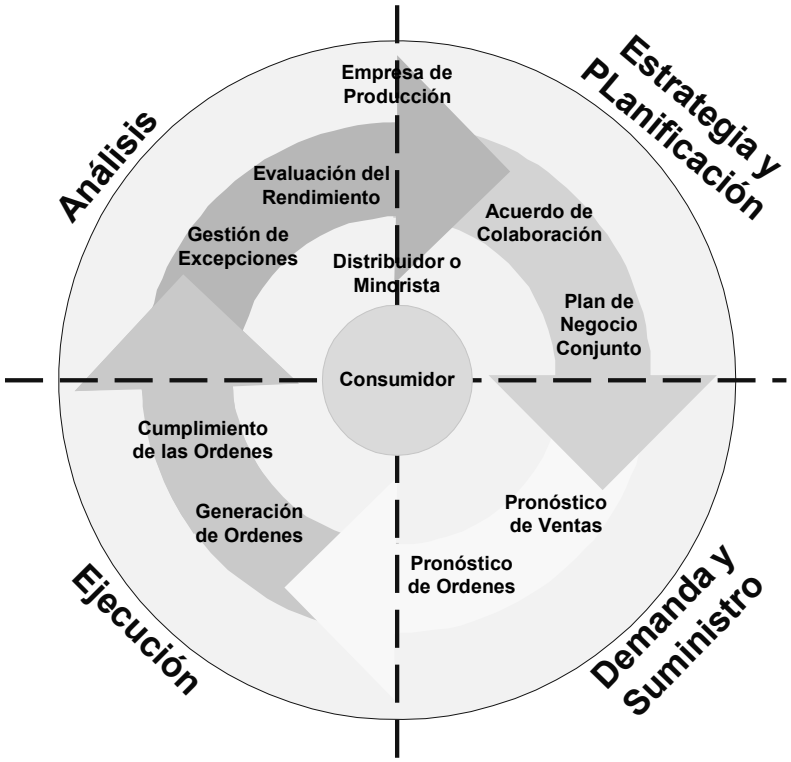


Figura 2-1. Modelo de colaboración CPFR – Etapas y procesos

Dentro de la etapa *estrategia y planificación*, el *acuerdo de colaboración* es el proceso de establecer las metas de negocio de la relación, definir el alcance de la colaboración, asignar roles y responsabilidades, definir criterios de excepción y determinar la información a ser compartida para dar soporte a la colaboración (frecuencia de actualizaciones, metodología de pronóstico, etc.). El *plan de negocio conjunto* identifica los eventos más significativos que afectan el suministro y la demanda en el período de planificación, tales como promociones, cambios en las políticas de inventario, días en que los puestos de ventas están abiertos, e introducciones de productos. El resultado es un plan de negocio acordado mutuamente que define estrategias y tácticas para cada uno de los productos sobre los que se realiza la

colaboración. Este plan es la información principal a tener en cuenta en los procesos de pronósticos, de manera tal de reducir las excepciones y el número de iteraciones.

La etapa *demanda y suministro* es dividida en dos procesos: el proceso de *pronóstico de ventas*, en el cual los socios definen una proyección de la demanda de los puntos de ventas basados en información causal y en los eventos planificados; y el proceso de *pronóstico/planificación de órdenes*, en el cual los socios determinan los requerimientos futuros de ordenación y entrega de los productos de acuerdo a los pronósticos de ventas, las disponibilidades de inventario, los tiempos de provisión, y otros factores.

La etapa de *ejecución* consiste en el proceso de *generación de órdenes*, el cual convierte a los pronósticos de órdenes firmes; y en el proceso de *cumplimiento de las órdenes*, el cual se refiere al proceso de producir, embarcar, entregar y almacenar los productos para su venta.

La etapa de *análisis* incluye el proceso de *gestión de excepciones*, el cual realiza el monitoreo de la planificación y de la ejecución de acuerdo a los criterios de excepción definidos en el acuerdo; y el proceso de *evaluación del rendimiento*, a través del cual se calculan las métricas claves para evaluar el cumplimiento de las metas de negocios comunes, descubrir tendencias o desarrollar alternativas.

Aunque estas etapas de colaboración son presentadas en un orden lógico, las empresas generalmente están involucradas en todas ellas en todo momento. No existe una secuencia de pasos predefinidos. Esto significa que habrá siempre instancias de estos procesos, las cuales dispararán nuevas instancias de otros procesos o modificarán valores que serán usados en instancias de otros procesos. Por ejemplo, problemas en la etapa de ejecución pueden impactar en la estrategia establecida por los socios, y los resultados de la etapa de análisis pueden conducir a ajustes en los pronósticos.

Además de los procesos, CPFR define cuatro escenarios, que representan variantes del modelo genérico, los cuales pueden ser aplicados a relaciones de negocio particulares de acuerdo a los recursos que cada socio posee. Los escenarios definen quién es el socio que en última instancia será el responsable de generar los pronósticos. Por ejemplo, el primer escenario delega al comprador la responsabilidad final de generar los pronósticos de ventas y de órdenes. Mientras los demás escenarios delegan el control final de la generación de las órdenes al vendedor.

El modelo CPFR ha sido desarrollado para llevar a cabo relaciones de colaboración entre dos empresas, principalmente entre una empresa de producción y distribuidores mayoristas o minoristas. Este modelo también es conocido como *Two-tier CPFR (o CPFR de dos bandas)*. Por otra parte, también se ha definido un modelo CPFR, basado en el anterior, para definir relaciones de colaboración entre más de dos empresas que forman parte de diferentes eslabones de la cadena de suministro. Este modelo se conoce como *N-Tier CPFR*. El mismo permite modelar relaciones de colaboración basadas en los procesos de CPFR, entre empresas de una misma cadena de suministro (entre centros de ventas al consumidor, distribuidores, empresas de producción y proveedores de los primeros eslabones de la cadena de suministro).

2.2.3. Modelo de Colaboración DAMA

El modelo DAMA (Demand Activated Manufacturing Architecture) (Chapman y Petersen, 2000) ha sido diseñado para gestionar la cadena de suministro de la industria textil en los Estados Unidos. Este modelo asume una cadena de suministro colaborativa para una línea de productos en particular, con múltiples socios de negocio trabajando para satisfacer la demanda de los consumidores finales. Los socios comparten información acerca de sus productos, capacidades de producción, asignaciones de capacidad y el estado operacional de cada uno día a día.

El modelo DAMA define un proceso de negocio colaborativo compuesto de diez actividades o subprocesos:

- Desarrollar los acuerdos de planificación de negocio.
- Publicar información en la utilidad de la cadena de suministro.
- Definir productos.
- Colaborar en las excepciones para la definición de productos.
- Comprometer pronósticos y capacidades.
- Colaborar sobre pronósticos para resolver excepciones.
- Planificar la producción y la entrega de productos.
- Colaborar en excepciones en la fecha de entrega de los productos.
- Finalizar la producción y despachar los productos.

- Ejecutar la entrega.

Luego de definir el acuerdo de colaboración, el segundo proceso implica que todos los socios deben publicar información inicial sobre las siguiente áreas: producción (tiempos de provisión, tiempos de procesos, tiempos de transporte), asignación de capacidad para los socios, capacidades de producción (líneas de productos, lista de materiales, especificaciones de productos) y criterios de excepción. La información aquí publicada será compartida por los socios y será utilizada por la utilidad de la cadena de suministro. Esta es un conjunto de aplicaciones de sistemas que da soporte a todos los socios en cada uno de los subprocesos o actividades de colaboración. El objetivo de la utilidad de la cadena de suministro es gestionar en forma automática los procesos y generar información que luego podrá ser compartida por los socios. Por ejemplo, la utilidad de la cadena de suministro es la encargada de generar las órdenes de producción para cada una de las empresas de producción, de acuerdo con la información inicial y los compromisos de capacidades definidos en los procesos previos. La utilidad de la cadena de suministro permite a cada uno de los socios acceder a la información generada, proveyéndoles a los mismos una visibilidad completa de toda la cadena de suministro. No obstante, no todos los socios tienen acceso a toda la información. En el acuerdo de colaboración se debe definir qué socio tiene acceso a qué información.

De esta manera, este modelo de colaboración propone una gestión centralizada de la cadena de suministro, basado en el concepto de visibilidad. Las relaciones entre las empresas en este tipo de cadena de suministro son lineales como se muestra en la Figura 2-2, en donde los datos que una empresa produce son consumidos por las empresas contiguas en la cadena a través de la utilidad de la cadena de suministro.

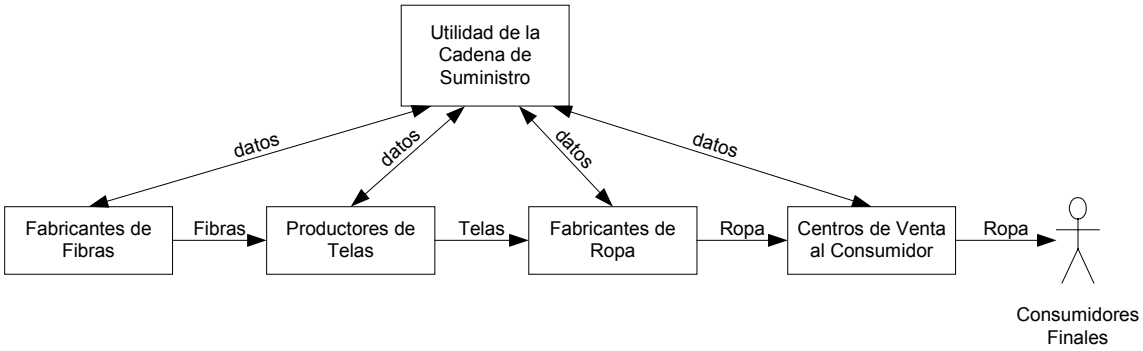


Figura 2-2. Modelo DAMA

2.2.4. Modelo de Colaboración Socio-a-Socio

El Modelo de Colaboración Socio-a-Socio (Villarreal y otros, 2003a) (Villarreal y otros, 2001) surge con el objetivo de dar soporte a dos requerimientos principales: gestionar relaciones de colaboración entre empresas de producción que forman parte de varias cadenas de suministro; y gestionar de manera descentralizada los procesos de negocio colaborativos que los socios deben definir, ejecutar y monitorear en forma conjunta para llevar a cabo una planificación colaborativa de la producción.

A diferencia de CPFR, en donde generalmente las relaciones son entre un distribuidor mayorista o minorista y un productor, este modelo está orientado a modelar relaciones de colaboración entre empresas de producción. En una relación entre un distribuidor y un productor, la integración implica sincronizar el proceso de producción del proveedor con una política de inventario del cliente. En cambio, en relaciones de colaboración entre empresas de producción, la integración es más compleja debido a que implica sincronizar el proceso de producción del proveedor con el proceso de producción del cliente. Para alcanzar esta sincronización, las empresas deben llevar a cabo negociaciones, no sólo sobre los pronósticos sino también sobre los planes de producción. Dichas negociaciones pueden involucrar varias iteraciones.

Este modelo propone que las relaciones de colaboración entre empresas sean definidas como relaciones de a pares e independientes, las cuales sólo involucran dos socios, en donde uno cumple el rol de proveedor y el otro cumple el rol de cliente. A través de relaciones de colaboración socio-a-socio, este modelo propone una gestión descentralizada de dichas relaciones, como así también una gestión descentralizada de toda la cadena de suministro, debido a que no considera la existencia de una entidad central que sea la coordinadora de toda la cadena. El beneficio obtenido a lo largo de una cadena de suministro será la suma de los beneficios obtenidos en cada una de las relaciones proveedor-cliente existentes en la cadena.

La definición de relaciones de colaboración socio-a-socio es un enfoque de integración que permite dar soporte al nuevo escenario actual, donde existen redes de cadenas de suministro que compiten unas con otras (Liu y Kumar, 2003). En estas redes, clientes y proveedores, no forman parte de una única cadena de suministro sino que colaboran con empresas que juntas conforman diferentes cadenas de suministro (Cavusoglu y otros, 2001). A su vez, diferentes cadenas de suministro comparten en sus

eslabones proveedores y clientes, los cuales generalmente son empresas de producción. Por ejemplo, una empresa que produce paquetes tetra brick suministra sus productos a diferentes industrias, como la láctea, la de jugo, la de vino, etc. Este es el caso de una red como se muestra en la Figura 2-3 (a), en donde una empresa de producción suministra productos a varias empresas de producción pertenecientes a diferentes cadenas de suministro.

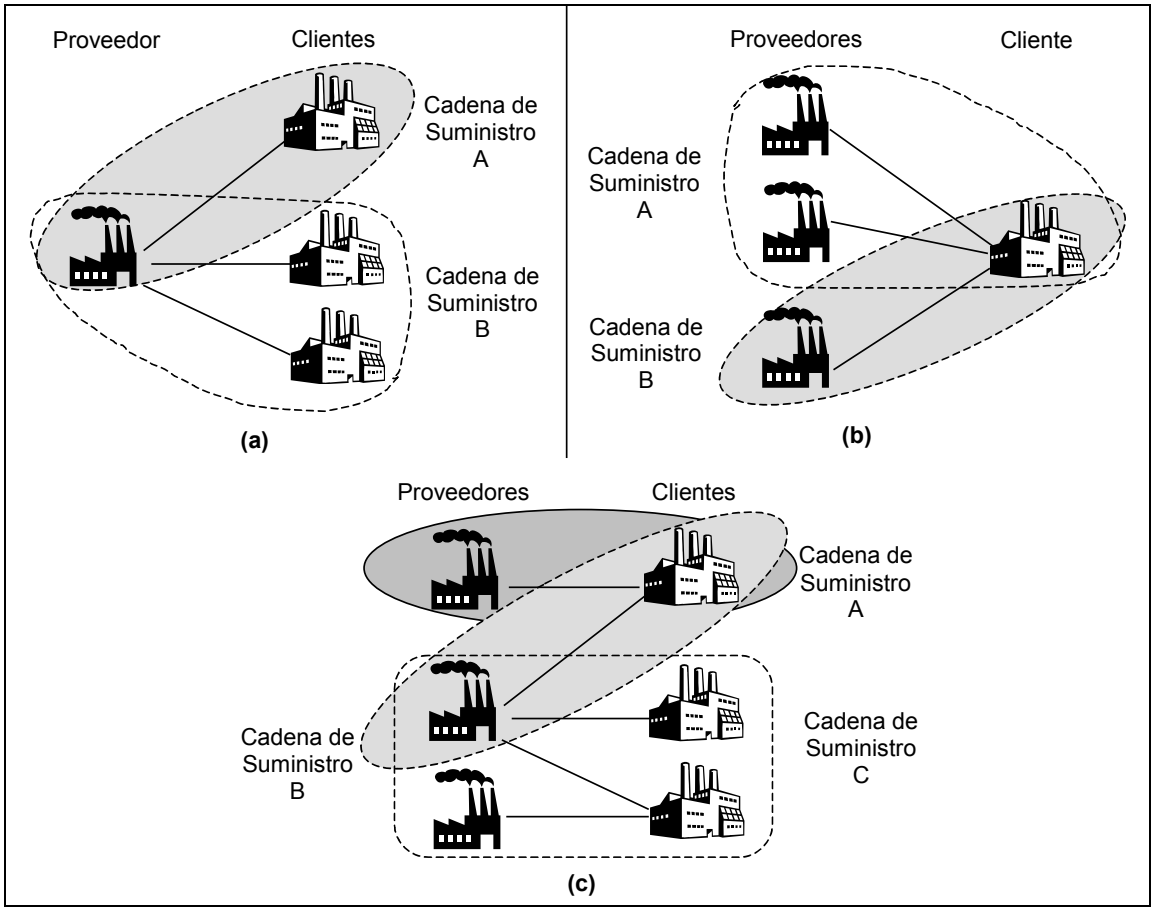


Figura 2-3. Relaciones de colaboración entre empresas través de diferentes cadenas de suministro

A su vez, otro tipo frecuente de red es el que muestra la Figura 2-3 (b), donde una empresa es provista por dos o más empresas que pertenecen a diferentes cadenas. Por ejemplo, una industria láctea es provista de materiales por una empresa que provee cartón corrugado, otra que provee tetra brick, otra que provee poliestireno, etc.

Analizando estos ejemplos, es posible ver que la relación entre estas empresas no conduce a una única cadena de suministro colaborativa lineal, como la soportada por el modelo DAMA, sino que es un conjunto de relaciones complejas que dan lugar a redes

o web de suministros donde existen varias cadenas de suministro, como se muestra en la Figura 2-3 (c). Este tipo de redes también se conoce como redes de colaboración (Turban y otros, 2003). En redes de colaboración, una empresa, en cualquier punto de la red, también puede relacionarse con otra evitando establecer relaciones con las empresas que tradicionalmente están contiguas en la cadena.

Para dar soporte a este tipo de relaciones entre empresas, un modelo de colaboración centralizado, como el que propone el modelo DAMA, no es adecuado. Un modelo centralizado apunta generalmente a gestionar una cadena de suministro lineal, para un tipo específico de industria y para una línea de productos, en donde cada empresa posee un alto nivel de participación en la cadena de suministro y además es altamente dependiente de la cadena en la cual está inserta. Un modelo centralizado considera a la cadena de suministro como un sistema cerrado que incluye a todas las empresas que forman parte de la misma.

En cambio, el modelo de colaboración socio-a-socio considera a cada empresa como un sistema abierto, el cual interactúa con otros sistemas abiertos (empresas) mientras cada una conserva su autonomía, con el objetivo de colaborar en la gestión de los flujos de materiales y de información. Esto permite una mayor flexibilidad a las empresas, en cuanto que las mismas pueden formar parte de varias cadenas, sin perder su autonomía y sin ser totalmente dependientes de una sola cadena.

El modelo de colaboración Socio-a-Socio considera que dos empresas deben compartir información y tomar decisiones conjuntas acerca de los planes de producción, los planes de órdenes y los planes de entrega de productos. Para ello el modelo define los procesos de negocio colaborativos genéricos que los socios deben gestionar en forma conjunta y descentralizada. Dichos procesos se basan en los conceptos del modelo de planificación y control de la producción jerárquico usado actualmente para la gestión interna de la empresa (Dominguez Machuca y otros, 1995), el cual es extendido para gestionar la colaboración entre empresas. El propósito es que las empresas lleven a cabo procesos de negocio colaborativos continuos, considerando todos los niveles de planificación y sin perder sus autonomías con respecto a sus decisiones y actividades internas.

El Modelo de Colaboración Socio-a-Socio define cuatro procesos de negocio colaborativos (Villarreal y otros, 2003a):

- **Acuerdo de Colaboración.** En este proceso los socios establecen las reglas generales que van a gobernar la relación (período de la colaboración, período de congelamiento de órdenes, niveles de inventario, etc.), definen las metas comunes a alcanzar junto con las métricas de rendimiento que permitirán evaluar dichas metas, e identifican los productos sobre los que se va a colaborar. Cada empresa evalúa sus propias estrategias y metas para asegurar que las mismas sean tenidas en cuenta a la hora de definir las metas comunes entre ambos.

- **Planificación Colaborativa de los Recursos y de la Producción**. El objetivo de este proceso es que los socios definan en forma conjunta un plan de provisión (o suministro) de productos al nivel de planificación agregada de la producción. A partir de este resultado del proceso, cada socio define su planificación agregada.

- **Programa Maestro de la Producción Colaborativo**. El objetivo de este proceso es que los socios definan en forma conjunta un plan de órdenes al nivel de la programación maestra de la producción. A partir de este resultado del proceso, cada socio define su programa maestro de producción definitivo.

- **Planificación de los Requerimientos de Materiales Colaborativa**. El objetivo de este proceso es que los socios definan en forma conjunta un plan de aprovisionamiento y de entrega, teniendo en cuenta sus requerimientos de materiales. A partir de este resultado del proceso, cada socio define su plan de órdenes de provisión y su plan de órdenes de producción interno.

En cada uno de los procesos, se considera que la capacidad de los recursos de los socios es finita y por lo tanto, cada uno, al momento de acordar los planes, deberá tener en cuenta sus capacidades.

Considerando lo discutido anteriormente en (Villarreal y otros, 2003a), las relaciones de colaboración socio-a-socio que debe gestionar cada empresa son clasificadas de acuerdo a dos tipos de relaciones y a dos niveles de colaboración. Esto da lugar a la definición de cuatro casos diferentes que una empresa debe considerar cuando interactúa con otras para obtener o suministrar un determinado producto a través de un modelo de colaboración socio-a-socio:

- *Relación 1 a N con colaboración total con respecto a un producto*: un proveedor establece relaciones de colaboración socio-a-socio con todos sus clientes con respecto a un determinado producto. Es decir que todo lo que se produzca de ese producto será utilizado para satisfacer a todos sus socios clientes.
- *Relación N a 1 con colaboración total con respecto a un producto:* un comprador establece relaciones de colaboración socio-a-socio con todos sus proveedores con respecto al suministro de un producto. Todo lo consumido por el comprador con respecto al producto considerado será provisto por sus socios proveedores.
- *Relación N a 1 con colaboración parcial con respecto a un producto:* un comprador que provee un producto a varios clientes, establece relaciones de colaboración sólo con algunos de sus clientes con respecto a ese producto, pero no con todos.
- *Relación 1 a N con colaboración parcial con respecto a un producto:* un proveedor establece relaciones de colaboración sólo con algunos de sus clientes con respecto a un producto, pero no con todos los clientes.

Estos casos deben ser tenidos en cuenta por las empresas al momento de definir sus procesos de gestión internos involucrados en las operaciones de planificación y control de la producción. Para cada caso, los procesos de negocio internos que coordinan las diferentes relaciones de colaboración tendrán características diferentes. En (Villarreal y otros, 2003a), para cada proceso colaborativo, se describe lo que debe realizar cada empresa cuando las mismas estén comprometidas en una única relación de colaboración, es decir una *Relación 1 a 1*. En cambio, en (Caliusco y otros, 2004a), se describen los procesos de negocio y decisión internos de los proveedores y el comprador para relaciones de colaboración N a 1, en donde el comprador tiene establecido una relación de colaboración con N proveedores.

2.3. Relaciones Business-to-Business

Las nuevas Tecnologías de Información y Comunicación (TICs), principalmente aquellas basadas en Internet, tienen el potencial de facilitar y posibilitar la integración de la cadena de suministro, permitiendo la implementación de modelos de colaboración

entre empresas (Hardaker y Graham, 2000) (Giménez y Lourenço, 2004) (Kauremaa, y otros, 2004). Estas tecnologías incrementan la riqueza de las comunicaciones a través de una interactividad mayor entre la empresa y sus clientes y proveedores. A través del uso de tecnologías de información las empresas conducen actividades transaccionales y comerciales por medios electrónicos. Esto da lugar a lo que se conoce como negocio electrónico o e-Business, aunque muchas veces también se lo confunda con e-Commerce, el cual se refiere sólo a la compra y venta de bienes en forma electrónica por parte de las personas. El concepto de e-Business es un concepto más general que abarca a otros dos conceptos: *Comercio Electrónico Negocio a Negocio*, más conocido como *Business-to-Business* (B2B), y *Comercio Electrónico Negocio a Consumidor*, más conocido como *Business-to-Consumer* (B2C). Este último no es de interés en esta tesis.

B2B se refiere a las transacciones comerciales y las comunicaciones entre empresas realizadas en forma electrónica a través de TICs. Aunque B2B se halla difundido a partir de la expansión de Internet, el mismo se ha llevado a cabo desde hace varias décadas a través del uso de vínculos basados en EDI (Electronic Data Interchange) (Kalakota y Robinson, 1999). No obstante, EDI sólo ha sido usado por las grandes corporaciones y fue resistido por las pequeñas y medianas empresas, debido a que requiere de altos costos de instalación y operación. En comparación a vínculos basados en EDI, las relaciones B2B basadas en Internet ofrecen la posibilidad de reducir sustancialmente dichos costos. Los principales beneficios de las relaciones B2B basadas en Internet son: disminución de los costos de las transacciones debido a la reducción de los costos tradicionales (costos de papel, fax, teléfono, etc.), intercambio de información en forma más confiable y más rápida, intercambio de información en tiempo real, reducción de errores y mejoramiento de la productividad.

Las relaciones B2B comprenden la integración, en alguna medida, de los sistemas de información de una empresa con los de sus socios de negocio (clientes, proveedores, bancos, etc.) (Markus y otros, 2003). La integración de las empresas al nivel de procesos de negocio requiere de la creación de una infraestructura de tecnología de información apropiada. Por lo tanto, los sistemas de información de las empresas también necesitan ser integrados. Las empresas requieren de sistemas de información B2B que lleven a cabo esta integración y posibiliten el intercambio de información y la gestión de los modelos de colaboración que implementan con sus socios de negocio.

Además del concepto de integración, los conceptos de gestión integrada de la cadena de suministro y relaciones B2B tienen en común que los mismos se refieren a cómo conducir interacciones entre empresas. Ambos son conceptos estratégicos fundamentales para que las empresas puedan forjar relaciones de colaboración con sus proveedores y clientes. No obstante, estos conceptos son diferentes y a la vez complementarios (Alt y otros, 2000). El primero apunta a la integración de las empresas desde el punto de vista de negocios. El segundo apunta a la integración de las empresas desde el punto de vista de los sistemas de información, con el objetivo de dar soporte a los modelos de colaboración.

Las empresas no sólo deben implementar relaciones B2B que les posibiliten intercambiar información y realizar transacciones comerciales, sino que también deben implementar relaciones B2B que posibiliten la gestión de los procesos de negocio colaborativos, a través de la aplicación de un modelo de colaboración que permita gestionar en forma integrada la cadena de suministro (Sahay y Gupta, 2004). De esta manera, complementando la aplicación de los modelos de colaboración entre empresas con un soporte tecnológico basado en relaciones B2B, es posible que las empresas puedan implementar con éxito relaciones de colaboración.

De la misma manera en que existen diferentes tipos de integración de la cadena de suministro, existen diferentes tipos de relaciones B2B, las cuales definen cómo los sistemas de las empresas serán integrados para soportar un modelo de negocio específico. Algunos autores denominan a las mismas como modelos B2B. No obstante, en esta tesis hablaremos de tipos de relaciones B2B. En definitiva, las relaciones B2B permiten conducir e implementar modelos de colaboración. Las mismas son clasificadas de acuerdo a dos aspectos que consideramos primordiales:

- Según los procesos de negocio a los que están dirigidas.
- Según la estructura o topología de las relaciones.

A continuación se describen estas dos clasificaciones.

2.3.1. Tipo de Relaciones B2B según los Procesos de Negocio Soportados

Una forma de clasificar a las relaciones B2B es según los procesos de negocio principales a los que dan soporte. Algunas relaciones B2B se enfocan en los procesos de venta de bienes y servicios, es decir, ven al B2B desde el punto de vista del vendedor.

Otras se enfocan en los procesos de compra de bienes y servicios, tomando al B2B sólo desde el punto de vista del comprador. Finalmente, otras se enfocan en los procesos relacionados a la gestión de la cadena de suministro. A continuación se describen brevemente cada una de ellas.

2.3.1.1. Relaciones B2B desde el Punto de Vista del Vendedor

Los modelos B2B desde el punto de vista del vendedor más conocidos son: catálogos electrónicos (e-Catalog), intercambios electrónicos (e-Exchange) y subastas electrónicas (e-Auctions). Estos modelos se diferencian de acuerdo al mecanismo de precio que utilizan los vendedores para conducir transacciones comerciales. El objetivo de los vendedores es agrupar un gran número de compradores. En el caso de catálogos electrónicos los precios de los productos son fijados de antemano y no están sujetos a negociación. No obstante, a los compradores se les provee de capacidades de búsquedas para que puedan encontrar el producto que reúne sus necesidades. En los modelos de intercambios electrónicos y subastas electrónicas, el precio de los productos es negociado en forma dinámica.

En estos modelos los compradores generalmente tienen la opción de colocar sus órdenes vía Web en los sistemas del vendedor. El objetivo de estos modelos es posibilitar la compra y venta de bienes y servicios entre empresas en forma electrónica. Es decir, que estos modelos sólo tienen como propósito realizar la ejecución de las transacciones comerciales en forma electrónica.

2.3.1.2. Relaciones B2B desde el Punto de Vista del Comprador

El modelo B2B más conocido desde el punto de vista del comprador es el modelo B2B e-procurement, el cual tiene que ver con los procesos de negocio involucrados en la obtención de materiales, ya sea, materiales directos (necesarios para la producción de los productos finales) o indirectos (necesarios para el mantenimiento de los recursos, como ser máquinas, repuestos de los equipos, insumos de oficina, etc.). Este modelo implica que dos o más empresas (compradores y vendedores) integran sus sistemas con el objetivo de llevar a cabo los diferentes procesos involucrados en la obtención de materiales (Dai y Kauffman, 2001). Estos procesos son: el de solicitud de propuestas (para obtener el producto deseado), el de colocación de las órdenes al proveedor

seleccionado, el de recepción de productos y el de facturación y pago (Raisinghani, 2005).

El objetivo de los modelos B2B de obtención de materiales es reducir los costos de búsqueda y de operación de los procesos de compras de las empresas (Dai y Kauffman, 2001). Este modelo plantea mayores desafíos que los anteriores, debido a que es necesaria una integración de los sistemas de las partes, para conducir la obtención de materiales. No obstante, debido a que sólo se tiene en cuenta la vista del comprador, generalmente los procesos involucrados en estos modelos son controlados y gestionados por el comprador. Por lo tanto, estos modelos no están dirigidos a establecer relaciones de colaboración estrechas entre los socios, sino a disminuir los costos de las transacciones comerciales tradicionales y hacer más eficientes y rápidas las actividades de compra de bienes y servicios.

2.3.1.3. Colaboraciones B2B para la Gestión de la Cadena de Suministro

Los anteriores tipos de B2B están orientados a dar soporte al intercambio de información y la ejecución de transacciones comerciales entre empresas que forman parte de una cadena de suministro, con el objetivo de comprar y vender bienes y servicios a través de Internet. Estos tipos de B2B dan soporte a los procesos de negocio que forman parte del nivel operacional de la cadena de suministro (por ej: compra, venta, cumplimiento de las órdenes, distribución, etc.). No obstante, para implementar los modelos de colaboración, las relaciones B2B deben dar soporte no sólo a los procesos de nivel operacional, sino también a los procesos que se refieren a la planificación de la demanda y del suministro y a los procesos de nivel estratégico (por ej.: definición del acuerdo de colaboración y diseño de la cadena de suministro). Aunque estos últimos generalmente son difíciles de automatizar, los procesos de planificación son los procesos claves y diferenciadores de las relaciones de colaboración entre empresas.

Las relaciones B2B que se enfocan en dar soporte a los modelos de colaboración para la gestión de la cadena de suministro se las conoce como *relaciones de colaboración B2B* o simplemente *colaboraciones B2B* (Carol, 2001) (Anderson, 2000) (Turban y otros, 2003). Las colaboraciones B2B son conducidas por los socios de negocio involucrados en una cadena de suministro o en redes de colaboración y refiere a la ejecución de procesos de negocio colaborativos definidos por los modelos de

colaboración y a la integración de los mismos con los procesos privados (Mulani y Lee, 2002).

Las colaboraciones B2B se enfocan en posibilitar que los socios aumenten sus beneficios y disminuyan sus costos a través de la toma de decisión en forma conjunta. Por lo tanto, este tipo de relación implica mayores desafíos desde el punto de vista de la tecnología de información, debido a que los socios deben integrar sus sistemas de información para conducir en forma conjunta procesos de negocio colaborativos, desde los procesos estratégicos y tácticos hasta los operativos de la cadena de suministro.

2.3.2. Tipo de Relaciones B2B según su Estructura o Topología

Otra forma de clasificar a las relaciones B2B es según la estructura o topología de las relaciones en red que entablan las empresas. Más específicamente, esta clasificación se refiere a cómo los sistemas de información externos de cada empresa serán integrados. En este caso las relaciones B2B caen en dos categorías: sistemas e-marketplaces (o mercados electrónicos) y sistemas de información peer-to-peer (o sistemas de información par a par). A continuación se describen las principales características de estas soluciones, como así también ventajas y desventajas de su implementación para dar soporte a los modelos de colaboración.

2.3.2.1. E-Marketplaces

Una de las soluciones B2B más populares son los e-marketplaces (o mercados electrónicos). No existe una única definición comúnmente aceptada de su significado sino que diferentes definiciones destacan algunos atributos especiales como virtual, digital, público, abierto, o neutral. La principal característica de un e-marketplace es que actúa de intermediario vinculando múltiples compradores y vendedores en un espacio de mercado central (Grieger, 2003b). Se trata de un sistema de información B2B, operado por una tercer parte independiente, y diseñado para conducir relaciones B2B entre empresas. Los e-markeplaces funcionan como intermediarios, generalmente verticales y enfocados en una industria específica, los cuales permiten crear y mediar conexiones muchos-a-muchos entre las empresas (Dai y Kauffman, 2001).

Existen varios aspectos que permiten diferenciar a los e-marketplaces. Estos pueden ser clasificados de acuerdo al tipo de bienes que comercian (ej: materiales directos), por el tipo de mecanismo de comercio utilizado (ej: catálogos, subastas, intercambios) o

según su propiedad (ej: públicos o privados). Un importante atributo son los servicios que proveen. En este caso se tienen básicamente dos tipos de e-marketplaces: transaccionales y colaborativos (Christiaanse y Markus, 2003) (Gogolin, 2003).

Los e-marketplaces transaccionales son entendidos como espacios donde compradores buscan por productos, por proveedores y/o por precios a través de catálogos electrónicos, subastas o mecanismos de precios dinámicos. Luego de seleccionar un producto y acordar su precio, el e-marketplace permite ejecutar la compra. Un e-marketplace basado en precios dinámicos o en subastas relaciona las órdenes en tiempo real con las ofertas que ingresan al mismo. En estos casos, actúa como un intermediario entre compradores y vendedores. Los e-marketplaces que dan soporte al proceso de obtención de materiales y de productos (e-procurement) también caen en la categoría de transaccionales. En este tipo de e-marketplaces, las empresas no establecen relaciones estrechas y a largo plazo, sino que interactúan de forma ad-hoc para llevar a cabo una única transacción comercial. En definitiva, el propósito es atraer un gran número de compradores y vendedores independientes proveyendo un simple acceso con pocas barreras de entrada. Estos e-marketplaces solo tienen como objetivo conducir transacciones comerciales e intercambios de información, por lo cual la coordinación entre empresas no es requerida (Goethals y otros, 2005).

Por sus características, los e-marketplaces transaccionales no pueden implementar los procesos de negocio requeridos por los modelos de colaboración. Para ello, muchos e-marketplaces están adicionando servicios que permitan implementar modelos de colaboración para la gestión de la cadena de suministro. Estos e-marketplaces se conocen como e-marketplaces colaborativos (Christiaanse y Markus, 2003)(Gogolin, 2003). Los servicios que ofrecen van desde la integración de la información que es intercambiada, a la integración de procesos a través del soporte para planificación en forma conjunta. El objetivo es fomentar la colaboración entre empresas que ya tienen establecidas relaciones de negocio estrechas. Así, estos e-marketplaces se orientan más a las relaciones específicas de una empresa con sus socios, donde el número de participantes es bajo o medio y además estable. Aunque estos e-marketplaces generalmente son públicos, las empresas deciden con quién van a colaborar.

Los e-marketplaces colaborativos ofrecen una solución centralizada para implementar los modelos de colaboración. Este enfoque provee algunas ventajas

(Småros y Främling 2001), las cuales son resumidas en la Tabla 2-1. En primer lugar, las empresas pueden colaborar con sus socios de negocio a través de un único vínculo de comunicación. De esta manera deben integrar sus sistemas de información con una única parte (el e-marketplace) y no con cada uno de sus socios. Esto trae aparejado la disminución de los costos para las empresas, tanto de los costos de instalación como así también de los costos de mantenimiento, los cuales son absorbidos por el e-marketplace. Además, permite que nuevas relaciones de colaboración de una empresa con nuevos socios sean implementadas rápidamente sin la necesidad de crear un nuevo vínculo de comunicación por cada socio. Otra ventaja es que pueden imponer la utilización de un estándar B2B, ya sea abierto o propietario, para el intercambio de información y ejecución de los procesos colaborativos. Este estándar debe ser adoptado por cada uno de los socios y de esta manera no es necesario tratar con diferentes estándares.

E-MARKETPLACES COLABORATIVOS	
Ventajas	**Desventajas**
• Implementación de un único vínculo de comunicación con el e-marketplace • Disminución de los costos de instalación y mantenimiento de la integración B2B • Los e-marketplaces pueden imponer un estándar B2B para el intercambio de información y ejecución de los procesos colaborativos. No es necesario manejar varios estándares B2B. • Visibilidad es provista a todos los socios con respecto a la información compartida y al estado de los procesos.	• Problemas de confianza de las empresas sobre los mecanismos utilizados para el control y distribución de la información compartida. • Los socios podrían requerir de soluciones más específicas a los procesos colaborativos estandarizados que ofrece el e-marketplace. • Las empresas deben pagar por los servicios ofrecidos por el e-marketplace. • Difícil de mantener actualizada la información de los socios.

Tabla 2-1. Ventajas y desventajas de los e-marketplaces para la gestión de modelos de colaboración

A pesar de estos beneficios, los e-marketplaces poseen importantes desventajas para su aplicación en modelos de colaboración (Småros y Främling, 2001). Una primera desventaja se refiere a la pérdida de control de la información. La empresa debe confiar en que el e-marketplace entregará la información al receptor correcto y solamente a ese receptor. Esto podría implicar una pérdida de poder estratégico en el mercado y generar

un problema de confianza entre los socios, debido al alto riesgo de que el e-marketplace beneficie a las empresas más grandes o sólo a algunos de los socios. Otra desventaja se refiere a los problemas de escalabilidad asociados a la implementación de los procesos de negocio colaborativos. Debido a que un e-marketplace ofrece procesos estandarizados, los mismos podrían sólo ser apropiados para algunos socios, mientras que otros necesitan soluciones más específicas. Una tercera desventaja se refiere a los costos de ejecución, ya que la empresa no sólo necesita invertir para vincularse con el e-marketplace, sino que debe pagar por los servicios ofrecidos. Otra desventaja, es el problema de mantener actualizada la información, lo cual es responsabilidad del socio.

Una aparente ventaja de los e-marketplace colaborativos es la visibilidad que proveen acerca de la información compartida entre los socios, debido a la gestión centralizada de la misma. Esta centralización además permite que los socios puedan ver el estado de ejecución de los procesos colaborativos. No obstante, aunque esta visibilidad podría ser requerida por modelos de colaboración que proponen una gestión centralizada de la cadena de suministro, la misma no es requerida por los modelos de gestión descentralizados. En estos modelos, la visibilidad de la información y del estado de los procesos está limitada a los dos o más socios que participan en una relación de colaboración (Villarreal y otros, 2004a).

Finalmente, la principal desventaja del uso de e-marketplaces colaborativos para la implementación de modelos de colaboración es la pérdida de la autonomía de las empresas (Villarreal y otros, 2003c). Esto se debe a que la gestión y control de la información compartida y de los procesos de negocio colaborativos es realizada por el e-marketplace, es decir por una tercer parte y no por cada uno de los socios involucrados. Por otra parte, muchos de los e-marketplaces colaborativos que actualmente se encuentran ofreciendo servicios, como por ejemplo ELEMICA para la industria química, no están aplicando todas las funciones requeridas para gestionar la cadena de suministro y sólo dan soporte a los procesos más sencillos que requieren poca interacción o negociación entre los socios (Grieger y Kotzab, 2002). Estas desventajas también son resumidas en la Tabla 2-1.

2.3.1.2. Sistemas de Información Peer-to-Peer

El concepto de Sistemas Peer-to-Peer (P2P) está obteniendo cada vez mayor atención. Desde el punto de vista de sistemas de información, no existe una definición

generalmente aceptada de sistemas de información P2P. La principal característica de estos sistemas es que el intercambio de información y la ejecución de los procesos se realiza a través del intercambio directo de información entre los sistemas de los socios, sin el uso de un servidor centralizado (Småros y Främling, 2001) (Chen y otros, 2001). De esta manera, no existe un intermediario como en el caso de los sistemas e-marketplaces. En una arquitectura P2P, los sistemas de los socios pueden actuar como clientes y servidores a la vez, de acuerdo al rol que desempeñan en cada interacción.

El concepto de P2P también propone la idea de una distribución física de los recursos usados en las relaciones B2B. De esta manera, los sistemas P2P reflejan la idea de que cada empresa es una entidad autónoma que interactúa con otras en forma descentralizada (cada una gestionando sus propios recursos, sistemas, procesos y actividades en forma local).

Existen varias ventajas de los sistemas de información P2P a favor de su implementación para el soporte de modelos de colaboración (Småros y Främling, 2001). Las ventajas y desventajas de los mismos son resumidas en la Tabla 2-2. Por un lado, las ventajas están involucradas con el control de la información y el poder. Cada parte puede decidir qué información compartir y con quién compartirla. Además, cada parte tiene un mismo estatus, sin tener en cuenta su peso o condición dentro de la red de colaboración, y puede elegir en forma independiente cómo colaborar con cada uno de sus socios. Otra ventaja se refiere a la escalabilidad, es decir, a la habilidad de dinámicamente definir relaciones de colaboración sin límites con respecto al número de socios y productos.

Otra de las principales ventajas de estos sistemas es que permiten un alto nivel de autonomía de los procesos de decisión y de los sistemas de información de las empresas (Villarreal y otros, 2003c). Con el uso de sistemas de información P2P, cada empresa es responsable de mantener y actualizar la información que ella misma produce y que será compartida con los socios.

Con respecto a la gestión de los procesos de negocio colaborativos, los sistemas de información P2P posibilitan la implementación de sistemas de gestión de procesos descentralizados (Chen y Hsu, 2001). Es decir, un sistema de información P2P está compuesto de sistemas de administración de procesos colaborativos, pertenecientes a cada uno de los socios que participan de la colaboración. Estos sistemas permiten a los

socios gestionar la parte de los procesos colaborativos, de la cual son responsables. Además, a través de estos sistemas los socios coordinan en forma conjunta la ejecución de los procesos colaborativos. De esta manera, las empresas poseen más autonomía para gestionar sus actividades y la parte de la que son responsables dentro de los procesos colaborativos. Además, toda la información requerida para ejecutar los procesos es gestionada por cada empresa y no por una tercer parte. Con lo cual, cada empresa puede definir la información que es confidencial y la información que compartirá con sus socios.

SISTEMAS DE INFORMACIÓN PEER-TO-PEER	
Ventajas	**Desventajas**
• Cada empresa mantiene el control de la información que comparte con sus socios. • Cada empresa tiene el mismo estatus y poder que sus socios. • Escalabilidad de relaciones de colaboración. • Las empresas mantienen un alto nivel de autonomía. • Gestión descentralizada de la información y de los procesos colaborativos. • Integración con los sistemas internos es realizada dentro de cada empresa. • Personalización de los procesos de negocio colaborativos para cada relación de colaboración. • Los socios pueden decidir qué estándar B2B utilizar de acuerdo a sus requerimientos.	• Con varios socios, los costos de desarrollo y de mantenimiento son más altos. • La solución de integración podría ser única para cada par de empresas.

Tabla 2-2. Ventajas y desventajas de los sistemas de información P2P para la gestión de modelos de colaboración

Por otra parte, debido a que cada empresa gestiona la parte que le corresponde dentro de los procesos colaborativos, la integración de su sistema de administración de procesos colaborativos, con los sistemas de administración de procesos internos puede

ser realizada dentro de la empresa. De esta manera, la coordinación de las actividades internas con las actividades de colaboración es realizada por la empresa misma y no por una tercer parte como en el caso de los e-marketplaces.

El uso de sistemas de información P2P permite dar soporte a la implementación de procesos de negocio colaborativos personalizados y específicos para cada socio (Markus y otros, 2003). Al igual que en un e-marketplace, se requiere el uso de un estándar B2B para el intercambio de información y ejecución de los procesos colaborativos. No obstante, en el caso de sistemas P2P, los socios son los que eligen y acuerdan en forma conjunta el estándar B2B a utilizar. En lugar de usar procesos estandarizados y predefinidos, los socios acuerdan las actividades a realizar y la información a intercambiar, de acuerdo a sus necesidades. De esta manera, cada relación B2B que establece una empresa con un socio es específica para ese socio.

Los beneficios de los sistemas de información P2P traen aparejado como contraparte costos de desarrollo y mantenimiento más altos que el de los e-marketplaces. Esto se debe a que cada empresa debe mantener un vínculo de comunicación diferente con cada uno de sus socios, pudiendo usar diferentes estándares B2B. Por lo tanto, con muchos socios los costos de mantenimiento son altos.

2.4. Costos y Beneficios de las Colaboraciones B2B

Los modelos de colaboración discutidos anteriormente están dirigidos a dar soporte a la colaboración entre empresas en algún tipo de industria específica o para algún tipo de relación específica (por ej.: relaciones fabricante-distribuidor, o frabricante-fabricante o entre todos los eslabones de una cadena de suministro). Si bien estos modelos pueden ser implementados con diferentes tipos de relaciones B2B, algunas de estas relaciones B2B son más adecuadas para ciertos modelos de colaboración.

Los e-marketpaces colaborativos se ajustan mejor a las necesidades de los modelos de colaboración centralizados, como el DAMA, debido a que los mismos requieren de una gestión centralizada de toda la cadena. Del mismo modo, los sistemas de información P2P se ajustan mejor a las necesidades del Modelo de Colaboración Socio-a-Socio, debido a sus requerimientos de autonomía y descentralización en la gestión de los procesos de negocio colaborativos. En cambio, otros modelos de colaboración, como

el modelo CPFR o el modelo VMI pueden ser implementados en forma adecuada tanto con el uso de e-marketplaces como con el uso de sistemas de información P2P.

No obstante, a pesar de que cada solución de colaboración entre empresas (modelo de colaboración + sistema B2B) tiene sus propias características, estas soluciones pueden ser comparadas de acuerdo a tres características claves que las diferencian (Figura 2-4): el grado de integración entre empresas (tanto integración de los negocios como integración de los sistemas), el grado de estandarización y personalización de los procesos de negocio colaborativos y el grado de autonomía que las empresas obtienen.

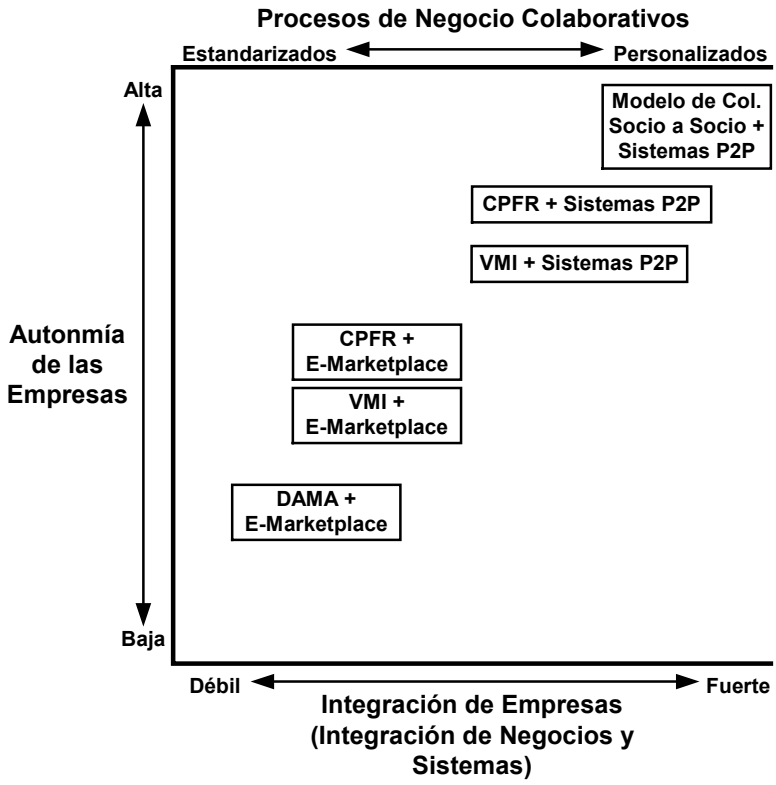


Figura 2-4. Características de las soluciones de colaboración entre empresas

Analizando los procesos colaborativos que son soportados, los modelos de colaboración implementados con e-marketplaces proveen procesos altamente estandarizados, mientras que modelos de colaboración implementados con sistemas de información P2P ofrecen a las empresas la posibilidad de personalizar los procesos con cada uno de sus socios.

Con respecto a la autonomía de las empresas, como se discutió anteriormente, modelos de colaboración centralizados implementados con sistemas e-marketplaces proveen una baja autonomía. No obstante, en la realidad, la mayoría de las empresas

funcionan como entidades autónomas que a la vez interactúan y colaboran unas con otras. Esta característica de las empresas puede ser alcanzada con modelos de colaboración descentralizados implementados con sistemas de información P2P.

La otra característica clave se refiere al grado de integración de las empresas que es alcanzado o requerido en cada solución. Esta integración se refiere tanto a los negocios como a los sistemas. Una alta integración implica una alineación cercana de los procesos de negocio colaborativos, de los sistemas de los socios, y de los documentos de negocio ha ser utilizados en la colaboración (McLaren y otros, 2002). En cambio, una débil integración entre los socios tiene como consecuencia la existencia de diferencias entre ellos con respecto a los procesos de negocio, los sistemas de los mismos, y las definiciones de los datos, lo cual acarrea más esfuerzos de integración internos para cada empresa.

Por lo tanto, las soluciones basadas en modelos de colaboración descentralizados e implementadas por sistemas de información P2P permiten alcanzar los niveles más altos de autonomía de empresas y descentralización, de integración de empresas y de personalización de los procesos de negocio colaborativos.

Por otra parte, además de estas características deseables de las colaboraciones B2B, es necesario comparar los costos y beneficios de cada una de estas soluciones B2B. En (McLaren y otros, 2002) se discuten los beneficios y costos de varios sistemas usados para implementar cadenas de suministro colaborativas. El costo total de estos sistemas es igual a los costos totales de posesión de los sistemas más el costo de oportunidad de las relaciones de colaboración (el costo asociado de estar vinculado con un socio específico y no con un posible socio más beneficioso). Los primeros incluyen los costos de implementación e integración de sistemas, los de coordinación e integración de los procesos de negocio colaborativos y los costos de traducción e integración con los sistemas internos y externos. El beneficio neto de la implementación de estos sistemas se deriva de los beneficios resultantes de la colaboración (reducción de los costos de las empresas, más el incremento del rendimiento de las mismas) menos los costos totales mencionados anteriormente.

Considerando los beneficios y costos establecidos en (McLaren y otros, 2002) con respecto a los sistemas, y considerando lo discutido en este capítulo, en la Figura 2-5 se

detallan los costos y beneficios en términos cualitativos que pueden ser esperados de las soluciones de colaboración entre empresas.

En general, las alternativas de costos más bajos producen los menores beneficios. Esto ocurre en el caso de modelos de colaboración centralizados implementados con e-marketplaces. Similarmente, las alternativas de costos más altos producen también beneficios más altos. Esto ocurre en el caso de modelos de colaboración descentralizados implementados con sistemas de información P2P. Estos beneficios y costos de cada solución pueden ser usados como una guía para anticipar los costos y beneficios de cualquier iniciativa de colaboración entre empresas.

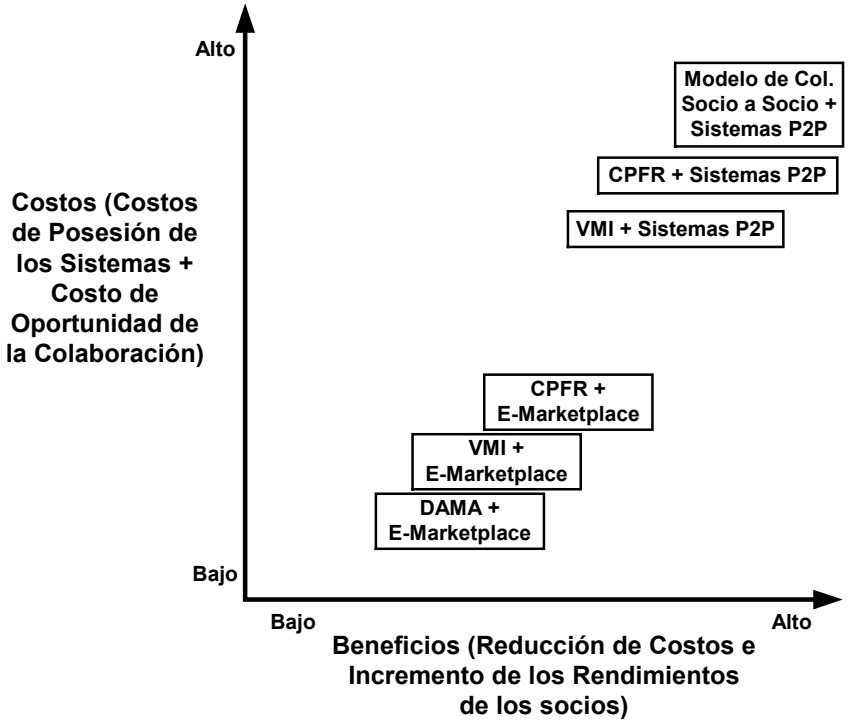


Figura 2-5. Costos-Beneficios esperados de las soluciones de colaboración entre empresas

2.5. Conclusiones

Como se ha descrito en este capítulo, la colaboración entre empresas provee numerosos beneficios para las mismas. La colaboración es el principal factor de éxito para incrementar las eficiencias y los rendimientos de las cadenas de suministro. Sólo a través de relaciones de colaboración las cadenas de suministro podrán ser competitivas en los mercados globales, y por ende las empresas insertas en estas cadenas podrán

también aumentar su competitividad. La colaboración entre empresas implica que las mismas son integradas a través de la definición y ejecución en forma conjunta de los procesos de negocio colaborativos.

En este capítulo se han descrito varios modelos de colaboración, los cuales tienen como objetivo integrar a las empresas desde el punto de vista de los negocios. La principal diferencia entre estos modelos se debe a la política de gobierno o control que utilizan para llevar a cabo dicha colaboración, la cual puede ser centralizada o descentralizada. Aquellos modelos de colaboración que proponen una gestión descentralizada posibilitan que las empresas puedan alcanzar mayores beneficios en la gestión integrada de la cadena de suministro mientras preservan sus autonomías. En este sentido, se ha propuesto el modelo de colaboración socio-a-socio, el cual surge con el objetivo de gestionar en forma descentralizada relaciones de colaboración entre empresas de producción.

Los modelos de colaboración sólo pueden ser implementados a través de colaboraciones B2B. El objetivo de estas relaciones B2B es integrar a las empresas desde el punto de vista de los sistemas de información. Dos posibles soluciones pueden ser usadas: Sistemas e-Marketplaces o Sistemas de Información P2P. Ambos son más adecuados para ciertos modelos de colaboración. No obstante, colaboraciones B2B soportadas por sistemas de información P2P permiten a las empresas alcanzar los niveles más altos de integración, de autonomía, de gestión descentralizada de los procesos, y de personalización de sus procesos de negocio. A su vez, este tipo de relaciones B2B permiten a los socios de negocio alcanzar mayores beneficios.

Como contraparte de estos beneficios, este tipo de relaciones B2B tiene como desventaja que las empresas deben incurrir en altos costos para el desarrollo e implementación de estas soluciones. No obstante, estos costos pueden ser disminuidos en forma considerada si se provee la tecnología de información adecuada, que permita una rápida implementación de los mismos. Los métodos, lenguajes, técnicas y herramientas, propuestos en esta tesis han sido desarrollados con este propósito, con el objetivo de que las empresas puedan implementar, con el menor esfuerzo posible, colaboraciones B2B avanzadas y beneficiosas, basadas en modelos de colaboración descentralizados y sistemas de información P2P.

Capítulo

3

HACIA EL DESARROLLO CONDUCIDO POR MODELOS DE PROCESOS DE NEGOCIO COLABORATIVOS

En este capítulo, en primer lugar se describen los conceptos involucrados en el nivel de negocio y el nivel tecnológico de una colaboración B2B. Luego se identifican los requerimientos para dar soporte al diseño conceptual de procesos de negocio colaborativos, como así también para la generación de especificaciones ejecutables de estos procesos y de las interfaces de los socios en un estándar B2B específico. A continuación se describe el método, basado en el desarrollo conducido por modelos, propuesto para el desarrollo de procesos colaborativos. Se introducen sus principales componentes y las técnicas propuestas que forman parte de este método. Finalmente, se discuten trabajos relacionados y se presentan las conclusiones.

3.1. Modelo Conceptual de una Colaboración B2B

Como se discutió en el capítulo anterior, para llevar a cabo una gestión integrada de la cadena de suministro, las empresas deben establecer colaboraciones B2B basadas en la aplicación de modelos de colaboración descentralizados soportados por sistemas de información Peer-to-Peer (P2P). El principal desafío en el desarrollo de este tipo de relaciones entre empresas es disminuir su complejidad, como así también los tiempos y costos implicados.

Para abordar este desafío, en primer lugar se deben considerar dos niveles de abstracción en el desarrollo de una colaboración B2B: el nivel de negocio, el cual refiere a la vista que los analistas de negocio y/o los diseñadores de sistemas tienen de la solución; y el nivel tecnológico, el cual refiere a la vista que los arquitectos y desarrolladores del sistema de información B2B tienen de la solución. La Figura 3-1 muestra un modelo conceptual de una colaboración B2B que describe los principales conceptos involucrados en estos niveles.

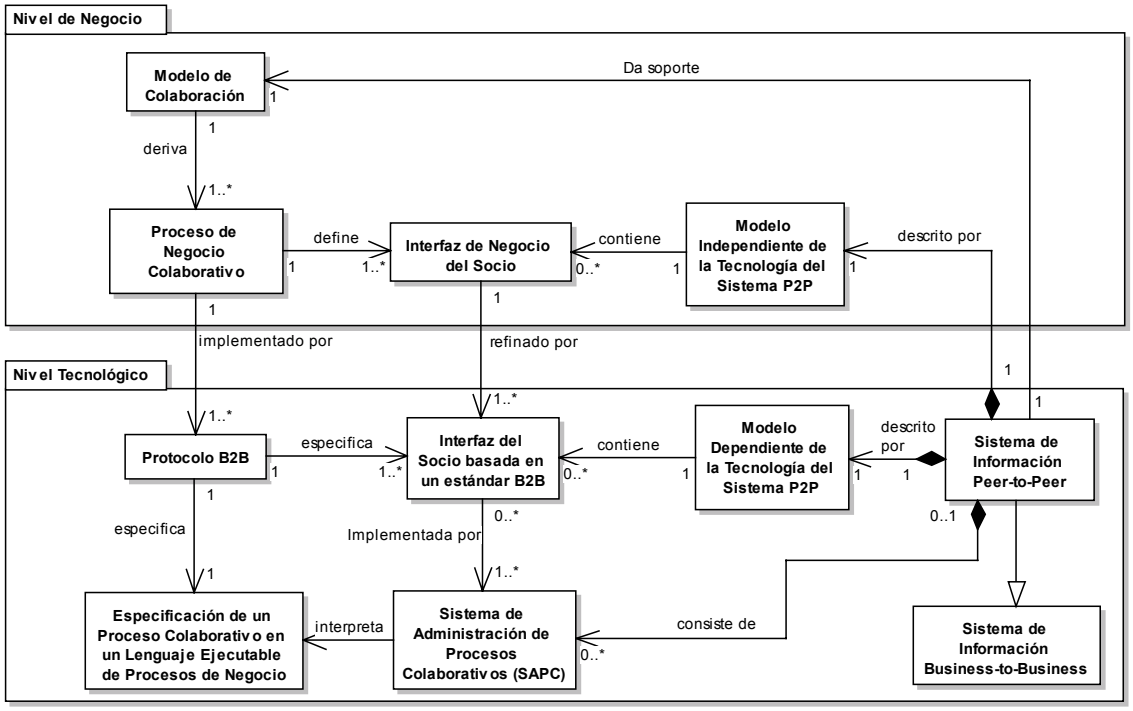


Figura 3-1. Modelo conceptual de una colaboración B2B

3.1.1 Conceptos del Nivel de Negocio

En el nivel de negocio, las empresas o socios de negocio acuerdan el *modelo de colaboración* a utilizar para llevar a cabo una gestión integrada de la cadena de suministro. Este es el primer paso en el desarrollo de una colaboración B2B.

Aunque el modelo explica la manera en que las empresas llevarán a cabo la colaboración, la definición explícita y detallada de cómo será su comportamiento es formalizada en los *procesos de negocio colaborativos*. Uno o más procesos colaborativos serán derivados y definidos respetando los lineamientos generales del modelo de colaboración. La definición de estos procesos debe ser realizada y acordada en forma conjunta por los socios. Un proceso colaborativo no sólo determina la información a ser intercambiada, sino también el comportamiento que tendrán las interacciones entre los socios y las responsabilidades de cada uno en dicho proceso. Mediante la ejecución de estos procesos las empresas podrán tomar decisiones en forma conjunta para alcanzar metas comunes, coordinar sus acciones e intercambiar información. Por lo tanto, el diseño de los procesos colaborativos es un objetivo clave para las empresas que participan en una colaboración B2B.

Debido a que un proceso colaborativo es gestionado conjuntamente entre las partes

en forma descentralizada (Chen y Hsu, 2001), éste tiene asociado *interfaces de negocio* que corresponden a cada uno de los socios involucrados en el mismo. Dichas interfaces determinan los servicios de cada socio requeridos para el envío y recepción de mensajes en un proceso. En consecuencia, cada socio definirá la interfaz que le posibilitará desempeñar el rol que tiene asignado en la colaboración. Estas interfaces son derivadas a partir de los procesos colaborativos compartidos y acordados entre los socios (Dayal y otros, 2001).

En el nivel de negocio, las empresas deben enfocarse en la definición de los procesos colaborativos y las interfaces de los socios, sin considerar la tecnología a ser utilizada para su implementación. Este enfoque es conveniente porque, en primer lugar, los analistas de negocio y/o diseñadores de sistemas de información que intervienen, no conocen o no desean compenetrarse en este nivel con los detalles de la solución tecnológica a usar en la implementación de los mismos. En segundo lugar, si bien existen diferentes soluciones tecnológicas para su implementación, el desarrollo de procesos colaborativos no debería ser conducido por una tecnología específica (Baghdadi, 2004). La tecnología de implementación debería ser decidida luego de la definición de los mismos.

De esta manera, las interfaces de negocio de los socios forman parte del *modelo independiente de la tecnología* que describe al *sistema de información P2P* en el nivel de negocio. Éste es un tipo específico de *sistema de información B2B* a través del cual se dará soporte al modelo de colaboración y a los procesos colaborativos. En adelante ambos conceptos, sistemas de información P2P y B2B, serán referidos indistintamente en esta tesis. Un sistema de información P2P está compuesto por *sistemas de administración de procesos colaborativos (SAPC)* autónomos, heterogéneos y distribuidos, que cada empresa debe implementar (Haller y Schuldt, 2003). En el nivel de negocio, los SAPC se corresponden con las interfaces de negocio de los socios.

3.1.2 Conceptos del Nivel Tecnológico

En el nivel tecnológico, las empresas deben enfocarse en la tecnología de información a utilizar para dar soporte a la ejecución de los procesos colaborativos. En este nivel, el mayor desafío es posibilitar las interacciones entre los SAPC que forman parte del sistema de información B2B. Estas interacciones consisten en el intercambio de mensajes como parte de la ejecución de los procesos colaborativos, y tienen lugar en

tres capas (Medjahed y otros, 2003): la de comunicación, la de contenido y la de procesos de negocio.

La *capa de comunicación* trata con la interoperabilidad desde el punto de vista de los protocolos de comunicación a ser utilizados (por ejemplo: HTTP, SOAP, etc.), a través de los cuales los SAPC intercambiarán mensajes vía Internet.

La *capa de contenido* trata con la interoperabilidad respecto a la información (documentos de negocio) a ser intercambiada entre los socios. Esta información debe ser definida y organizada de tal manera que pueda ser entendida y utilizada de la misma forma por los socios. Las interacciones en esta capa requieren que los SAPC entiendan la sintaxis y semántica de los documentos de negocio (Caliusco y otros, 2004b).

La *capa de procesos de negocio* trata con las interacciones entre los servicios que forman parte de las interfaces de los socios. La interoperabilidad entre estas interfaces para dar soporte a los procesos colaborativos es una cuestión importante y requiere (Medjahed y otros, 2003) (Bernauer y otros, 2003a):

- Especificar en forma explícita las interfaces de los SAPC de los socios.
- Especificar la semántica de los procesos colaborativos en un formato que pueda ser procesado y entendido por los SAPC.

Estos requerimientos de interoperabilidad son soportados por los estándares B2B, los cuales proveen un conjunto de lenguajes para la especificación de los denominados *Protocolos B2B* (Bussler, 2002) (Bernauer y otros, 2003a). Estos protocolos dan soporte a la interoperabilidad entre los SAPC en cada una de las capas anteriores, posibilitando el intercambio de mensajes basados en XML (eXtensible Markup Language) (W3C, 1997) entre los mismos.

Un protocolo B2B permite definir varios aspectos de las *Interfaces de los Socios basadas en un Estándar B2B*, tales como el protocolo de transporte, los tipos de documentos, requerimientos de seguridad, etc. (Bussler, 2002). Dichas interfaces forman parte del *Modelo Dependiente de la Tecnología del Sistema P2P*. Los servicios que las mismas proveen deben corresponderse con los servicios especificados en las interfaces de negocio. Cada uno de los socios debe proveer su propio SAPC que implementa las interfaces correspondientes, para posibilitar el intercambio de los mensajes basados en XML, de acuerdo a lo definido en los protocolos B2B.

Por otro lado, un protocolo B2B permite definir la lógica de los procesos colaborativos a través de su especificación en un *lenguaje ejecutable de procesos de negocio* provisto por el estándar B2B. Estos lenguajes, que en su mayoría están basados en XML, permiten especificar un proceso en un formato que puede ser procesado e interpretado por software. Los SAPC deben poder interpretar estas especificaciones con el objetivo de ejecutar, controlar, y monitorear los procesos colaborativos, intercambiando mensajes basados en XML. De esta forma, se cumple con el importante requisito de las colaboraciones B2B de llevar a cabo una gestión descentralizada de los procesos colaborativos a través de interacciones P2P entre SAPC autónomos, distribuidos y heterogéneos (Villarreal y otros, 2003c). La implementación de cada uno de los SAPC puede ser realizada en forma independiente (Villarreal y otros, 2003b). Esto significa que pueden ser construidos de diferentes maneras usando diferentes tecnologías y plataformas. La única restricción es que deben interoperar usando protocolos B2B. El SAPC no sólo debe gestionar los procesos colaborativos sino que también debe integrar estos procesos con los procesos internos o privados del socio. Los detalles de cómo se realiza esta integración son aspectos privados de cada uno de los socios, por lo cual no deben ser tenidos en cuenta en el diseño de los procesos colaborativos.

Por lo tanto, un proceso colaborativo es implementado a través de un protocolo B2B, que contiene la especificación del proceso en un lenguaje ejecutable y la especificación de las interfaces correspondientes a los SAPC de los socios. En este nivel, los socios deben seleccionar y acordar el estándar B2B que consideren más adecuado para la implementación de los mismos. Además, un proceso colaborativo puede ser implementado por más de un protocolo B2B, cada uno basado en un estándar diferente.

3.1.3 Representación Esquemática de una Colaboración B2B

En resumen, para el desarrollo de colaboraciones B2B, es necesario primero dar soporte al diseño de procesos colaborativos en el nivel de negocio, esto es el dominio del problema, y luego dar soporte a la definición de los mismos en el nivel tecnológico, esto es el dominio de la solución, a través de la generación de las especificaciones de estos procesos y de las interfaces de los SAPC, ambos basados en un estándar B2B.

La Figura 3-2 representa un esquema de una colaboración B2B entre dos empresas. En el nivel de negocio, estas empresas acuerdan el modelo de colaboración a ser aplicado para la gestión integrada de la cadena de suministro. Luego, diseñan y definen los procesos colaborativos necesarios para gestionar la colaboración. Posteriormente, a partir de estos procesos, se derivan las interfaces de negocio de cada empresa con sus servicios correspondientes.

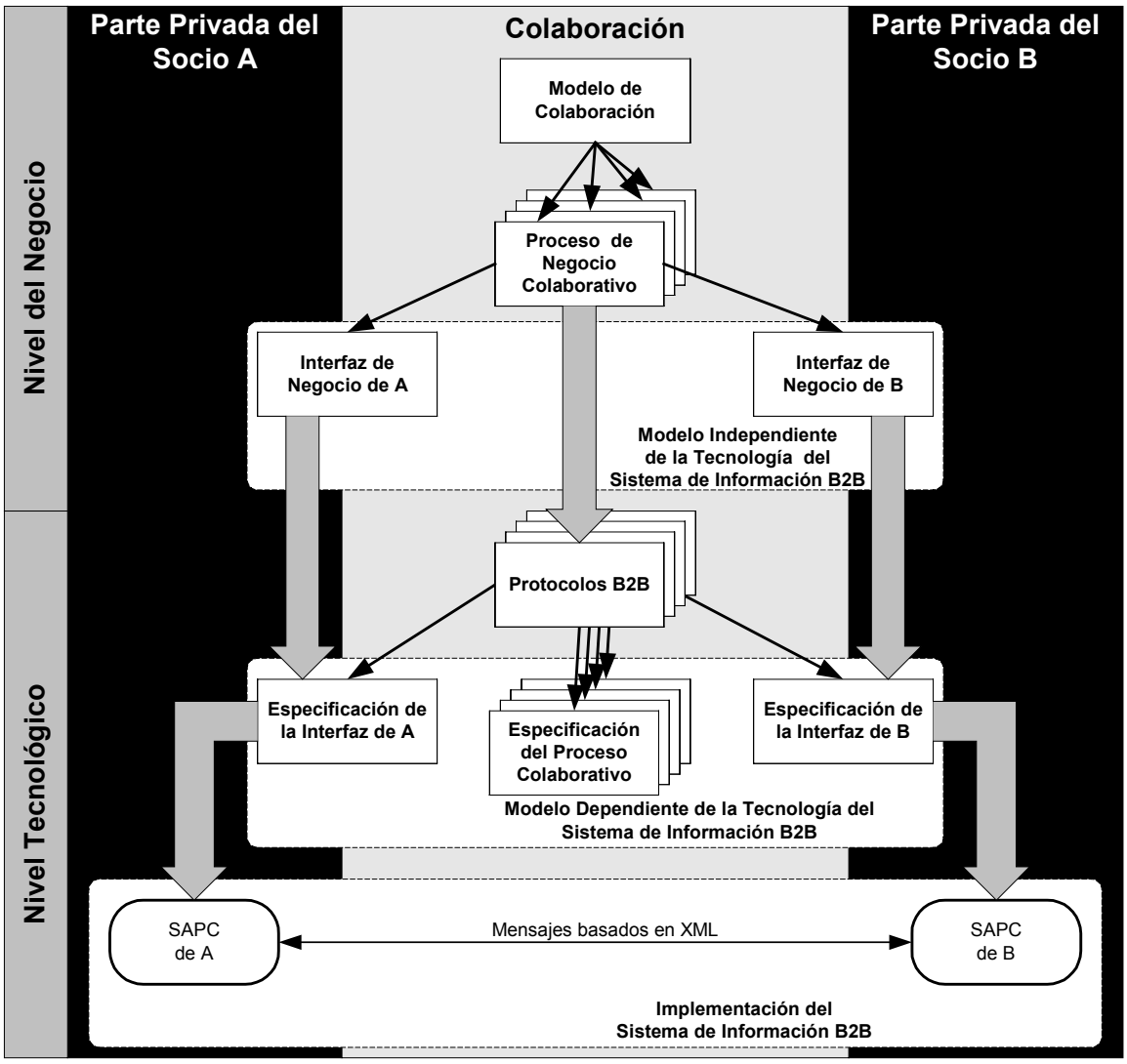


Figura 3-2. Ejemplo de una colaboración B2B

Una vez definidos los elementos del nivel de negocio, las empresas acuerdan el estándar B2B a utilizar y definen los protocolos B2B. La definición de los procesos colaborativos es realizada de acuerdo al lenguaje ejecutable de procesos de negocio que provea el estándar y se generan las interfaces de los SAPC también conforme a dicho

estándar. Estas interfaces son un refinamiento de las interfaces de negocio definidas anteriormente.

Posteriormente, para que la colaboración pueda hacerse efectiva, las empresas desplegarán los SAPC que implementan las interfaces correspondientes y que posibilitan la gestión conjunta de los procesos colaborativos. Estos componentes pueden estar basados en diferentes plataformas provistas por diferentes proveedores de software.

3.2. Requerimientos para el Desarrollo de Procesos Colaborativos

De acuerdo a lo expresado anteriormente, se puede inferir que el principal concepto en una colaboración B2B es el de proceso de negocio colaborativo independiente de la tecnología. Esta inferencia se basa en que a partir de estos procesos:

- Las empresas definen y acuerdan el comportamiento de la colaboración y los roles que cada una debe desempeñar.
- Es posible derivar las interfaces de negocio de los socios.
- Es posible derivar las especificaciones de los mismos, como así también las especificaciones de las interfaces de los SAPC, en base a un estándar B2B.

De esta forma, el diseño de procesos colaborativos en el nivel de negocio es una tarea clave para establecer colaboraciones B2B. Para ello, un importante requerimiento es el uso de modelos conceptuales que representen procesos colaborativos independientes de la tecnología.

En el nivel tecnológico, las especificaciones de los procesos colaborativos y las interfaces de los SAPC en un estándar B2B, deben tener una correspondencia mutua con los procesos y las interfaces definidas en el nivel de negocio. Esto significa que las definiciones en ambos niveles deben ser consistentes entre sí. Para ello, es necesario que las especificaciones de los procesos y las interfaces en un estándar B2B específico sean derivadas en forma automática, a partir de los modelos conceptuales. Varios trabajos (Baghdadi, 2004) (Bernauer y otros, 2003a) han reconocido la necesidad de proveer métodos o guías tanto para el diseño conceptual de los procesos colaborativos, como

para la derivación automática de las especificaciones de dichos procesos y de las interfaces de los socios en un estándar B2B.

Por lo tanto, con el propósito de que las empresas puedan implementar, de manera rápida y con el mínimo esfuerzo posible, colaboraciones B2B para dar soporte a modelos de colaboración descentralizados, es necesario proveer un *método sistemático de desarrollo de procesos de negocio colaborativos*. Dicho método debe dar soporte a tres etapas: análisis, diseño y especificación. La etapa de *análisis* trata con la captura de requerimientos y la identificación de los procesos a partir del modelo de colaboración seleccionado. La etapa de *diseño* trata con la definición del comportamiento de los procesos colaborativos, es decir sus flujos de control. Estas dos etapas comprenden la modelación de dichos procesos en forma independiente de la tecnología. La última etapa, la de *especificación*, comprende la especificación de dichos procesos y de las interfaces de los SAPC, en un estándar B2B específico. Luego, el objetivo de tal método debería ser disminuir el tiempo y la complejidad en el desarrollo de procesos colaborativos, a través de la provisión de guías, lenguajes, técnicas y herramientas que permitan llevar a cabo dos tareas fundamentales:

- *Modelado Conceptual de Procesos Colaborativos Independientes de la Tecnología*, para dar soporte a las etapas de análisis y diseño de estos procesos, sin considerar las idiosincrasias de un estándar B2B específico.
- *Generación Automática de Especificaciones de los Procesos Colaborativos y de las Interfaces de los SAPC en un estándar B2B*, para dar soporte a la etapa de especificación. Dichas especificaciones deben ser generadas a partir de los modelos conceptuales, con el objetivo de mantener consistencia entre lo definido en el nivel de negocio y lo definido en el nivel tecnológico.

En las siguientes secciones se describen los requerimientos particulares de cada una de estas dos tareas.

3.2.1 Requerimientos para el Modelado Conceptual de Procesos Colaborativos

En primer lugar, para dar soporte al modelado conceptual de procesos colaborativos, se debe contar con un lenguaje de modelado que posea las siguientes propiedades:

- Fácil de usar y comprender por los analistas de negocio y los diseñadores de sistemas que no conocen, o no están interesados, en los lenguajes provistos por los estándares B2B.
- Posibilite a las empresas definir los procesos colaborativos usando una semántica común, que les permita entender de la misma manera estos procesos.
- Provea abstracciones conceptuales adecuadas para el dominio de las colaboraciones B2B.

Con respecto a este último aspecto, las características de las colaboraciones B2B imponen ciertos requerimientos para el modelado conceptual de procesos de negocio colaborativos, los cuales deberían ser satisfechos por el lenguaje de modelado. Estos requerimientos son discutidos a continuación.

3.2.1.1. Etapas o Fases de las interacciones B2B

Las interacciones entre empresas en la gestión de la cadena de suministro a través de colaboraciones B2B siguen diferentes fases de acuerdo al tipo de relación que se está intentado implementar (Alt y otros, 2000). Algunos trabajos (Schoop y otros, 2002) (Grieger, 2003b) han identificado fases para relaciones B2B basadas en e-marketplaces, en donde las interacciones son de corta duración y los socios no son conocidos de antemano sino que deben ser encontrados en el e-marketplace.

Las colaboraciones B2B que son de interés de esta tesis, implican relaciones de larga duración y socios que se conocen de antemano. En estas relaciones, las interacciones entre empresas siguen cuatro fases: información, contratación o acuerdo, negociación y cumplimiento.

En la *fase de información* las empresas eligen entre una variedad de productos y socios. Esta fase se realiza generalmente de modo informal y no es primordial disponer de un soporte informático como en el caso de relaciones basadas en un e-Marketplace. Generalmente, las relaciones de colaboración de largo plazo son creadas cuando las empresas han estado realizando transacciones comerciales durante un tiempo, y por lo tanto ya se conocen.

En la *fase de contratación*, los socios definen un acuerdo de colaboración. Dicho acuerdo implica un contrato a largo plazo, por el cual compartirán información y

llevarán a cabo la ejecución de procesos colaborativos. En el acuerdo se establecen el modelo de colaboración que se aplicará y los productos que se tomarán en cuenta, y se definen los parámetros y reglas requeridos por el modelo (Villarreal y otros, 2003a).

En la *fase de negociación*, las partes llevan a cabo negociaciones y toman decisiones en forma conjunta. Como ejemplo, en esta fase las empresas pueden negociar y acordar pronósticos de demanda, planes de producción, planes de órdenes, etc.

Finalmente, en la *fase de cumplimiento*, los compromisos asumidos en la fase de negociación son cumplidos. Por ejemplo, los productos acordados en las órdenes son entregados al cliente, quien luego realiza el pago de los mismos. Estos cumplimientos requieren el intercambio de información entre empresas a través de transacciones comerciales.

Para que una colaboración B2B se haga efectiva, es necesario que al menos una de estas fases esté soportada por tecnología de información. Aunque todas las fases podrían ser soportadas por procesos colaborativos, las últimas dos fases son las más apropiadas para ello debido a que es posible estructurar las interacciones entre empresas y por lo tanto es posible automatizarlas. Por lo tanto, estos procesos son definidos tanto para llevar a cabo negociaciones entre las partes como para intercambiar información acerca del cumplimiento de lo acordado en las negociaciones.

Las interacciones entre empresas en la fase de contratación son semiestructuradas. Generalmente son llevadas a cabo de modo informal a través de reuniones personales entre representantes de las empresas y por lo tanto son difíciles de automatizar. Algunos autores (Angelen y Grefen, 2002) proponen dar soporte informático al proceso de contratación. No obstante, este tipo de proceso está fuera del alcance de esta tesis.

Debido a que los procesos son definidos teniendo en cuenta lo acordado en la fase de contratación, desde el punto de vista conceptual, es necesario capturar y modelar tanto los detalles del acuerdo en la fase de contratación, como los procesos colaborativos de la fase de negociación y cumplimiento.

3.2.1.2. Vista Global de las Interacciones entre los Socios

Un modelo de proceso colaborativo debe expresar la vista global de las interacciones entre los socios, con el objetivo de describir el comportamiento público esperado en dichas interacciones. Esto se refiere a que las responsabilidades de los roles

(que cada socio juega en un proceso), en términos de la información a enviar o recibir por cada uno, deben estar definidas en forma explícita en los procesos. Un proceso colaborativo no debe ser definido desde el punto de vista de un único rol desempeñado por un socio, sino que debe describir la vista de cada uno de los roles. Esta vista global posibilita que los socios entiendan y visualicen de la misma forma un proceso, y conduce al establecimiento de confianza entre ellos.

3.2.1.3. Autonomía de las Empresas

En colaboraciones B2B, se asume que cada empresa participante es autónoma (Chen y Hsu, 2001) (Medjahed y otros, 2003). Esto significa que además de las metas comunes a alcanzar con sus socios, cada empresa tiene sus propias metas y por lo tanto tiene el control de sus acciones internas y puede decidir diferentes cursos de acción en las interacciones con sus socios. Por lo tanto, esta autonomía debe ser contemplada en un modelo de proceso colaborativo. Esto significa que aquellas actividades internas de cada empresa, requeridas para el procesamiento de la información recibida desde sus socios o para la generación de la información a enviar a sus socios, no deben ser definidas en un modelo de proceso colaborativo, debido a que forman parte de los aspectos privados de la empresa (Villarreal y otros, 2003b). Cada empresa puede definir de diferente forma el flujo de control y las interdependencias entre estas actividades privadas como parte de la solución interna (procesos de negocio privados) que cada una implementará para dar soporte a los procesos colaborativos.

De esta manera, los procesos colaborativos deben ser definidos en forma completamente independiente de los procesos privados de las empresas que dan soporte a los mismos. Esto tiene por objetivo permitir un débil acoplamiento entre estos procesos y una mayor autonomía de las empresas (Bussler, 2001).

3.2.1.4. Gestión Descentralizada e Interacciones Peer-to-Peer

La gestión descentralizada de los procesos colaborativos puede ser alcanzada a través de interacciones P2P entre los socios, en donde cada uno gestiona el rol que va a desempeñar en la colaboración B2B (Chen y Hsu, 2001) (Villarreal y otros, 2003b). Por lo tanto, estas interacciones también deberían ser expresadas en un modelo de proceso colaborativo. Esto se correlaciona con el requerimiento de dar soporte a una vista global

del proceso, debido a que sólo con esta vista es posible describir interacciones P2P entre los socios.

3.2.1.5. Procesos Colaborativos como Procesos Abstractos

Un proceso colaborativo es un proceso abstracto (Baghdadi, 2004). Esto significa que la ejecución de un proceso colaborativo no puede ser realizada en forma directa, sin que cada socio provea la lógica necesaria para la gestión del mismo. La lógica detrás de la creación o procesamiento de un mensaje a ser intercambiado, como así también la lógica para determinar el envío de uno u otro mensaje, es soportada por los procesos privados de cada socio. Esto significa que para la ejecución de un proceso colaborativo es necesario que cada empresa provea la implementación de los procesos y actividades internas necesarias para dar soporte a los mensajes recibidos desde los socios o a los mensajes a enviar hacia los socios. Esta implementación es realizada a través de procesos privados que gestionan la lógica local requerida para soportar el proceso colaborativo. El SAPC de cada socio es encargado de realizar esta tarea.

Por lo tanto, un proceso colaborativo es una abstracción, cuya ejecución está a cargo de varios SAPC, los cuales intercambian mensajes y gestionan el rol que un socio desempeña en el proceso. Cada socio es responsable de la gestión de la parte que le corresponde en el proceso colaborativo.

3.2.1.5. Derivación de las Interfaces de los Socios

Así como un proceso colaborativo que define el comportamiento de las interacciones entre los socios constituye la vista dinámica de una colaboración B2B, las interfaces de los socios que definen los mensajes a enviar y recibir constituyen la vista estática de la colaboración. Estas interfaces, que soportan el rol de cada socio en los procesos, deben ser derivadas a partir de los modelos de procesos colaborativos (Dayal y otros, 2001). Esto garantiza que dichas interfaces sean interoperables, estén definidas conforme a los procesos y por lo tanto no sean ambiguas.

3.2.1.6. Perspectivas de un Modelo de Proceso Colaborativo

Existen diferentes perspectivas que todo modelo de proceso de negocio debería poseer (Jablonsky y Bussler, 1996). Si bien estas perspectivas han sido consideradas para evaluar procesos de negocio como workflows, la mayoría de las mismas deberían

ser tenidas en cuenta para el modelado de procesos colaborativos. La *perspectiva funcional* describe qué es lo que tiene que ser realizado en un proceso. Esto significa, las actividades, subprocesos e información de entrada y salida del proceso. La *perspectiva de comportamiento* describe cuándo las actividades tienen que ser ejecutadas. Esto es definido por el flujo de control de las actividades de un proceso. También describe las restricciones de tiempo sobre la duración de las actividades y del proceso, y el manejo de excepciones que puedan ocurrir y necesiten ser manejadas en el mismo. La *perspectiva de información* comprende los datos e información intercambiada entre las actividades. La *perspectiva organizacional* describe quién participa en el proceso y cuáles son los roles que los participantes desempeñan en el mismo. La *perspectiva operacional* describe cómo las actividades son implementadas, es decir, qué aplicaciones son requeridas para la ejecución de las mismas. Esta perspectiva no debe ser expresada en un modelo de proceso colaborativo debido a que quebranta el requerimiento de autonomía de las empresas.

En (Bernauer y otros, 2003b) los autores distinguen otra perspectiva adicional, la *perspectiva transaccional*. Ésta describe las partes del proceso que requieren de propiedades transaccionales. Las transacciones entre organizaciones son diferentes a las transacciones utilizadas en sistemas de gestión de workflow o sistemas de gestión de base de datos, debido a que se requiere una semántica de bajo acoplamiento para transacciones de larga duración.

3.2.1.7. Soporte para Procesos de Negociación

En la gestión de la cadena de suministro a través de modelos de colaboración, las empresas necesitan acordar pronósticos de demanda, planes de producción, planes de órdenes, planes de aprovisionamiento, etc. Esto requiere de negociaciones complejas entre las partes. Una negociación electrónica es un proceso de toma de decisiones conjunta entre dos o más partes (Rebstock y Thun, 2003). El objetivo de este proceso es establecer un contrato o compromiso entre las partes con respecto a cierta información compartida. Dentro de este proceso las partes llevan a cabo negociaciones a través del intercambio de propuestas, las cuales serán aceptadas para alcanzar un compromiso o bien serán rechazadas. De esta manera, una colaboración B2B es más que el mero intercambio de mensajes, ya que implica que las empresas definan compromisos y

acuerdos mediante el intercambio de mensajes para poder tomar decisiones en forma conjunta.

Debido a que los procesos de negociación requieren de un soporte tecnológico, los mismos pueden ser definidos como procesos colaborativos. Por lo tanto, los modelos conceptuales de estos procesos requieren de abstracciones conceptuales adecuadas que posibiliten la definición de propuestas, aceptaciones, rechazos, etc., para definir negociaciones complejas entre los socios.

3.2.2. Requerimientos para la Generación Automática de Especificaciones de Procesos Colaborativos basados en un Estándar B2B

Como se describió en la sección 3.1.2, los procesos colaborativos pueden ser implementados utilizando protocolos B2B, los cuales soportan la interoperabilidad entre los SAPC en las capas de procesos de negocio, contenido y transporte (Bussler, 2002). Algunos estándares B2B, como por ejemplo xCML[1] y OAGIS[2], dan soporte únicamente a la interoperabilidad en la capa de contenido (Medjahed y otros, 2003). No obstante, para dar soporte a la ejecución de procesos colaborativos, se requiere de un estándar que permita especificar los elementos de un protocolo B2B en las diferentes capas de interoperabilidad.

Actualmente existen dos tipos de tecnologías propuestas para la especificación completa de protocolos B2B (Bernauer y otros, 2003a), las cuales en esta tesis se clasifican en: estándares basados en composición de servicios web, tales como *Business Process Execution Language for Web Services (BPEL)* (BEA y otros, 2003) y *Web Service Choreography Interface (WSCI)* (W3C, 2002); y estándares basados en transacciones de negocio, tales como *Electronic Business using eXchange Markup Language (ebXML)* (OASIS, 1999) y *RosettaNET* (RosettaNet, 1999).

Los estándares basados en composición de servicios web usan como lenguaje base, el lenguaje *Web Service Description Language (WSDL)* (W3C, 2001), el cual permite definir las interfaces de los socios como servicios web desacoplados. Los servicios web proveen un nuevo paradigma para computación distribuida, el cual es una evolución de

[1] Commerce XML (cXML) http://www.cxml.org/

[2] OAGIS, http://www.openapplications.org

los paradigmas de orientación a objetos y de componentes (Tsalgatidou y Pilioura, 2002). Un servicio web es un componente independiente, débilmente acoplado, que puede ser implementado en diferentes lenguajes de programación y plataformas, y define un conjunto de operaciones que pueden ser invocadas remotamente a través del intercambio de mensajes basados en XML sobre un protocolo de Internet como HTTP o SOAP.

Por encima de WSDL, lenguajes como BPEL y WSCI permiten definir la coreografía de servicios web como parte de un servicio web compuesto. Estos lenguajes de composición de servicios web han sido desarrollados con el objetivo de dar soporte tanto a la definición de procesos privados de los socios, como a la definición de procesos colaborativos. La actividad de definir procesos privados basados en servicios web se conoce como *orquestación de servicios web*, mientras la actividad de definir procesos colaborativos se conoce como *coreografía de servicios web* (Peltz, 2003). Esta última consiste en definir protocolos de conversación, también conocidos como protocolos de negocio, que definen la secuencia de mensajes entre un cliente y el servicio como parte de la invocación de un servicio web, sin revelar la lógica de negocio interna del servicio. Los estándares BPEL y WSCI dan soporte a la definición de los protocolos de conversación, los cuales soportan la especificación de la mayoría de los elementos de un protocolo B2B definidos anteriormente, excepto la estructura de los documentos de negocio. Es decir, no dan soporte a la capa de contenido, para lo cual es posible utilizar algún otro estándar que de soporte a la misma.

Por otra parte, los estándares basados en transacciones de negocio usan el concepto de *transacción de negocio* para referirse a interacciones de tipo solicitud/respuesta, entre dos roles autónomos (sin especificar concretamente el socio), los cuales usan diferentes recursos para ejecutar las actividades que permiten el intercambio de información. Una transacción de negocio es la unidad atómica de trabajo en un proceso colaborativo y posee varias características que la distinguen de una transacción tradicional de base de datos (Yang y Papazoglou, 2000). Principalmente, es gobernada por tipos no convencionales de atomicidad y su duración puede ser de varios días.

Dentro de los estándares basados en transacciones de negocio, RosettaNet nació con el objetivo de dar soporte al intercambio de documentos entre empresas involucradas en una cadena de suministro de la industria electrónica. Aunque soporta la

definición de la mayoría de los elementos de un protocolo B2B, no provee un lenguaje para especificar procesos colaborativos. En su lugar, provee los denominados Partner Interface Processes (PIPs), los cuales representan una transacción de negocio, pero no provee soporte para definir el orden de las transacciones, por lo cual no es posible definir procesos colaborativos.

El estándar ebXML nació con el objetivo de proveer un conjunto de lenguajes específicamente dirigidos para la definición de todos los elementos de un protocolo B2B. En la capa de procesos de negocio, la cual es de interés de esta tesis, ebXML provee dos lenguajes: *Business Process Specification Schema (BPSS)* (UN/CEFACT and OASIS, 2001) y *Collaborative Protocol Profile and Agreement (CPPA)* (OASIS, 2002). BPSS es utilizado para describir procesos colaborativos como colaboraciones de negocio, las cuales definen una coreografía de transacciones de negocio. CPPA es utilizado para especificar los detalles operacionales y los parámetros de configuración de las interfaces de los socios con respecto a los procesos definidos con BPSS.

Los estándares basados en composición de servicios web y los basados en transacciones de negocio no son compatibles entre sí. Además, estos estándares están aún en su etapa de evolución por lo cual, actualmente existen varias versiones de los mismos. En consecuencia, la generación automática de especificaciones B2B basadas en estos estándares requiere de un procedimiento particular para cada versión de cada estándar.

Por otra parte, como consecuencia de las distintas soluciones posibles para procesos colaborativos, el problema de interoperabilidad ha cambiado desde el nivel de aplicaciones al nivel de estándares (Medjahed y otros, 2003). Esto significa que un socio debería ser capaz de establecer colaboraciones B2B con varios socios usando estándares diferentes. Más aún, un socio podría requerir la implementación de un mismo proceso colaborativo con varios socios usando diferentes estándares. Este problema puede ser resuelto en tiempo de diseño, definiendo procesos colaborativos independientes de la tecnología y reutilizarlos para generar soluciones basadas en diferentes estándares.

En resumen, para generar en forma automática especificaciones B2B de los procesos colaborativos, es necesario de un método que:

- Soporte la definición de transformaciones de modelos de procesos colaborativos independientes de la tecnología, en especificaciones de dichos procesos y de las interfaces de los socios en un estándar B2B específico. De esta manera, las empresas pueden seleccionar distintas soluciones de implementación y trasladar sus diseños de alto nivel a diferentes estándares y versiones de los mismos.
- Permita ejecutar en forma automática dichas transformaciones, para disminuir los tiempos y costos de desarrollo de la solución tecnológica.

Con el propósito de dar respuesta a estas necesidades, en el marco de esta tesis se propone un Método para el Desarrollo de Procesos Colaborativos, el cual se describe en la siguiente sección.

3.3. Método de Desarrollo Conducido por Modelos de Procesos Colaborativos

Según se describió en las secciones precedentes de este capítulo, los modelos de procesos colaborativos definidos en el nivel de negocio son una parte clave en el desarrollo de colaboraciones B2B, y debido a la necesidad de poder generar en forma automática a partir de estos modelos las especificaciones de los procesos y de las interfaces de los socios, se considera al *Desarrollo Conducido por Modelos* (Model-Driven Development, MDD) como una práctica de diseño adecuada para ser explotada en un método que soporte todas las etapas del desarrollo de procesos colaborativos.

Precisamente, la principal característica del desarrollo conducido por modelos es que los modelos juegan un rol importante en el desarrollo de software, ya que los productos o artefactos principales son los modelos en lugar de los programas (Selic, 2003). La ventaja de esto es que es posible expresar modelos usando conceptos menos ligados a la tecnología de implementación y más cercanos al dominio del problema. Esto permite que los modelos sean más fáciles de especificar, entender y mantener. Por otra parte, permite que los modelos sean menos sensibles a la tecnología de implementación y a los cambios en dicha tecnología. En consecuencia, el concepto de modelos independientes de la tecnología está muy relacionado con el desarrollo conducido por modelos.

Otra premisa importante de esta práctica de diseño es que el código sea generado automáticamente a partir de sus correspondientes modelos (Selic, 2003). De esta

manera, los modelos no sólo son utilizados para documentar el sistema sino también para construir el producto final. Esto es diferente al enfoque tradicional de desarrollo de software en donde los modelos eran meramente utilizados para propósitos de documentación, ya que los mismos tenían un valor muy limitado debido a que la documentación generalmente diverge de la realidad o no existe una garantía de correspondencia.

Los beneficios señalados pueden ser aprovechados para dar soporte a los principales requerimientos identificados anteriormente (sección 3.2). Por lo cual, la solución propuesta consiste de: un método basado en el desarrollo conducido por modelos que soporta las etapas del desarrollo de procesos colaborativos, junto con la provisión de un conjunto de técnicas necesarias que cumplen con la filosofía del desarrollo conducido por modelos (Villarreal y otros, 2005). Por un lado, este método tiene como objetivo permitir que el diseño de los procesos colaborativos se realice desde el punto de vista del dominio del problema, es decir, en el nivel de negocio, posibilitando una mayor abstracción y permitiendo que la tecnología a utilizar para su implementación pueda ser decidida posteriormente. Por otro lado, el objetivo es reducir la complejidad inherente, los tiempos y los costos de desarrollo. A través de este método los analistas de negocio y diseñadores de sistemas pueden construir y transformar modelos de procesos colaborativos con el objetivo de generar automáticamente el código XML, correspondiente a las especificaciones de los procesos colaborativos y de las interfaces de los socios, de acuerdo a un estándar B2B.

3.3.1. Arquitectura Conducida por Modelos

El uso de estándares es de suma importancia debido a que permite capturar y comunicar las mejores prácticas de diseño, posibilita y alienta el reuso, facilita la interoperación entre diferentes herramientas, y fomenta la especialización permitiendo la construcción de herramientas más potentes y sofisticadas (Selic, 2003). Por lo cual, el método de desarrollo conducido por modelos de procesos colaborativos que se propone en esta tesis está basado en los principios y componentes de la iniciativa Arquitectura Conducida por Modelos (Model-Driven Architecture, MDA) del Object Management Group (OMG) (OMG, 2003a). Esta iniciativa propone un marco conceptual junto con un conjunto de estándares (UML, MOF, XMI, QVT, etc.) para construir métodos de desarrollo conducidos por modelos (Mellor y otros, 2004).

Un estándar clave en MDA es el Lenguaje de Modelado Unificado (UML) (OMG, 2003b), el cual da soporte al modelado visual de sistemas. UML ha sido extensamente aceptado y utilizado en la industria y en la academia, al punto que la mayoría de las técnicas y métodos de modelado de software y sistemas utilizan UML. Este lenguaje, junto con su mecanismo de extensión basado en perfiles, se han convertido en un habilitador clave del desarrollo conducido por modelos. Con UML y sus perfiles, los modelos pueden ser utilizados no sólo en forma horizontal (para describir diferentes aspectos de un sistema) sino también en forma vertical (para permitir el refinamiento de las abstracciones de mayor nivel a los niveles más bajos de detalle de un sistema) (Sendall y Kozaczynski, 2003). Otro estándar importante que también puede ser utilizado para definir lenguajes de modelado es Meta-Object Facility (MOF) (OMGc, 2003).

Los principios que propone MDA son (Mellor y otros, 2004):

- Los modelos expresados en una notación bien definida utilizando un estándar como UML son la base fundamental para el entendimiento y construcción de los sistemas.
- La construcción de sistemas puede ser organizada alrededor de un conjunto de modelos en diferentes niveles de abstracción, los cuales son vinculados por medio de transformaciones entre los modelos, a través de una arquitectura de capas y transformaciones.
- Una base formal para describir modelos a través de metamodelos facilita la integración y transformación entre modelos.

Los componentes de MDA (Figura 3-3) son: Modelos Independientes de la Plataforma (MIPs), Modelos Específicos de la Plataforma (MEPs), Transformaciones de MIPs a MEPs y Transformaciones de MEPs a código fuente. Una plataforma hace referencia a la tecnología que da soporte al sistema que está siendo construido.

De esta manera MDA propone un enfoque para:

- Especificar un sistema con independencia de la plataforma que lo soporta, esto es, definir un MIP.
- Seleccionar una o varias plataformas para el sistema.

- Transformar un MIP hacia uno o más MEPs en las plataformas seleccionadas.
- Generar el código, esto es, la definición de la transformación de un MEP a código, para la generación automática del código.

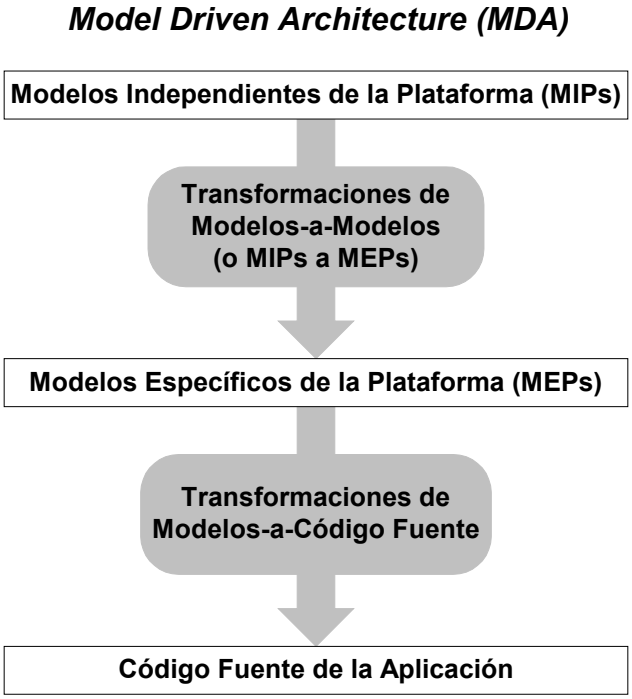


Figura 3-3. Componentes de MDA

Este enfoque propuesto por MDA promete varios beneficios (Czarnecki y Helse, 2003): *portabilidad e interoperabilidad* debido a la separación de los aspectos del sistema con respecto a su transformación en una tecnología de implementación específica; incremento de la *productividad* debido a la automatización de la transformación; mejoramiento de la *calidad* debido al reuso de patrones bien probados y de las mejores prácticas en el procedimiento de transformación; y mejoramiento del *mantenimiento* del sistema debido a una mejor separación y una mejor consistencia entre modelos y código.

3.3.2. Componentes y Técnicas del Método de Desarrollo de Procesos Colaborativos

En MDA, el concepto de sistema no sólo se refiere al software sino que también puede referirse a una empresa, a un conjunto de empresas, o combinación de partes de diferentes sistemas, etc. (OMG, 2003a). En el contexto de esta tesis y por ende del método propuesto, el sistema a ser desarrollado comprende: las especificaciones de los

procesos colaborativos en un lenguaje ejecutable y las interfaces de los SAPC de los socios, las cuales forman parte del sistema de información B2B.

Aunque MDA define los componentes que deberían formar parte de un método de desarrollo conducido por modelos, no provee las técnicas concretas que son requeridas para definir cada componente, las cuales dependerán del tipo de sistema a desarrollar. Por lo tanto, siguiendo el enfoque de MDA, las técnicas propuestas para la definición de los componentes del método de desarrollo conducido por modelos para procesos colaborativos son (Figura 3-4):

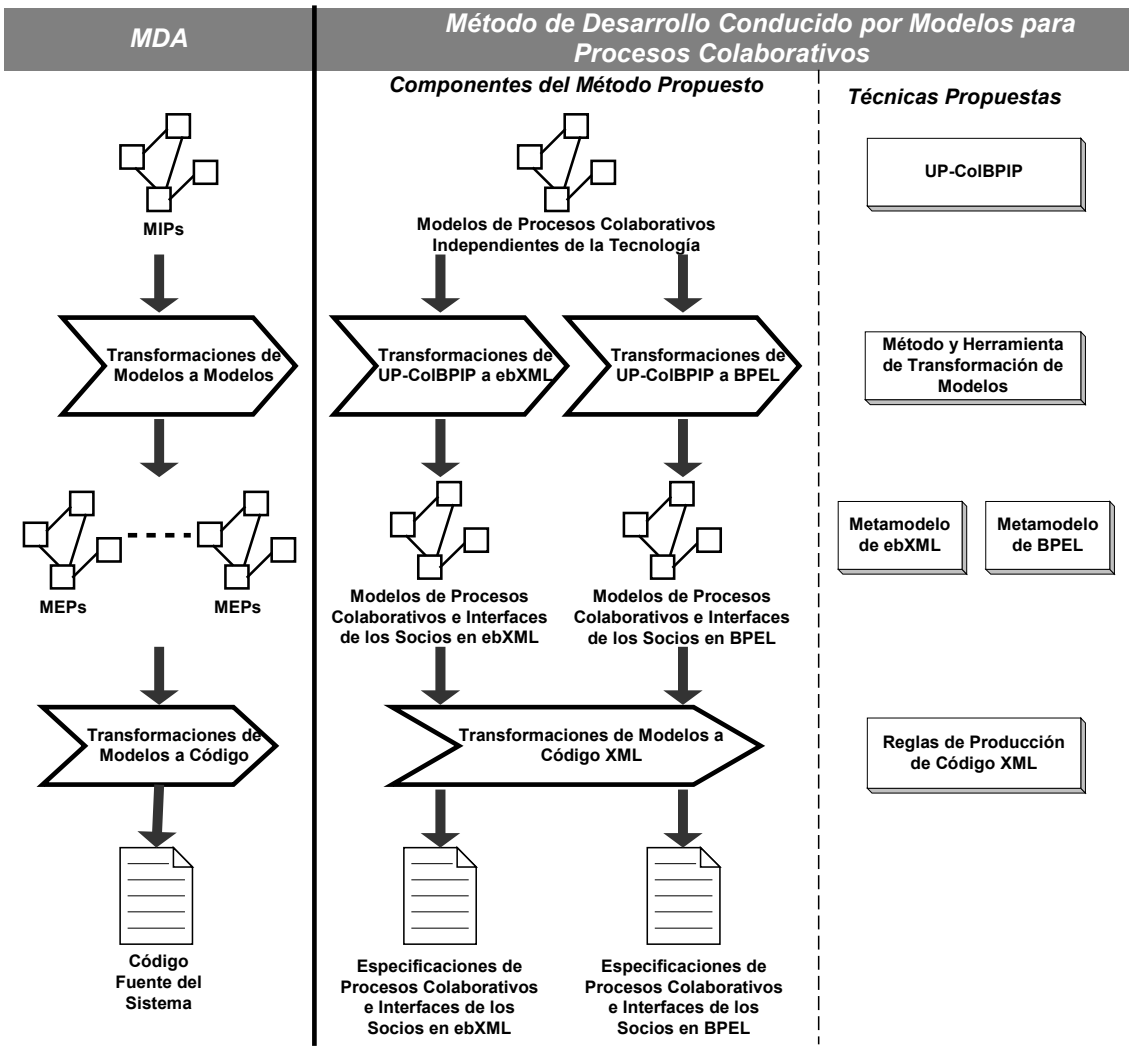


Figura 3-4. Componentes y Técnicas del Método de Desarrollo Conducido por Modelos para Procesos Colaborativos.

- El **Perfil UML para Procesos de Negocio Colaborativos basados en Protocolos de Interacción**, denominado **UP-ColBPIP** (Villarreal y otros, 2004b). Éste es el lenguaje de modelado propuesto en el marco de esta tesis para

la construcción de modelos de procesos colaborativos independientes de la tecnología. UP-ColBPIP permite definir MIPs en el contexto de colaboraciones B2B brindando soporte a las etapas de análisis y diseño de los procesos colaborativos. Los principios de diseño de este lenguaje, sus objetivos y la semántica de los conceptos provistos por el mismo son detallados en el capítulo 4.

- Uno o más **metamodelos** utilizados para construir modelos de procesos colaborativos basados en un estándar B2B. Generalmente, estos estándares no proveen un lenguaje de modelado visual, sino que proveen lenguajes basados en XML. Por lo tanto, para construir modelos basados en estos estándares, una alternativa es usar metamodelos, sin proveer una notación específica para describir los modelos. Los metamodelos pueden ser generados a partir de los esquemas XML de los lenguajes provistos por el estándar B2B, los cuales definen la estructura de los documentos XML que representan las especificaciones B2B. De esta manera, un modelo se corresponde con un documento XML. También es posible utilizar lenguajes de modelado específicos, los cuales proveen una sintaxis concreta (notación gráfica) para definir los modelos basados en un estándar. En el caso que los MEPs sean utilizados por los diseñadores, para agregar detalles o definir aspectos que no pueden ser derivados a partir de un MIP, el uso de un lenguaje de modelado específico para cada estándar B2B es más adecuado, dado que el mismo seguramente proveerá una notación más intuitiva para visualizar los modelos. Por el contrario, si los MEPs serán utilizados sólo como una representación intermedia, debido a que la mayoría de los conceptos de un MEP pueden ser derivados desde un MIP, el uso de metamodelos es más adecuado. Esto es discutido en el capítulo 5.

- El **Método de Transformación de Modelos**. Este es el método propuesto en esta tesis para dar soporte a la definición y ejecución de transformaciones de modelos de procesos colaborativos definidos con UP-ColBPIP a modelos basados en un estándar B2B. El método es soportado por una herramienta de transformación de modelos. En el capítulo 5 se proveen los detalles del método y de la herramienta prototipo que lo soporta. En el capítulo 6 se describe la

transformación de modelos definidos con UP-ColBPIP a ebXML. Con esto se da soporte a la transformación de modelos independientes de la tecnología a una de las soluciones tecnológicas posibles: un estándar basado en transacciones de negocio. En el capítulo 7 se describe la transformación de modelos definidos con UP-ColBPIP a BPEL. Con esto se da soporte a la transformación de modelos independientes de la tecnología a otra de las soluciones tecnológicas posibles: un estándar basado en composición de servicios web.

- **Reglas de Producción para la Generación de código XML**. Estas reglas permiten generar, a partir de modelos basados en un estándar B2B, la salida final de las transformaciones, esto es, uno o más documentos XML especificando las interfaces de los SAPC de los socios y los procesos colaborativos, de acuerdo a los lenguajes provistos por el estándar B2B. La herramienta de transformación de modelos desarrollada también da soporte a este último paso. Estas reglas son descritas en el capítulo 5.

3.4. Trabajos Relacionados

Un método para transformaciones conducidas por modelos, específicamente apuntado al desarrollo de procesos de negocio para e-Business es el propuesto por Koehler y otros (2003). El método posibilita la transformación de modelos de la vista de negocio en modelos de la tecnología de información. Para el modelado de procesos en la vista de negocio, los autores analizan dos enfoques de modelado: ADF de Holosofx/IBM y diagramas de actividades de UML2. Luego, a través de una máquina de transformación estos modelos son transformados a una representación intermedia denominada grafo de procesos, la cual representa sólo el flujo de información estructurada del proceso, descartando los restantes detalles que puedan haber sido definidos con los enfoques de modelado anteriores. A partir del grafo de proceso se crean otras representaciones intermedias: subgrafos por cada tipo de objeto identificado en el proceso, los cuales muestran el workflow interno para este tipo de objeto; y subgrafos que representan las dependencias del flujo de información entre los pasos del proceso y los socios involucrados. Estos subgrafos o flujos de información estructurada son utilizados por la máquina de transformaciones para automáticamente generar: especificaciones de protocolos de negocio en el lenguaje de composición de servicios

web BPEL, y documentos "adaptativos" cuyo comportamiento es controlado por una máquina de estado finita.

Aunque este método da soporte a las diferentes etapas del proceso de desarrollo conducido por modelos para procesos de negocio, el mismo no cumple con algunos de los requerimientos para el desarrollo de colaboraciones B2B. En primer lugar, el método apunta a arquitecturas de tecnología de información basadas en servicios web, con lo cual sólo consideran como posibles implementaciones de procesos colaborativos estándares basados en composición de servicios web. En segundo lugar, los modelos de procesos en la vista de negocio son definidos teniendo en cuenta la vista de un único rol, lo cual no es adecuado para representar la vista global y las interacciones peer-to-peer requeridas en un modelo de proceso colaborativo. Esto se debe a que el método de transformación es utilizado para generar los denominados protocolos de negocio o protocolos de conversación. Este enfoque puede ser utilizado para definir procesos colaborativos pero sólo desde el punto de vista de un socio. En tercer lugar, no se provee un lenguaje para definir transformaciones entre modelos, sino que las mismas ya están incorporadas dentro de la herramienta. Por lo tanto, una transformación no puede ser definida o modificada por los desarrolladores y tanto el uso de lenguajes de modelado para MIPs como así también para MEPs está limitado a lo que soporte la herramienta.

Otro método basado en el desarrollo conducido por modelos para soportar procesos colaborativos es el propuesto por Baïna y otros (2004). Este método se enfoca en el modelado y la generación automática de protocolos de conversación definidos en los servicios web compuestos. De esta manera, al igual que el trabajo mencionado anteriormente, éste se enfoca en definir y especificar el comportamiento del proceso colaborativo sólo desde el punto de vista de un único socio. El método propone definir modelos de protocolos de conversación independientes de la tecnología como máquinas de estado extendidas, a través de un lenguaje soportado por la herramienta Selv-Serf (Benatallah y otros, 2003). A partir de estas máquinas de estado se generan esqueletos del protocolo de conversación en el lenguaje BPEL. La transformación es realizada a través de una serie de pasos lógicos que comienza con la transformación de las transiciones, luego de los estados y finalmente la vinculación de ambas. Por lo tanto, este método tampoco permite a los desarrolladores definir la transformación de un MIP a un MEP, sino que la lógica detrás de la misma ya está definida e implementada en la

herramienta. Además, este método tampoco considera el uso de MEPs, debido a que el código XML es generado directamente a partir de las máquinas de estado que describen el protocolo de conversación.

En el trabajo propuesto por Gardner (2003) se provee un perfil UML para el lenguaje BPEL, el cual permite definir procesos colaborativos basados en dicho estándar sin necesidad de escribir código XML. Asociado a este perfil se proveen las reglas para generar el código XML a partir de los modelos UML definidos con el perfil. La principal desventaja de este trabajo es que no considera la definición de procesos de negocio independientes de la tecnología. Sólo provee un lenguaje para soportar la definición de MEPs, en este caso, procesos colaborativos basados en BPEL. Además, al igual que los trabajos anteriores, debido a que los procesos son definidos como protocolos de conversación, el comportamiento de los mismos es especificado desde el punto de vista de un único socio.

Existen también otras propuestas para el desarrollo conducido por modelos de servicios web (Grønmo y Solheim, 2004) (Thöne y otros, 2002). No obstante, las mismas se enfocan en la implementación de los procesos privados basados en la composición de servicios web y no en los procesos colaborativos.

Uno de los trabajos que destaca la importancia del uso de un método basado en el desarrollo conducido por modelos para definir procesos colaborativos es el propuesto por Kramler (2004). Aquí se destaca la importancia de considerar dos tipos de soluciones para la implementación de procesos colaborativos: ebXML y Composición de Servicios Web. No obstante, como parte del método de desarrollo conducido por modelos no se provee ni tampoco se utiliza un lenguaje para definir procesos colaborativos independientes de la tecnología. En su lugar, debido a que los lenguajes que los estándares B2B proveen están basados en XML, se propone un perfil UML para modelar esquemas XML junto con un conjunto de reglas para transformar modelos definidos en dicho perfil a Esquemas XML.

Finalmente, la generación de especificaciones en ebXML a partir de modelos conceptuales de procesos colaborativos también ha sido descripta por Hofreiter y Huemer (2004a). Aquí, la metodología utilizada para el análisis y diseño de procesos colaborativos es UN/CEFACT Modeling Metodology (UMM), la cual provee un perfil UML. Aunque UMM alega independencia de la tecnología, los principales elementos

conceptuales son aquellos también provistos en ebXML BPSS, debido a que el metamodelo de BPSS es un subconjunto del metamodelo de UMM. Por lo tanto, UMM influencia la adopción de un estándar específico. No obstante, la transformación de modelos de procesos colaborativos a especificaciones en BPSS puede ser realizada en forma directa.

Hofreiter y Huemer (2004b) también proponen la transformación de modelos de procesos colaborativos definidos con UMM a BPEL, con el objetivo de demostrar la independencia de la tecnología de UMM. No obstante, las transformaciones son descriptas en forma conceptual y no se considera la utilización de un método basado en el desarrollo conducido por modelos para soportar las etapas del desarrollo de procesos colaborativos.

De esta manera se puede destacar que aunque los beneficios del desarrollo conducido por modelos para el dominio de procesos colaborativos y colaboraciones B2B son claros, actualmente el mismo no ha sido completamente explotado en este dominio. Ninguno de los anteriores trabajos está basado en un método de desarrollo conducido por modelos bien definido, que provea las técnicas y métodos necesarios para definir cada uno de los componentes que se requieren en este tipo de métodos. Las principales razones de esto son:

- Existe una carencia de lenguajes de modelado independientes de la tecnología que soporten el requerimiento de modelar procesos colaborativos que representen la vista global de las interacciones peer-to-peer, mostrando las responsabilidades de los roles que desempeñan los socios en los procesos.
- Como consecuencia de lo anterior, la mayoría de los trabajos anteriores sólo consideran como tecnología de implementación el uso de los actuales estándares basados en composición de servicios web.
- Un método de transformación de modelos para pasar de MIPs a MEPs no es considerado o bien no es provisto. La transformación es codificada directamente en la herramienta que soporta las transformaciones de modelos. Esto trae aparejado que los diseñadores de los procesos colaborativos y de los sistemas de información B2B no puedan definir o modificar las transformaciones. Además, los lenguajes de modelado posibles a utilizar, como así también la tecnología en la cual pueden ser definidos los procesos depende del soporte que provee la

herramienta a cada uno de los lenguajes y estándares B2B. Esto limita la utilización de diferentes lenguajes de modelado y diferentes tecnologías de implementación.

- Ninguno de los métodos mencionados anteriormente están basados en un marco conceptual para el desarrollo conducido por modelos como el propuesto por MDA, y en consecuencia, algunos no consideran y no emplean el uso de los estándares asociados a MDA.
- Algunos de los métodos anteriores se enfocan en las transformaciones de modelos a código, es decir, que sólo proveen un lenguaje para definir un tipo específico de MEP y generar a partir de estos modelos el código XML correspondiente al estándar B2B que se está considerando. Esto trae como consecuencia que los procesos colaborativos estén influenciados por la tecnología de implementación.

3.5. Conclusiones

En este capítulo se ha destacado la importancia de considerar dos niveles de abstracción en el desarrollo de colaboraciones B2B: el nivel de negocio y el nivel tecnológico. En primer lugar, los analistas de negocio y los diseñadores de sistemas se enfocan en el nivel de negocio (el dominio del problema), a través del diseño de los procesos de negocio colaborativos y de las interfaces de negocio de los socios, sin considerar la tecnología de implementación. Luego, en el nivel tecnológico (el dominio de la solución), los arquitectos y desarrolladores de sistemas se enfocan en la selección y utilización de un estándar para la especificación de los protocolos B2B, los cuales implican la especificación de: los procesos colaborativos en un lenguaje ejecutable de proceso de negocio, y las interfaces de los SAPC de los socios que conforman el sistema de información B2B.

El diseño de procesos colaborativos en el nivel de negocio es una tarea clave en el desarrollo de colaboraciones B2B, principalmente debido a que la forma en que los socios colaborarán es formalizada en dichos procesos, los cuales son compartidos y acordados en forma conjunta entre las partes. Para ello, el uso de modelos conceptuales de procesos colaborativos es un importante requisito. Más importante aún es que dichos modelos expresen todas las características de las colaboraciones B2B a las que se está

apuntando. A partir de estas características se han identificado los requerimientos para el modelado conceptual de procesos colaborativos: vista global de las interacciones entre los socios, fases de las interacciones B2B, autonomía de las empresas, descentralización e interacciones peer-to-peer, procesos colaborativos como procesos abstractos, derivación de las interfaces de los socios a partir de los procesos, perspectivas de los modelos de procesos y soporte a procesos con negociaciones.

En el nivel tecnológico, dos tipos de tecnologías para la especificación de protocolos B2B pueden ser considerados: estándares basados en Composición de Servicios Web y estándares basados en Transacciones de Negocio. Estos estándares no son compatibles entre sí y las empresas deben decidir la tecnología que consideran más adecuada para implementar los procesos colaborativos.

La generación automática de las especificaciones de los protocolos B2B basados en un estándar particular, a partir de los modelos de procesos colaborativos independientes de la tecnología, es de suma importancia para disminuir la complejidad y los costos en el establecimiento de colaboraciones B2B entre las empresas. A través de la generación automática de dichas especificaciones es posible: garantizar una correspondencia entre lo definido en un alto nivel de abstracción y su implementación con una tecnología específica; y dar soporte al problema de interoperabilidad en tiempo de diseño como consecuencia de que una empresa puede implementar procesos colaborativos con diferentes socios usando diferentes estándares.

El desarrollo conducido por modelos ha sido identificado como un habilitador clave para permitir el diseño de procesos colaborativos en un alto nivel de abstracción, y generar en forma automática especificaciones de dichos procesos y de las interfaces de los socios en un estándar B2B. Por lo tanto, se ha propuesto un método de desarrollo de procesos de negocio colaborativos, el cual explota los beneficios del desarrollo conducido por modelos en el dominio de las colaboraciones B2B. Dicho método está basado en los principios, componentes y estándares propuestos por la iniciativa MDA para la definición de métodos de desarrollo conducidos por modelos. Cada uno de los componentes de dicho método, junto con las técnicas provistas para la definición de dichos componentes han sido presentados.

El método propuesto provee una separación de los conceptos del nivel de negocio, de los del nivel tecnológico, lo cual permite pasar en forma automática desde el dominio

del problema al dominio de la solución. Esto permite incrementar el nivel de reutilización de los modelos de procesos colaborativos, y también posibilita la implementación de procesos colaborativos usando diferentes tipos de tecnologías, permitiendo a los socios poder elegir entre una mayor variedad de tecnologías de acuerdo a sus requerimientos e intereses.

En definitiva, el método propuesto tiene como principal objetivo disminuir la complejidad y los costos en el desarrollo de colaboraciones B2B. Los principales beneficios son:

- Incremento del nivel de abstracción en el desarrollo de colaboraciones B2B, debido a que los principales productos del desarrollo son los modelos de procesos colaborativos independientes de la tecnología.
- Disminución del tiempo de desarrollo, ya que la generación de la solución es realizada en forma automática a partir de los modelos conceptuales de procesos colaborativos.
- Disminución de los costos como consecuencia de la automatización en la generación de especificaciones de los protocolos B2B.
- Garantía de correspondencia entre los modelos conceptuales de procesos y las especificaciones de los mismos en un estándar específico.
- Independencia de los modelos de procesos colaborativos de las nuevas o diferentes tecnologías, incrementando el reuso de los mismos para generar diferentes soluciones tecnológicas.

Capítulo

4

UP-ColBPIP: Lenguaje de Modelado de Procesos de Negocio Colaborativos basados en Protocolos de Interacción

En este capítulo se presenta el lenguaje UP-ColBIPIP propuesto para el modelado de procesos colaborativos independientes de la tecnología. En primer lugar, se exponen los objetivos del lenguaje y el enfoque de modelado propuesto. Posteriormente, se describe el metamodelo del lenguaje y se provee la semántica de sus elementos conceptuales. Luego, se presenta la definición de UP-ColBPIP como un perfil UML que implementa dicho metamodelo. Se describe la aplicación del lenguaje a un caso de estudio. Finalmente, se discuten trabajos relacionados y se exponen las conclusiones.

4.1. Objetivos y Principios de Diseño de UP-ColBPIP

El enfoque tradicional utilizado para el modelado y gestión de procesos de negocio ha sido el uso de "workflows". El término "workflow" refiere a la automatización de un proceso de negocio, en todo o en parte, en el cual documentos, información y tareas (actividades) son pasadas desde un participante a otro de acuerdo a un conjunto de reglas de procedimiento (WfMC, 1995). Un *sistema de gestión de workflows* (WfMS) es el software que automatiza la coordinación y el control de las tareas durante la ejecución del proceso de negocio. Los workflows han sido utilizados con éxito por las empresas para modelar y gestionar sus procesos de negocio internos o privados. Para el modelado de workflows generalmente se utilizan lenguajes de modelado de procesos orientados a las actividades, como por ejemplo redes de petri (van der Aalst, 2000), diagramas de actividades de UML (Dumas y Hofstede, 2000), IDEF03 (IDEF, 1995).

No obstante, como se ha señalado en numerosos trabajos, el uso de workflows para la gestión de procesos colaborativos no es apropiado (Chen y Hsu, 2001) (Bussler, 2001). En primer lugar, porque los sistemas de gestión de workflow están basados en un modelo de gestión centralizada de los procesos de negocio, el cual no es apropiado para

procesos colaborativos, ya que atenta contra la autonomía de las empresas. En segundo lugar, los enfoques de modelado de workflows no dan soporte a los principales requerimientos del modelado conceptual de procesos colaborativos (Sección 3.2.1). Principalmente, estos enfoques no permiten describir las interacciones peer-to-peer entre las empresas ocultando las decisiones internas que afectan el flujo de control de los procesos. En su lugar, los procesos basados en workflows deben ser definidos teniendo en cuenta todos los detalles del flujo de control (Chen y otros, 2003) (Jung y otros, 2004). Más aún, en un workflow no sólo se definen las actividades involucradas en cada paso del proceso, sino también las aplicaciones que serán invocadas en el caso que estas actividades sean automatizadas. Esto es diferente en colaboraciones B2B, debido a que las aplicaciones o recursos utilizados para realizar cierta acción no deben ser definidos en los procesos colaborativos, ya que esto forma parte de los detalles internos de la empresa. De esta manera, los enfoques tradicionales basados en workflows no consideran la autonomía de las partes involucradas en el proceso.

Si bien existen propuestas que adaptan el uso de workflows para modelar procesos entre empresas (Lazcano y otros, 2000) (Grefen y otros, 2000), las mismas no descentralizan la gestión de dichos procesos. Por lo tanto, son apropiadas para procesos de negocio entre empresas en relaciones B2B basadas en e-marketplaces, en donde la gestión es centralizada; pero no son adecuadas para colaboraciones B2B basadas en modelos descentralizados y sistemas de información P2P (Villarreal y otros, 2003b).

Por otra parte, las interacciones B2B no pueden restringirse a la mera transferencia de información (Goldkuhl y Lind, 2004), sino que éstas deben apuntar principalmente a la realización de acciones por parte de cada empresa y la comunicación de esas acciones hacia sus otros socios. De este modo, los procesos colaborativos no sólo deberían expresar la información a ser intercambiada, sino también la comunicación de las acciones entre los socios. Estos aspectos comunicativos no son contemplados en los lenguajes de modelado de procesos de negocio orientados a las actividades, utilizados para modelar workflows (Goldkuhl y Lind, 2004) (Weigand y otros, 1998). Estos lenguajes se enfocan básicamente en la definición de las actividades que deben ser realizadas por las partes y en cómo la información transferida entre dichas actividades es transformada. Las bases teóricas de estos enfoques son menos rígidas, ya que dependen del entendimiento intuitivo de sus nociones axiomáticas, como el concepto de actividad o tarea (Dietz, 2002). La consecuencia de esto es la ambigüedad en la

definición de las actividades, ya que dos analistas pueden tener diferentes interpretaciones del significado de las mismas. Además, estos lenguajes se enfocan en capturar el "qué" de un proceso, es decir, qué actividades se realizan, en lugar de enfocarse en el "por qué" tales acciones son realizadas. El "por qué" es necesario expresarlo en los procesos colaborativos debido a que una acción realizada por una parte conlleva una intención, con el objetivo de asumir un compromiso, romper un compromiso previamente asumido, solicitar la creación de un compromiso, o simplemente informar un hecho.

Estos aspectos de comunicación requeridos en los procesos colaborativos pueden ser capturados a través de un enfoque de modelado basado en la teoría Language/Action Perspective (LAP). Esta teoría fue introducida en el campo de los sistemas de información en los ochenta (Flores y Ludlow, 1980) y se ha convertido en un paradigma prometedor para el diseño de procesos de negocio y de los sistemas de información que los soportan (Dietz y otros, 1998). La principal premisa de LAP es que la comunicación no es sólo la transferencia de información sino que, además, la comunicación es un tipo de acción que crea compromisos entre las partes. Esto constituye la principal diferencia con respecto a las técnicas de modelado de procesos orientados a las actividades.

LAP enfatiza qué es lo que las partes realizan, cómo el lenguaje es utilizado para crear una realidad común para todas las partes, y cómo sus actividades son coordinadas a través del lenguaje (Kethers y Schoop, 2000). Es decir, que el lenguaje es utilizado para alcanzar un entendimiento mutuo entre las partes, como ser compromisos, acuerdos y metas. Las bases teóricas de LAP están sustentadas en la Teoría de Actos de Comunicación (conocida como Speech Acts Theory) (Austin, 1962) (Searle, 1975). Esta teoría tiene sus raíces en la lingüística y filosofía del lenguaje natural.

Un *acto de comunicación (speech act)* es la unidad básica de comunicación en LAP (Kethers y Schoop, 2000) y es definido como una acción realizada a través del lenguaje, la cual cambia el universo de discurso y es ejecutada por un interlocutor y comprendida por un receptor (Austin, 1962). La principal idea de esta teoría es que un acto de comunicación expresa una actitud lógica de un interlocutor con respecto a alguna proposición. Esto significa que puede ser analizado formalmente a través de la estructura *F(P)*, donde *F* es la fuerza o intención del acto aplicada a una proposición *P*, la cual es el contenido del mismo. Dicha intención representa la "actitud" del

interlocutor, por ejemplo: solicitar, proponer, aceptar, etc. y el contenido es el hecho transferido desde el interlocutor al receptor, por ejemplo un "pronóstico de demanda". De esta manera, diferentes *F* pueden ser aplicadas al mismo *P*, representando diferentes intenciones del interlocutor con respecto a un mismo hecho o proposición. Por ejemplo, "solicitar pronóstico de demanda" y "proponer pronóstico de demanda" tienen el mismo contenido pero claramente expresan diferentes intenciones del interlocutor.

De esta manera, un enfoque de modelado basado en LAP permite capturar las características mencionadas anteriormente acerca de las colaboraciones B2B, ya que no sólo se centra en el intercambio de información sino también en la comunicación y creación de compromisos entre las partes. Además, a través del uso de actos de comunicación, el lenguaje es utilizado para crear un entendimiento común entre las partes, ya que se asume que la semántica de la intención del acto de comunicación es conocida y entendida por las partes que están interactuando. Esto es de suma importancia en colaboraciones B2B en donde los socios deben cooperar y colaborar para establecer compromisos y alcanzar metas comunes. Si tal entendimiento no existiera, la colaboración sería difícil de alcanzar.

Existen diferentes métodos o enfoques orientados a la comunicación, los cuales explotan los beneficios de LAP para el modelado de procesos de negocio: Action Workflow (Medina-Mora y otros, 1992), Business Action Theory (Goldkuhl, 1996), DEMO (Dietz, 1999) y las propuestas de patrones en capa (Lind y Goldkuhl, 2001) (Weigand y Heuvel, 1998). Estos enfoques han sido propuestos para el diseño de procesos de negocio privados, aunque posteriormente algunos de ellos también han sido aplicados a escenarios de e-business. Una característica común de los mismos es que enfatizan el uso de patrones de interacción, los cuales agrupan un conjunto de actos de comunicación. Estos patrones son genéricos y ya están predefinidos. En consecuencia, los procesos de negocio son definidos a través de la composición de diferentes patrones de interacción predefinidos. En algunos enfoques, los patrones de interacción representan un concepto específico, como por ejemplo, el concepto de transacción de negocio (Dietz, 1999) (Lind y Goldkuhl, 2001). De esta manera, los diseñadores no deben enfocarse en determinar cuál es el acto de comunicación más apropiado para describir la interacción, sino que deben enfocarse en modelar los procesos de negocio utilizando conceptos tales como patrones genéricos y transacciones de negocio. Por lo tanto, aunque un acto de comunicación es el concepto atómico, no es el principal

concepto utilizado para modelar procesos de negocio. El mismo es utilizado en un bajo nivel de abstracción. Esto conduce a una pérdida de poder descriptivo en los modelos de procesos de negocio.

Por otra parte, los patrones de interacción predefinidos son menos flexibles para el diseño, debido a que existen situaciones que pueden no estar contempladas en dichos patrones. Esto ha sido identificado como una de las principales desventajas de estos enfoques (Goldkuhl y Lind, 2004).

Estas dificultades pueden ser subsanadas si se considera a un acto de comunicación, no sólo como el concepto atómico sino también como el concepto principal y de más alto nivel para definir las interacciones entre las partes dentro de un proceso colaborativo. De esta manera es posible aprovechar los beneficios de LAP para permitir que los diseñadores se enfoquen en las interacciones y la creación de compromisos entre las partes, usando los actos de comunicación como el principal bloque de construcción de los procesos colaborativos.

En función de lo discutido anteriormente, para modelar procesos colaborativos independientes de la tecnología se considera conveniente disponer de un enfoque de modelado que cumpla con los siguientes objetivos:

- Soportar los requerimientos identificados previamente (Sección 3.2.1) con respecto al modelado conceptual de procesos de negocio colaborativos;
- Tomar como base la teoría LAP y la teoría de actos de comunicación, con el propósito de proveer un enfoque orientado a la comunicación que permita capturar no sólo el intercambio de información sino también los aspectos de comunicación y creación de compromisos entre las partes;
- Resolver los inconvenientes de los enfoques de modelado de procesos de negocio orientados a la comunicación, elevando el nivel de abstracción de los actos de comunicación, utilizándolos como los bloques de construcción principales de un proceso.

Para alcanzar estos objetivos, en esta tesis se propone el uso de *Protocolos de Interacción*, los cuales están basados en las teorías de LAP y de actos de comunicación, para modelar procesos de negocio colaborativos. El concepto de protocolo de interacción ha sido utilizado en el área de los sistemas multi-agentes para analizar y

diseñar comunicaciones entre agentes de software (Bauer y otros, 2001). Las propiedades típicas de los agentes de software son: autonomía, descentralización, heterogeneidad, independencia y habilidad para mantener interacciones sociales (Jennings, 2001). Debido a que estas propiedades son también deseables para las entidades (empresas) involucradas en colaboraciones B2B, el uso de protocolos de interacción para representar interacciones B2B se considera un enfoque apropiado a ser explotado en este dominio (Villarreal y otros, 2003b) (Villarreal y otros, 2004b).

4.1.1. Modelado de Procesos Colaborativos basados en Protocolos de Interacción

En el contexto de relaciones de colaboración B2B, se propone el concepto de *protocolo de interacción* para describir un patrón de comunicación de alto nivel a través de una secuencia admisible de mensajes entre empresas (Villarreal y otros, 2003b). Un protocolo de interacción representa un proceso colaborativo que las empresas acuerdan llevar a cabo. Puede también ser visto como la política de conversación que deben seguir las empresas en las interacciones con sus socios, como parte de la ejecución de un proceso colaborativo. Esta política considera las condiciones a ser tenidas en cuenta en el flujo de control de los mensajes, pero no incluye los detalles de las decisiones internas de cada empresa, en función de las cuales se asigna valor a dichas condiciones. Tampoco incluye la definición explícita de las actividades que cada parte debe realizar para posibilitar las interacciones entre las mismas.

Un protocolo de interacción puede ser definido como una tupla P = <R,ch,M,A, OD>, en donde *R* es el conjunto de roles involucrados en el proceso, *ch* es la coreografía de los mensajes intercambiados entre los roles, *M* es el conjunto de mensajes considerados en *ch*, *A* es el conjunto de actos de comunicación que pueden ser utilizados para describir los mensajes, y *OD* es la ontología utilizada para describir el contenido transportado en los mensajes.

Un *mensaje* describe una interacción entre dos roles indicando una dirección, esto es, el rol que envía el mensaje y el rol que responde al mensaje. Un Mensaje puede ser definido como una tupla m = <e,r,a,d,c,t> en la cual:

- $e \in R$, *e* es el rol que envía el mensaje (rol remitente).
- $s \in R$, *s* es el rol que recibe el mensaje (rol receptor).

- *a* $\in$ *A*, *a* es el acto de comunicación que representa la intención del remitente en la interacción.
- *d* $\in$ *OD*, *d* es el documento de negocio transportado por el mensaje, es decir, la información intercambiada en la interacción.
- *c* es una condición sobre el mensaje que representa la expresión condicional utilizada para indicar cuando un mensaje puede enviarse. (Esta es opcional).
- *t* representa una restricción de tiempo que establece una fecha límite dentro de la cual el mensaje debe ser ejecutado. (Esta es opcional).

Un rol *r* $\in$ *R* representa la responsabilidad, en términos de una secuencia de mensajes de envío y de recepción, que una empresa tiene en un proceso colaborativo. De esta manera, un rol *r* puede ser definido como r = <ME_r, MR_r>, en donde ME_r es el conjunto de mensajes que el rol *r* puede enviar y MR_r es el conjunto de mensajes que el rol *r* puede recibir.

La *coreografía de mensajes* define la secuencia compleja de mensajes, para representar mensajes paralelos, alternativos, etc., generalmente usando *operadores de flujo de control* tales como *And*, *Or*, y *Xor*. También define los estados finales de un protocolo que expresan la terminación exitosa o sin éxito del mismo. La especificación concreta de la coreografía estará dada por el formalismo utilizado para modelar los protocolos, tales como diagramas de secuencia de UML, máquinas de estado o redes de petri. Como se describe posteriormente, el lenguaje propuesto en el marco de esta tesis extiende los diagramas de secuencia de UML2 para permitir el correcto modelado de protocolos de interacción.

El principal propósito de modelar procesos colaborativos a través de protocolos de interacción (Villarreal y otros, 2003b) (Villarreal y otros, 2003d) es satisfacer los requerimientos definidos para soportar colaboraciones B2B. Uno de estos requerimientos es la preservación de la autonomía de las empresas (Sección 3.2.1.3). A diferencia de los enfoques tradicionales de modelado de procesos orientados a las actividades, un modelo de proceso colaborativo basado en un protocolo de interacción se enfoca principalmente en el intercambio de mensajes entre los roles desempeñados por los socios. De esta manera, las actividades que cada socio debe realizar, para procesar la información recibida o producir la información a ser enviada, no son

definidas en un protocolo de interacción, sino que son mantenidas ocultas a los restantes socios. Las acciones que las empresas realizan son indicadas a través de la intención del mensaje según la semántica del acto de comunicación asociado, de acuerdo a los principios de la teoría LAP. Esto permite desacoplar los procesos colaborativos de los procesos privados que lo soportan.

Un protocolo de interacción permite expresar la vista global de las interacciones entre los socios (Sección 3.2.1.2). La coreografía de mensajes describe las interacciones peer-to-peer y las responsabilidades de los socios a través de los roles que cada uno juega. Además, el carácter descentralizado de los procesos colaborativos también es resaltado a través de los protocolos de interacción ya que cada rol es definido en términos de la secuencia de mensajes que debe recibir y enviar. Esto significa que cada empresa debe gestionar en forma independiente el rol que tenga asignado en el proceso. La coordinación y sincronización de los mensajes intercambiados entre los roles puede ser realizada en forma conjunta entre las empresas, debido a que a través del protocolo de interacción se conoce qué mensaje un rol debe esperar o enviar en cada etapa del protocolo.

Lo anterior está relacionado con el objetivo de definir procesos colaborativos como procesos abstractos (Sección 3.2.1.5), en el sentido que no exista una instancia única de los mismos gestionada por una de las partes. En realidad, una instancia de un protocolo consistirá de dos o más instancias independientes, una por cada rol, de modo que cada socio gestionará la instancia del rol que el mismo desempeña en el protocolo. Dichas instancias son sincronizadas mediante el intercambio de mensajes. De esta manera, la instancia de un protocolo puede ser vista como la composición de las instancias de cada uno de los roles.

Los protocolos de interacción también dan soporte a las perspectivas requeridas en un modelo de proceso colaborativo (Sección 3.2.1.6). La perspectiva organizacional es representada a través de la definición de los roles y las empresas que los desempeñan. La perspectiva de información es representada a través de la definición del contenido de cada mensaje. La perspectiva de comportamiento es representada a través de la definición de la coreografía de mensajes. La perspectiva funcional (las actividades de cada rol) está implícitamente soportada por las acciones de comunicación expresadas en los mensajes.

Los protocolos de interacción soportan además una perspectiva intencional a través de los actos de comunicación, los cuales permiten representar las intenciones de una empresa en una interacción o comunicación dentro de un proceso colaborativo. Las decisiones y compromisos de las empresas son conocidos a partir de los actos de comunicación, lo cual posibilita definir negociaciones complejas dentro de los procesos colaborativos (Sección 3.2.1.7).

Finalmente, los protocolos de interacción permiten a los diseñadores explotar las capacidades de un enfoque basado en la teoría LAP, y usar los actos de comunicación en un alto nivel de abstracción. Un mensaje basado en un acto de comunicación no sólo es un concepto atómico sino también el principal bloque de construcción en un proceso. En consecuencia, los diseñadores deben centrarse en definir los mensajes utilizando el acto de comunicación que mejor represente la intención del rol remitente, posibilitando la creación de compromisos entre las partes. Los compromisos creados son deducidos automáticamente a partir de los actos de comunicación de los mensajes. Un protocolo de interacción no está predefinido de antemano y es el diseñador quien define la secuencia de mensajes.

De esta manera, a través de los protocolos de interacción se provee un enfoque orientado a la comunicación para el modelado conceptual de procesos colaborativos.

4.2. El Lenguaje UP-ColBPIP

En esta sección se describe el lenguaje de modelado propuesto para el diseño de procesos de negocio colaborativos independientes de la tecnología. El fundamento teórico principal del lenguaje es el uso del enfoque de modelado basado en protocolos de interacción, descrito en la sección anterior. El lenguaje también da soporte a la etapa de análisis de los procesos colaborativos, en donde los mismos son identificados a partir de los requerimientos y objetivos asumidos en un acuerdo de colaboración entre las empresas. También soporta la definición de las interfaces de negocio que los socios deben implementar, las cuales son derivadas de los protocolos de interacción.

Siguiendo los lineamientos propuestos por MDA para el desarrollo conducido por modelos, un lenguaje de modelado específico del dominio puede ser definido de dos maneras diferentes (Fuentes y Vallecillo, 2004). Una alternativa consiste en definir un

nuevo lenguaje describiendo su metamodelo utilizando MOF (Meta-Object Facility) (OMG, 2003c), el cual es un lenguaje para definir lenguajes de modelado.

Otra alternativa consiste en definir un nuevo lenguaje extendiendo el metamodelo de UML (OMG, 2003b), a través de un conjunto de mecanismos de extensión basados en el uso de perfiles UML, los cuales respetan y mantienen la semántica original de UML (clases, asociaciones, atributos, etc.). La intención de los perfiles en UML es ofrecer un mecanismo para adaptar un metamodelo existente con conceptos que son específicos de un dominio particular. Los mecanismos de extensión utilizados para definir un perfil son: estereotipos, valores etiquetados y restricciones (OMG, 2003b). Un *estereotipo* representa un nuevo elemento de modelado que extiende una o varias metaclases de UML (clases del metamodelo de UML). Asociaciones entre estereotipos no son permitidas debido a que un perfil UML no puede modificar la sintaxis del metamodelo de UML. Los *valores etiquetados* definen los atributos de un estereotipo. Las *restricciones* pueden ser utilizadas para restringir el uso de los nuevos elementos de modelado o para restringir el uso de partes del metamodelo que es extendido. Las restricciones expresan las condiciones que ha de verificar un modelo "bien formado" definido con el perfil. Las restricciones pueden ser definidas en lenguaje natural o con OCL (OMG, 2003e).

La primer alternativa puede ser utilizada cuando la semántica de los constructores requeridos no coincide con la de UML. De este modo, definiendo un nuevo lenguaje desde MOF se puede lograr mayor flexibilidad y correspondencia con los conceptos del dominio específico del lenguaje. Sin embargo, debido a que un lenguaje definido con MOF no respeta el metamodelo de UML, los conceptos del mismo no podrán ser manejados por las herramientas Case UML existentes en el mercado. Es decir, que se requeriría una herramienta específica para soportar dicho lenguaje, adicionando también una notación específica. En su lugar, el uso de perfiles UML posibilita que los conceptos del mismo puedan ser soportados por cualquier herramienta Case UML. No obstante, más allá de estas diferencias, es difícil determinar cuándo es mejor utilizar una de las dos alternativas (Fuentes y Vallecillo, 2004) (OMG, 2003b).

En esta tesis, el lenguaje de modelado de procesos de negocio colaborativos propuesto se ha definido como un perfil UML (Villarreal y otros, 2004b), basado en UML 2 (OMG, 2003b). El objetivo ha sido desarrollar un lenguaje que pueda ser

utilizado por las herramientas Case UML existentes, que permita el reuso de la notación provista por UML. De este modo se obtiene un lenguaje fácil de usar por los analistas de negocio y los diseñadores de sistemas, dado que UML es un lenguaje de modelado visual extensamente conocido y utilizado tanto para el modelado de sistemas de información como para el modelado de procesos de negocio (Eriksson y Penker, 2000).

Para la elaboración del *Perfil UML de Procesos de Negocio Colaborativos basados en Protocolos de Interacción (UP-ColBPIP)*, se han seguido las guías y recomendaciones propuestas en (Fuentes y Vallecillo, 2004) para la definición de perfiles UML. Un procedimiento similar también ha sido adoptado por el OMG para definir algunos de sus perfiles UML estándares (OMG, 2004). En primer lugar se define el metamodelo que describe el dominio del problema a modelar, utilizando los mecanismos del propio UML para definir modelos conceptuales. El metamodelo define los principales elementos conceptuales del lenguaje, las relaciones entre los mismos y las restricciones sobre dichos conceptos y relaciones. Las restricciones son definidas utilizando el lenguaje OCL (Object Constraint Language) (OMG, 2003e). Posteriormente, a partir del metamodelo, se construye el perfil UML siguiendo los siguientes pasos:

- Se definen los estereotipos que corresponden a los elementos conceptuales del metamodelo, y se indican las metaclases del metamodelo de UML que los estereotipos están extendiendo.
- Se definen los valores etiquetados de los estereotipos a partir de los atributos de los elementos del metamodelo.
- Se definen las restricciones de acuerdo a las restricciones del metamodelo.

De esta manera, se establece una relación entre el metamodelo y el perfil.

4.2.1. Metamodelo del Lenguaje UP-ColBPIP

El modelado conceptual de procesos de negocio colaborativos utilizando el lenguaje UP-ColBPIP consiste de cuatro pasos principales:

1. La definición de la colaboración B2B, especificando los roles de los socios, el acuerdo de colaboración y las metas de negocio comunes.

2. La identificación de los procesos colaborativos requeridos para satisfacer las metas de negocio definidas.
3. El diseño de dichos procesos siguiendo el enfoque de modelado de procesos colaborativos basados en protocolos de interacción.
4. La derivación de las interfaces de negocio a implementar por los socios a partir de los protocolos de interacción.

Para ello, el metamodelo del lenguaje se ha estructurado en cuatro paquetes (Figura 4-1), los cuales proveen los elementos conceptuales que dan soporte a la definición de cuatro vistas: la *Vista de la Colaboración B2B*, la *Vista de Procesos de Negocio Colaborativos*, la *Vista de Protocolos de Interacción* y la *Vista de Interfaces de Negocio*. Cada vista provee una perspectiva diferente de una colaboración B2B y los procesos colaborativos que la componen.

A continuación se describen cada uno de los paquetes que conforman el metamodelo. En la definición de cada elemento conceptual se provee la semántica del mismo, y en el caso que corresponda se provee: los atributos, las asociaciones con los otros elementos y las restricciones asociadas al elemento.

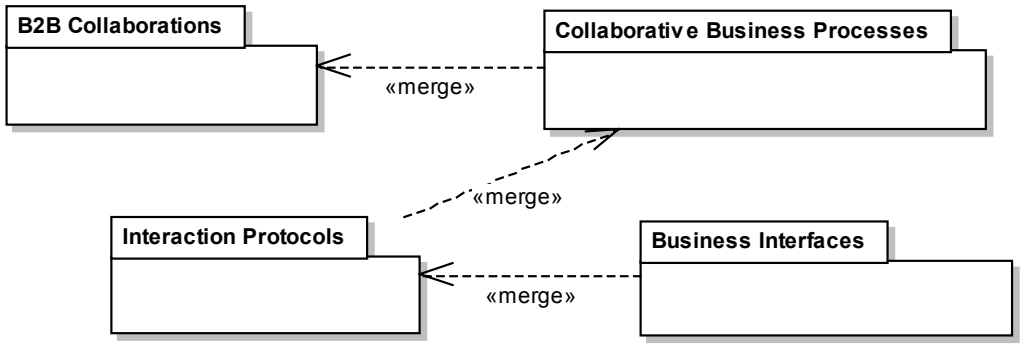


Figura 4-1. Metamodelo del lenguaje UP-ColBPIP

4.2.1.1. El Paquete B2B Collaborations

Este paquete especifica los elementos conceptuales para la definición de la *Vista de la Colaboración B2B*, la cual define los participantes de una colaboración B2B, los roles de los mismos y los requerimientos y metas de negocio ha ser cumplidos como parte de la aplicación de un modelo de colaboración para la gestión integrada de la cadena de suministro. La Figura 4-2, muestra la sintaxis abstracta del metamodelo

descrita por este paquete. Los elementos conceptuales provistos por el mismo son descritos a continuación.

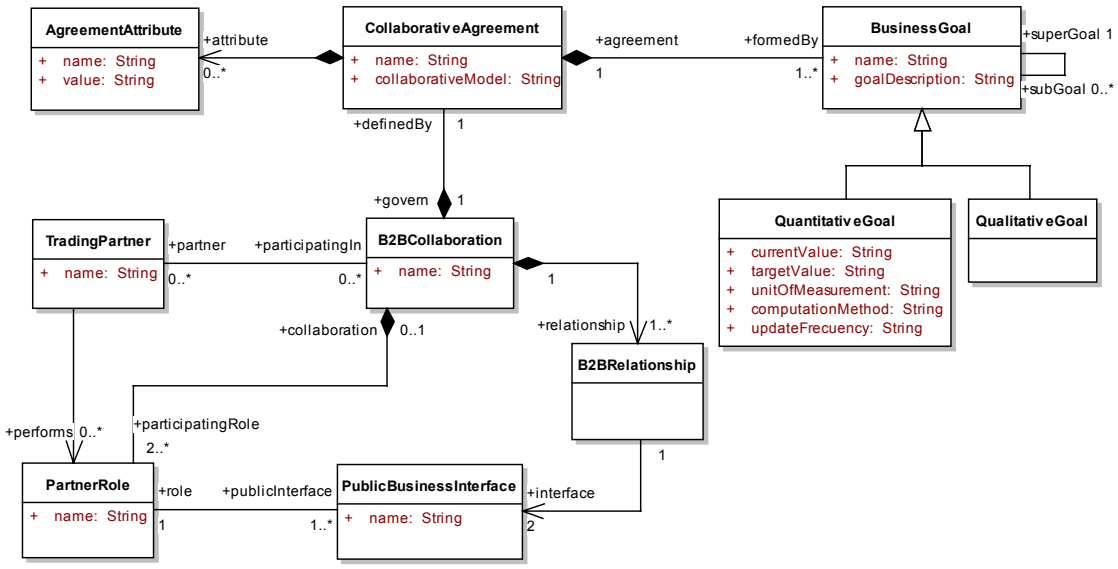


Figura 4-2. Sintaxis abstracta del paquete *B2B Collaborations*

B2B COLLABORATION (COLABORACIÓN B2B)

Semántica

Una colaboración B2B representa un conjunto de socios de negocio cooperando entre sí con el objetivo de llevar a cabo una gestión integrada de la cadena de suministro a través de la aplicación de un modelo de colaboración, el cual será soportado por tecnología de información. Define los roles que serán desempeñados por cada uno de los socios.

En una colaboración B2B, el vínculo de comunicación entre dos roles se especifica a través de una relación B2B, la cual implícitamente especifica la comunicación que debe existir entre los socios que desempeñan dichos roles. Para vincularse a través de relaciones B2B, los roles poseen interfaces de negocio públicas que representan el comportamiento público de los mismos. Dichas interfaces especifican las interacciones que pueden ocurrir entre dos roles.

En una colaboración B2B siempre deben definirse al menos dos roles. No obstante, ésta puede no tener socios de negocio definidos. Esto puede ser útil para definir colaboraciones B2B genéricas que pueden ser aplicadas a diferentes casos. Cuando un

par o conjunto de empresas desean entablar una colaboración B2B basada en una colaboración B2B genérica, deberán especificar los roles que las mismas desempeñarán.

Atributos

- *name: String*. Indica el nombre dado a la colaboración B2B.

Asociaciones

- *definedBy: CollaborativeAgreement [1]*. Referencia el acuerdo de colaboración que gobierna la colaboración B2B.
- *participatingRole: PartnerRole [2..*]*. Referencia el conjunto de roles que los socios de negocio pueden desempeñar en la colaboración B2B.
- *partner: TradingPartner [0..*]*. Referencia el conjunto de socios de negocio que participan en esta colaboración B2B desempeñando algún rol.
- *relationship: B2BRelationship [1..*]*. Referencia las relaciones B2B contenidas en esta colaboración B2B.
- *Process: CollaborativeBusinessProcess [0..*]*. Referencia los procesos colaborativos que definen el comportamiento de la colaboración B2B.
- *document: BusinessDocument [0..*]*. Referencia los documentos de negocio a ser intercambiados entre los roles en los diferentes procesos colaborativos.

Restricciones

[1] Si una colaboración B2B tiene definida socios de negocio, los mismos deben tener asociado un rol definido en la colaboración.

```
context B2BCollaboration
inv: C1
self.partner->forAll(p | p.performs -> exists(r | r =
self.participatingRole)
```

TRADING PARTNER (SOCIO DE NEGOCIO)

Semántica

Un socio de negocio representa una empresa que forma parte de una colaboración B2B, en la cual dicha empresa entabla una relación de negocio estrecha con otras empresas. Un socio de negocio desempeña un rol específico dentro de una colaboración B2B. No obstante, puede desempeñar más de un rol cuando está involucrado en más de una colaboración B2B.

Un socio de negocio deberá poseer una estructura interna, compuesta de procesos privados, para soportar las interfaces de negocio públicas del rol que está desempeñando. No obstante, la forma en que implementa las interfaces de negocio públicas asociadas al rol, está fuera del alcance del lenguaje UP-ColBPIP.

Atributos

- *name: String*. Indica el nombre del socio de negocio.

Asociaciones

- *participatingIn: B2Bcollaboration [0..*]*. Referencia la colaboración B2B en la que el socio está participando.
- *performs: PartnerRole [0..*]*. Referencia el conjunto de roles que desempeña en diferentes colaboraciones B2B.

Restricciones

[1] Un socio de negocio puede desempeñar un único rol en una colaboración B2B.

```
context TradingPartner
inv: C2
self.performs->forAll(r1,r2:PartnerRole | r1.collaboration <>
r2.collaboration)
```

PARTNER ROLE (ROL DEL SOCIO)

Semántica

Un rol define el papel que juega un socio de negocio en una colaboración B2B. El comportamiento público del rol es especificado por las interfaces de negocio públicas que éste posee. Un rol entabla una relación B2B con otro rol a través de una interfaz de negocio pública. Si un rol entabla varias relaciones con otros roles, por cada relación debe definirse una interfaz de negocio pública que exponga el comportamiento que el rol debe cumplir cuando interactúa con otro rol.

Atributos

- *name: String*. Indica el nombre del rol.

Asociaciones

- *collaboration: B2BCollaboration [0..1]*. Referencia la colaboración B2B que contiene al rol.

- *publicInterface: PublicBusinessInterface [1..*]*. Referencia la interfaz de negocio pública que soporta las responsabilidades del rol en una colaboración B2B.
- *participates: CollaborativeBusinessProcess [0..*]*. Referencia los procesos colaborativos en los cuales el rol participa.
- *protocol: InteractionProtocol [0..*]*. Referencia los protocolos de interacción en los cuales el rol participa.

PUBLIC BUSINESS INTERFACE (INTERFAZ DE NEGOCIO PÚBLICA)

Semántica

Una interfaz de negocio pública define el comportamiento público que un rol expone para colaborar con otro rol y representa el punto de interacción entre dos roles conectados por una relación B2B.

Una interfaz de negocio pública contiene dos interfaces: una interfaz de negocio provista y una interfaz de negocio requerida. Una interfaz de negocio requerida define el conjunto de servicios que un rol espera de otro rol, para poder realizar el envío de los mensajes requeridos en los protocolos de interacción en los que dichos roles están involucrados. Una interfaz de negocio provista define el conjunto de servicios que un rol ofrece y provee a otro rol, para poder recibir los mensajes de negocio requeridos en los protocolos de interacción en los que dichos roles están involucrados. Estas interfaces dan soporte al intercambio de mensajes entre los socios en los procesos colaborativos.

Atributos

- *name: String*. Indica el nombre dado a la interfaz de negocio pública.

Asociaciones

- *role: PartnerRole [1]*. Referencia el rol que la interfaz está soportando.
- *provided: BusinessInterface [1]*. Referencia la interfaz provista en la interfaz de negocio pública de un rol.
- *provided: BusinessInterface [1]*. Referencia la interfaz requerida en la interfaz de negocio pública de un rol.

B2B RELATIONSHIP (RELACIÓN B2B)

Semántica

Una relación B2B define un vínculo de comunicación entre dos roles, a través de la cual los socios intercambiarán mensajes como consecuencia de la ejecución de procesos colaborativos. La vinculación de los roles es realizada a través de las interfaces de negocio públicas que los roles definen para colaborar entre sí.

Asociaciones

- *interface: PublicBusinessInterface [2]*. Referencia las interfaces públicas de los dos roles que entablan una relación B2B.

COLLABORATIVE AGREEMENT (ACUERDO DE COLABORACIÓN)

Semántica

Un acuerdo de colaboración define en términos generales los parámetros y reglas que gobiernan una colaboración B2B. Estos atributos del acuerdo son definidos y acordados en forma conjunta por los socios involucrados en la colaboración B2B. Más importante aún, dicho acuerdo establece el modelo de colaboración a ser aplicado. Un acuerdo de colaboración también posee los requerimientos de negocio de una colaboración B2B, esto es, las metas de negocio que los socios han acordado cumplir.

Atributos

- *name: String*. Indica el nombre del acuerdo de colaboración.
- *collaborativeModel: String*. Indica el modelo de colaboración que los socios acuerdan llevar a cabo.

Asociaciones

- *attribute: AgreementAttribute[0..*]*. Referencia los atributos del acuerdo de colaboración.
- *formedBy: BusinessGoal[1..*]*. Referencia las metas de negocio que representan los requerimientos de la colaboración B2B.

AGREEMENT ATTRIBUTE (ATRIBUTO DEL ACUERDO)

Un atributo del acuerdo permite especificar los parámetros y reglas que los socios definen en un acuerdo de colaboración, tales como la duración de la colaboración, la

fecha de inicio de la misma, los productos sobre los que se va a colaborar, los precios acordados para los mismos, los niveles de inventario máximos y mínimos, etc.

Atributos

- *name: String*. Indica el nombre dado al atributo.
- *value: String.* Indica el valor del atributo.

BUSINESS GOAL (META DE NEGOCIO)

Semántica

Una meta de negocio representa un objetivo común a ser alcanzado por los socios en una colaboración B2B. También puede ser vista como un requerimiento a ser cumplido en la colaboración, el cual forma parte de la definición de un acuerdo de colaboración. A través de las metas de negocio los socios podrán luego evaluar el desempeño de una colaboración B2B, comparando los objetivos compartidos con los resultados obtenidos.

Las metas de negocio son definidas utilizando los patrones de metas propuestos en (Eriksson y Penker, 2000) para modelar metas organizacionales. Una meta de negocio puede estar compuesta de una jerarquía de submetas, en donde una meta padre tiene una dependencia con varias submetas. Esta jerarquía indica que las submetas deben ser alcanzadas para que la meta padre también pueda ser alcanzada. El nombre de una meta puede ser establecido en términos del estado deseado a alcanzar con respecto a alguna información compartida por los socios, o bien en términos que indiquen optimización, tales como incrementar las ventas, disminuir el inventario, etc.

Una meta de negocio debe poder ser medida para posibilitar el seguimiento de lo acordado en una colaboración B2B. Para ello, es descrita en base a una medida de rendimiento, la cual puede ser expresada en términos cuantitativos o bien en términos cualitativos. Esto es expresado por los subtipos de metas.

Toda meta de negocio sólo puede ser alcanzada a través de la ejecución de un proceso colaborativo. Por lo tanto, cuanto más clara y precisa una meta sea definida, más sencillo será identificar y definir el proceso colaborativo correspondiente para alcanzar tal meta.

Atributos

- *name:String*. Indica el nombre de la meta de negocio.
- *goalDescription: String*. Define el significado de la meta en lenguaje natural.

Asociaciones

- *agreement: CollaborativeAgreement [1]*. Referencia el acuerdo de colaboración que contiene la meta.
- *subGoal*: *BusinessGoal [0..*]*. Referencia las submetas de una meta.
- *superGoal: BusinessGoal[1]*. Referencia la meta padre de una meta.

QUANTITATIVE GOAL (META CUANTITATIVA)

Semántica

Una meta cuantitativa es una meta descrita en base a una métrica o medida de rendimiento acerca de alguna información compartida por los socios, la cual puede ser expresada en alguna unidad de medida específica, tal como volumen, cantidad, tiempo, etc. Una meta cuantitativa consiste del valor actual (si se conoce), el valor a alcanzar, la unidad de medida utilizada, el método de cálculo de la métrica y la frecuencia de evaluación de la misma (semanal, mensual, anual, etc.).

Los indicadores claves de rendimiento (conocidos como KPI's – Key Performace Indicators) definidos en algunos modelos de colaboración, tales como CPFR, pueden ser utilizados para expresar metas cuantitativas.

Atributos

- *currentValue:String*. Indica el valor actual de la información que se está considerando.
- *targetValue:String*. Indica el valor a alcanzar de la información que se está considerando.
- *unitOfMeasurement:String*. Indica la unidad de medida específica con la cual la información que se está considerando será medida.
- *computationMethod:String*. Documenta el método o fórmula de cálculo de la métrica, dado que el método debe ser acordado y definido también en forma conjunta por los socios.
- *updateFrecuency: String.* Indica la frecuencia de evaluación de la meta.

QUALITATIVE GOAL (META CUALITATIVA)

Semántica

Una meta cualitativa es descrita informalmente en lenguaje natural y el cumplimiento de la misma no depende de un valor específico sino del juicio de los socios sobre si la meta ha sido alcanzada.

4.2.1.2. El paquete Collaborative Business Proceses

Este paquete especifica los elementos conceptuales para la definición de la *Vista de Negocio Procesos Colaborativos*, la cual permite la identificación de los procesos de negocio necesarios a partir de los requerimientos definidos en la vista previa (Sección 4.2.1.1). La Figura 4-3, muestra la sintaxis abstracta del metamodelo descrita por este paquete. Los elementos conceptuales provistos por el mismo se describen a continuación.

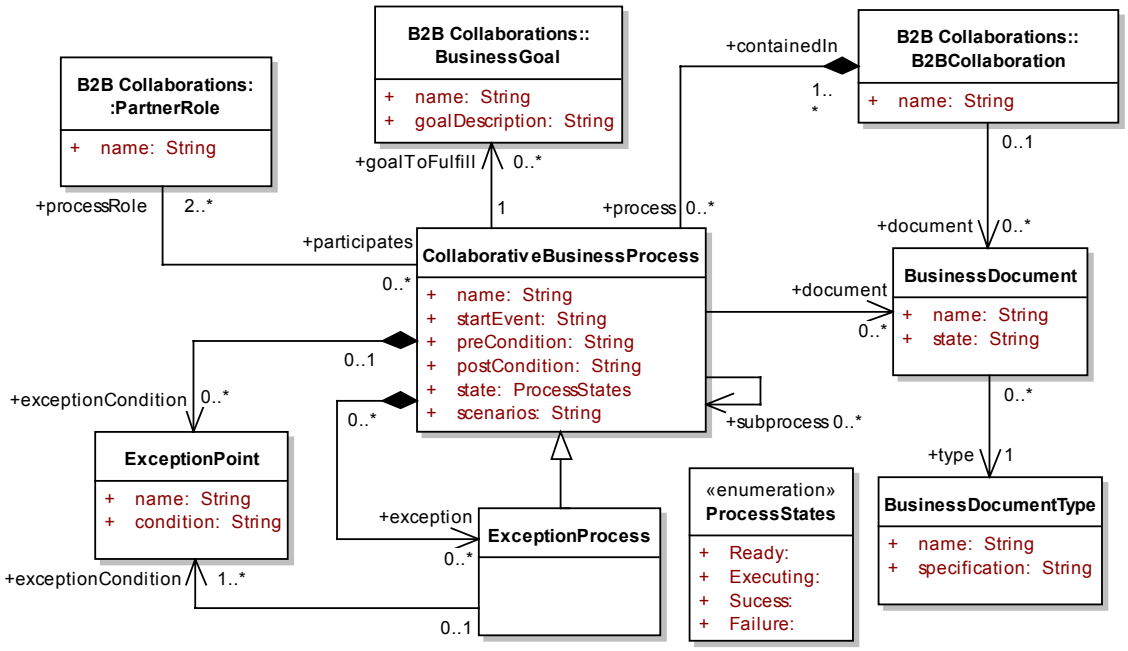


Figura 4-3. Sintaxis abstracta del paquete *Collaborative Business Processes*

COLLABORATIVE BUSINESS PROCESS (PROCESO DE NEGOCIO COLABORATIVO)

Semántica

Un proceso de negocio colaborativo define un conjunto de acciones ejecutadas por los socios para alcanzar una meta de negocio común, como así también la información a

ser intercambiada en dicho proceso. La especificación del mismo es realizada de una manera informal describiendo en lenguaje natural el escenario principal del proceso y los escenarios alternativos. De esta manera, un proceso colaborativo es utilizado para recolectar los requerimientos de la colaboración en cuanto a las acciones que deben realizar los roles y los documentos de negocio que deben ser intercambiados. La definición explícita del comportamiento del proceso, esto es, la coreografía de los mensajes a ser intercambiados estará especificada por el protocolo de interacción que materializa al proceso.

En un proceso colaborativo están involucrados uno o más roles desempeñados por los socios de negocio. Tanto los roles como los socios deben corresponderse con aquellos definidos en la colaboración B2B, de la cual el proceso forma parte.

Un proceso colaborativo puede tener asociado una o varias metas de negocio. El proceso es definido con el objetivo de que los socios puedan alcanzar dichas metas.

Un proceso colaborativo puede estar compuesto de un conjunto de subprocesos, lo cual significa que el comportamiento del proceso también contendrá el comportamiento provisto por otros procesos.

Además, un proceso colaborativo también puede estar compuesto de un conjunto de procesos de excepciones, los cuales especifican el comportamiento a seguir ante la ocurrencia de una excepción predefinida en el proceso. La condición de excepción a ser evaluada y el lugar dentro del proceso donde dicha excepción puede ocurrir es indicado por el punto de excepción asociado al proceso.

Atributos

- *name: String*. Indica el nombre del proceso colaborativo.
- *scenarios: String*. Indica el texto en lenguaje natural que expresa el escenario principal y los escenarios alternativos del proceso.
- *startEvent: String*. Describe el evento de inicio que proviene del rol que inicia el proceso. Este atributo puede ser definido en lenguaje natural o bien utilizando un lenguaje formal.
- *preCondition: String*. Describe una condición que debe ser satisfecha antes del inicio de un proceso. Esta condición puede estar expresada de acuerdo al estado de un documento de negocio o al estado de una instancia de un proceso

colaborativo, esto es, si la misma ha sido iniciada, está en curso o ha sido finalizada con éxito o con una falla. Este atributo puede ser definido en lenguaje natural o bien utilizando un lenguaje formal.

- *postCondition: String*. Describe una condición que debe ser cumplida después de la finalización de un proceso. Esta condición puede estar expresada de acuerdo a uno de los documentos de negocio involucrados en el proceso o de acuerdo al estado de una instancia de un subproceso. Este atributo puede ser definido en lenguaje natural o bien utilizando un lenguaje formal.
- *state: String*. Indica el estado de una instancia del proceso. El estado puede ser: *Ready*, que puede ser iniciado debido a que su precondición es satisfecha; *Ejecuting*, que fue iniciado y está en ejecución; *Success*, que fue finalizado con éxito, *Failure*, que fue finalizado sin éxito. Las transiciones entre dichos estados son mostradas en la Figura 4-4.

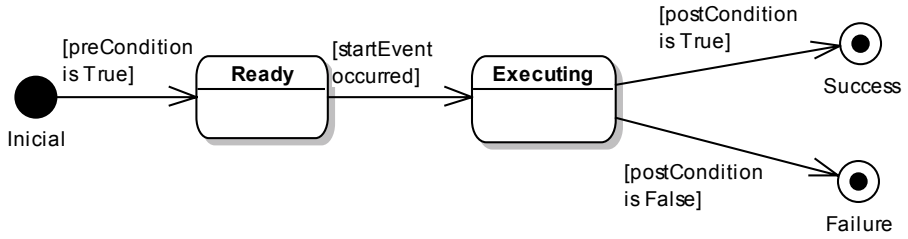


Figura 4-4. Estados posibles de una instancia de un proceso colaborativo

Asociaciones

- *goalToFulfill: BusinessGoal [0..*]*. Referencia las metas de negocio a ser alcanzadas a través de la ejecución del proceso colaborativo.
- *containedIn: B2BCollaboration [1..*]*. Referencia la colaboración B2B de la cual el proceso forma parte.
- *processRole: PartnerRole [2..*]* Referencia los roles que participan en el proceso.
- *document: BusinessDocument [0..*]*. Referencia los documentos de negocio a ser intercambiados entre los roles en el proceso colaborativo.
- *subprocess: CollaborativeBusibessProcess [0..*]*. Referencia los subprocesos que componen al proceso.
- *exception: ExceptionProcess [0..*]*. Referencia los procesos de excepción que el proceso tiene asociado.

- exceptionCondition: ExceptionPoint [0..*].
- *realizedBy: InteractionProtocol[0..1]*. Referencia el protocolo de interacción que es utilizado para expresar el comportamiento del proceso colaborativo.

Restricciones

[1] Tanto los roles como los socios deben corresponderse con aquellos definidos en la colaboración B2B, de la cual el proceso forma parte

```
context CollaborativeBusinessProcess
inv: C2
self.processRole->forAll(pr | self.containdIn.partipatingRole->
exists(cr | cr = pr)
```

[2] Los roles que forman parte de un proceso colaborativo deben ser diferentes.

```
context CollaborativeBusinessProcess
inv: C2
self.processRole->forAll(r1,r2:PartnerRole | r1 <> r2)
```

[3] Si un proceso tiene asociado un proceso de excepción también debe tener asociado un punto de excepción, y el mismo debe estar asociado a dicho proceso de excepción.

```
context CollaborativeBusinessProcess
inv: C3
self.exception->forAll(p1 | self.exceptionCondition =
p1.exceptionCondition)
```

EXCEPTION PROCESS (PROCESO DE EXCEPCIÓN)

Semántica

Un proceso de excepción es un tipo específico de proceso colaborativo. Éste es utilizado para gestionar una excepción determinada que puede ocurrir en un proceso colaborativo. El proceso de excepción indica cómo el proceso colaborativo es finalizado o corregido en caso que tal excepción ocurra.

Aunque un proceso de excepción puede ser definido independientemente de un proceso colaborativo, éste tiene significado solamente cuando está asociado a un proceso colaborativo. La excepción es contemplada en tiempo de diseño y la condición de excepción a ser evaluada es indicada por el punto de excepción asociado al proceso.

Asociaciones

- *exceptionCondition: ExceptionPoint[1..*]*. Referencia la condición a ser evaluada para la activación del proceso de excepción.

EXCEPTION POINT (PUNTO DE EXCEPCIÓN)

Semántica

Un punto de excepción representa una condición de excepción a ser evaluada, como así también el lugar donde la excepción puede ocurrir en el comportamiento de un proceso colaborativo. El punto de excepción indica que cuando la condición de excepción es evaluada en *True* dentro de un proceso colaborativo, el proceso de excepción que tiene asociado este punto de excepción será activado. La activación de este proceso tiene como objetivo gestionar dicha excepción en el proceso colaborativo que también tiene definido este punto de excepción.

Atributos

- *name: String*. Indica el nombre del proceso colaborativo.
- *condition*. Define la condición a ser evaluada para determinar la ocurrencia de una excepción.

BUSINESS DOCUMENT (DOCUMENTO DE NEGOCIO)

Semántica

Un documento de negocio representa la información que es intercambiada por los roles dentro de un proceso colaborativo. Aunque los documentos a ser intercambiados entre los socios son identificados cuando se definen los procesos colaborativos, estos forman parte de una colaboración B2B específica, con el objetivo de que los mismos puedan ser utilizados en uno o más procesos colaborativos.

El propósito del lenguaje UP-ColBPIP no es definir la estructura de los documentos de negocio, sino solamente hacer referencia a los mismos. La estructura sintáctica y semántica de los documentos de negocio debería ser definida por un método específico para el modelado de documentos de negocio (Caliusco y otros, 2004). O bien, la definición sintáctica de los documentos debería ser provista por un estándar B2B que provea tipos de documentos de negocio estándares. Por lo tanto, el documento de negocio también posee un tipo, que referencia uno de los tipos de documentos

específicos del estándar B2B utilizado o bien, un tipo de documento de negocio definido por el usuario.

Atributos

- *name: String*. Indica el nombre del documento de negocio.
- *state: String*. Indica el estado del documento. Los posibles estados del mismo son definidos por el usuario debido a que los mismos dependerán de la semántica que tenga el documento. Estos estados pueden ser utilizados para definir diferentes condiciones, como por ejemplo condiciones de excepciones o precondiciones y poscondiciones de los procesos.

Asociaciones

- *type: BusinessDocumentType [1]*. Referencia el tipo de documento de negocio

BUSINESS DOCUMENT TYPE (TIPO DE DOCUMENTO DE NEGOCIO)

Semántica

Este concepto representa un tipo de documento de negocio provisto por un estándar B2B o definido por el usuario. El nombre del tipo de documento se corresponderá con el nombre del esquema XML, el cual indica la estructura sintáctica del documento. Los tipos de documentos de negocio no forman parte de una colaboración B2B específica. Los mismos pueden ser reutilizados para definir documentos de negocio en uno o más procesos colaborativos de una o más colaboraciones B2B. Dos documentos de negocio podrían ser definidos en base al mismo tipo de documento de negocio.

Atributos

- *name: String*. Indica el nombre del documento de negocio.
- *specification: String*. Referencia el esquema XML que define la estructura sintáctica del tipo de documento de negocio.

4.2.1.3. El paquete Interaction Protocols

Este paquete especifica los elementos conceptuales para la definición de la *Vista de Protocolos de Interacción*, la cual permite la especificación del comportamiento de los procesos colaborativos a través de protocolos de interacción. La Figura 4-5, muestra la sintaxis abstracta del metamodelo descrita por este paquete. Los elementos conceptuales provistos por el mismo se describen a continuación.

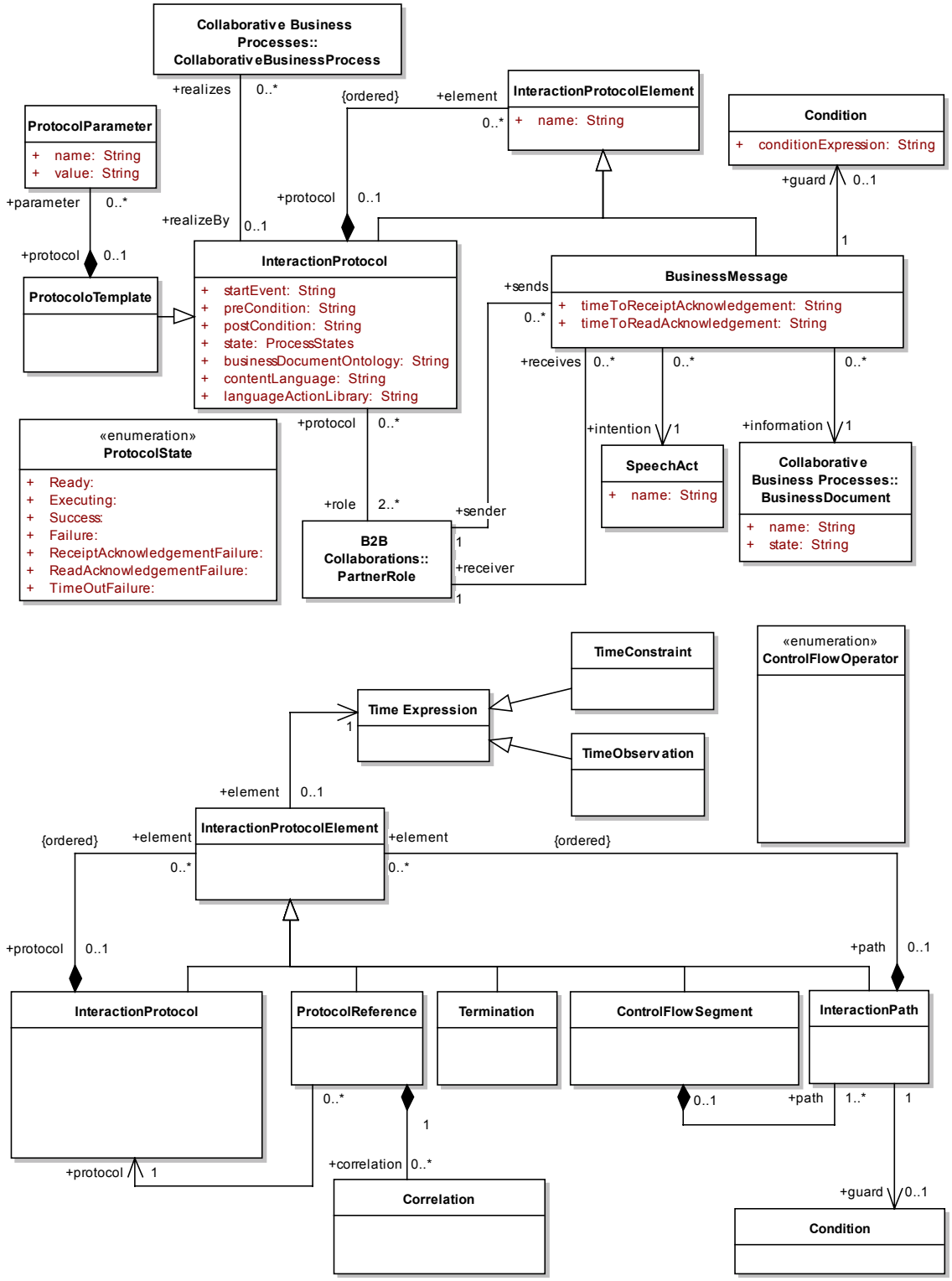


Figura 4-5. Sintaxis abstracta del paquete Interaction Protocols

INTERACTION PROTOCOL (PROTOCOLO DE INTERACCIÓN)

Semántica

Un Protocolo de Interacción describe un patrón de comunicación de alto nivel a través de una coreografía de mensajes entre dos o más roles desempeñados por socios de negocio. Un protocolo de interacción define las interacciones entre los roles desde una perspectiva global, describiendo las interacciones peer-to-peer entre los roles y las responsabilidades de cada uno de ellos en términos de los mensajes que pueden enviar y recibir.

La semántica de comportamiento del protocolo de interacción, es decir la coreografía de mensajes del mismo, está dada por la secuencia ordenada de los siguientes elementos: los mensajes, los segmentos de flujo de control, los caminos de interacción de los segmentos, las referencias a otros protocolos y las terminaciones. El orden de los elementos de un protocolo representa el orden de los mismos en el tiempo.

Los primeros atributos de un protocolo de interacción son aquellos que han sido definidos en un proceso de negocio colaborativo. Esto se debe a que cuando un protocolo de interacción está materializando a un proceso colaborativo, los atributos del protocolo deben ser derivados a partir de los del proceso.

Además, un protocolo tiene otros tres atributos: *businessDocumentOntology*, *contentLanguage* y *languageActionLibrary*. El primero referencia la ontología que describe la semántica de los documentos de negocio involucrados en el protocolo (Caliusco y otros, 2004b). Esto puede ayudar a los diseñadores a comprender los documentos que son intercambiados. No obstante, la definición de la semántica del documento está fuera del alcance de UP-ColBPIP. El atributo *contentLanguage* referencia el estándar B2B de contenido que describe la sintaxis de los documentos de negocio. Posibles estándares de contenido son: OAGIS [3] o UBL [4] , los cuales proveen documentos genéricos que pueden ser aplicados a diferentes dominios de cadena de suministro. Otros estándares, tales como (CompTIA EIDX, 2004) y CIDX (CIDX,

[3] OAGIS, http://www.openapplications.org

[4] UBL, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl

2004), proveen documentos para un dominio en particular, como lo son las cadenas de suministro de las industrias electrónica y química. Los documentos de negocio definidos en la vista previa deberían estar basados en uno de estos estándares. El atributo *languageActionLibrary* referencia la librería de actos de comunicación utilizada para definir los mensajes basados en actos de comunicación. Esta librería debería definir los actos de comunicación que pueden ser utilizados en el protocolo como así también la semántica de cada uno. Ejemplo de tal librería es FIPA ACL (Agent Communication Language) (FIPA, 2002), la cual define el conjunto de actos de comunicación que es posible utilizar para la comunicación entre agentes de software. No obstante, los socios que entablan una colaboración B2B pueden optar por definir su propia librería o utilizar una ya existente. La ventaja de utilizar librerías estándares y conocidas de actos de comunicación radica en que esto posibilita un mayor reuso de los protocolos y un mejor entendimiento de los mismos. Como ejemplo de tal tipo de librería, en el Anexo A se describe la semántica de un subconjunto de los actos de comunicación provistos por FIPA ACL. Estos actos de comunicación son los que se han utilizado en todos los ejemplos de protocolos descriptos en esta tesis.

Atributos

- *startEvent: String*. Describe el evento de inicio y el rol que comienza el protocolo.
- *preCondition: String*. Describe una condición que debe ser satisfecha antes del inicio del protocolo. Esta condición puede estar expresada de acuerdo al estado de un documento de negocio o al estado de la instancia de otro protocolo de interacción, esto es, si la misma ha sido iniciada, está en curso o ha sido finalizada. Este atributo puede ser definido en lenguaje natural o bien utilizando un lenguaje formal.
- *postCondition: String*. Describe una condición que debe ser cumplida después de la finalización de un protocolo. Esta condición puede estar expresada de acuerdo a uno de los documentos de negocio involucrados en el proceso que el protocolo está materializando o de acuerdo al estado de la instancia de otro protocolo. Este atributo puede ser definido en lenguaje natural o bien utilizando un lenguaje formal.
- *state: ProtocolState.* Indica el estado de una instancia del protocolo (Figura 4-6). El estado puede ser: *Ready*, que puede ser iniciado debido a que su

precondición es satisfecha; *Executing*, que fue iniciado y está en ejecución; *Success*, que fue finalizado con éxito, *Failure*, que fue finalizado sin éxito. Además de estos estados finales, el protocolo puede finalizar en los estados *ReceiptAcknowledgementFailure, ReadAcknowledgementFailure* o *TimeOutFailure*. Estos estados representan la finalización del protocolo como consecuencia de excepciones. El primero de ellos indica que no se ha recibido un acuse de recibo de recepción para un mensaje dentro del período de tiempo establecido. El segundo indica que no se ha recibido un acuse de recibo de lectura para un mensaje dentro del período de tiempo establecido. El último indica una excepción de tiempo, la cual ocurre debido a que un elemento del protocolo (mensaje, segmento, etc.) no se realizó dentro del período de tiempo establecido por la restricción de tiempo asociada.

- *businessDocumentOntology: String*. Referencia la ontología que describe la semántica de los documentos de negocio involucrados en el protocolo.
- *contentLanguage: String*. Referencia el estándar B2B de contenido que describe la sintaxis de los documentos de negocio involucrados en el protocolo.
- *languageActionLibrary: String*. Referencia la librería de actos de comunicación utilizada para definir los mensajes del protocolo.

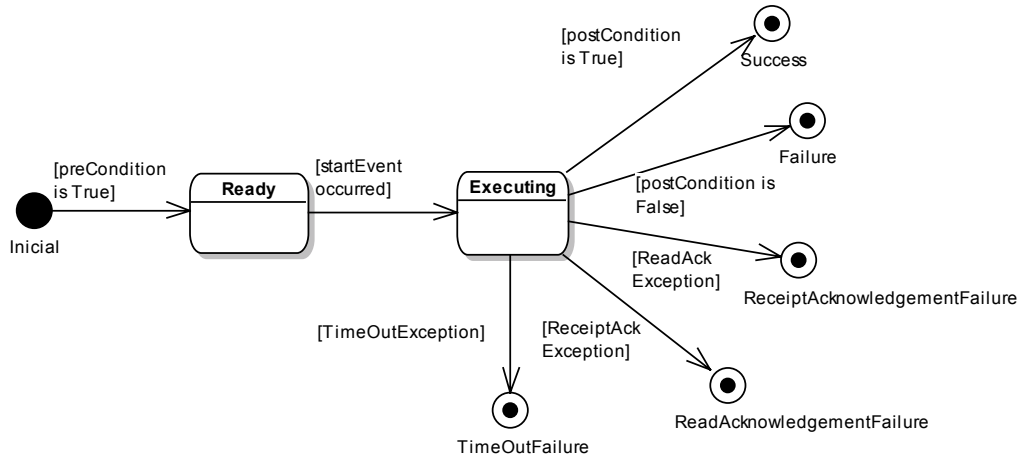


Figura 4-6. Estados posibles de una instancia de un protocolo de interacción

Asociaciones

- *realizes: CollaborativeBusinessProcess [0..*]*. Referencia el proceso colaborativo que el protocolo de interacción está materializando. Un protocolo podría expresar el comportamiento de uno o más procesos colaborativos.

- *element: InteractionProtocolElement [0..*]*: Referencia el conjunto ordenado de elementos que componen el protocolo de interacción. El orden establecido en este atributo de asociación indica el orden en el tiempo de los componentes del protocolo.
- *role*: *PartnerRole [2..*]*. Referencia los roles involucrados en el protocolo, los cuales están asociados a alguno de los socios involucrados en la colaboración B2B. Dos o más roles pueden formar parte de un protocolo de interacción.
- *parameter: ProtocolParameter [0..*]*. Referencia los parámetros que contiene el protocolo de interacción.

Restricciones

[1] Los roles del protocolo deben corresponderse con aquellos definidos en el proceso colaborativo que el protocolo está materializando, excepto cuando el protocolo es una plantilla de protocolo (Protocol Template).

```
context InteractionProtocol
inv: C2
not self.oclIsTypeOf(ProtocolTemplate) implies self.role->forAll(r
| self.realizes.processRole-> exists(pr | pr = r)
```

INTERACTION PROTOCOL ELEMENT(ELEMENTO PROTOCOLO DE INTERACCIÓN)

Semántica

Un elemento del protocolo de interacción es un concepto abstracto que representa un componente o elemento de un protocolo de interacción. La semántica concreta de cada elemento está dada por la semántica de los subtipos de elementos específicos. Un elemento puede estar contenido en un protocolo de interacción y también puede estar contenido en un camino de interacción de un segmento del protocolo.

Atributos

- *name: String*. El nombre del elemento.

Asociaciones

- *protocol: InteractionProtocol [0..1]*. Referencia el protocolo de interacción que está conteniendo al elemento.
- *path: InteractionPath [0..1]*. Referencia al camino de interacción que está conteniendo al elemento.

BUSINESS MESSAGE (MENSAJE DE NEGOCIO)

Semántica

Un mensaje de negocio define una interacción o comunicación entre dos roles, el que envía el mensaje (el remitente) y el que lo recibe (el receptor). La semántica concreta de cada mensaje está dada por la semántica del acto de comunicación asociado al mensaje. Además, un mensaje de negocio indica la transferencia de información de un rol hacia otro. El contenido que transporta el mensaje es indicado por el documento de negocio asociado al mismo.

Un mensaje de negocio expresa que el rol remitente ha realizado una acción que ha generado el envío de un acto de comunicación que representa su intención en la interacción, e implica la expectativa del remitente de que el receptor reaccione de acuerdo a la semántica del acto de comunicación.

Además, un mensaje de negocio es una comunicación asincrónica y en un solo sentido. La comunicación asincrónica es esencial en interacciones B2B, debido a que el rol que envía el mensaje no está subordinado a la respuesta del rol receptor, con lo cual el control interno que un socio posee con respecto al protocolo no está subordinado al control de los otros socios. Esto implica que el remitente puede enviar el mensaje y continuar con su ejecución interna, sin necesidad de esperar una respuesta del receptor. No obstante, un mensaje de negocio podría requerir el envío de diferentes acuses de recibo por parte del rol receptor, con el objetivo de indicar al rol remitente que el mensaje ha sido leído y/o entendido por el rol receptor. De esta manera, los acuses de recibo pueden ser utilizados para garantizar la sincronización del estado entre dos roles, después del intercambio de un mensaje entre ellos.

Por otra parte, una condición puede estar asociada a un mensaje para indicar cuándo el mensaje puede ser enviado. Además, un mensaje de negocio puede ser enviado varias veces. En este caso, los acuses de recibo deben ser retornados por el receptor por cada vez que es enviado el mensaje. La repetición de un mensaje es definida por una condición asociada al mismo, la cual debe ser del tipo “N:Condición”, donde “N” representa el número de veces que el mensaje puede ser repetido y “Condición” representa la expresión condicional a ser evaluada que determina cuándo el mensaje puede ser enviado nuevamente. No obstante, un mensaje a ser repetido puede tener únicamente el número de veces que éste tiene que ser enviado, sin una expresión

condicional asociada. En este caso el número de repetición debe ser un número natural absoluto.

Atributos

- *timeToReceiptAcknowledgement: String*. Cuando este atributo tiene valor, significa que se ha acordado que el receptor del mensaje tiene que enviar un acuse de recibo, dentro del tiempo indicado en el atributo. El envío de un acuse de recibo indicará al remitente que el mensaje ha sido recibido con éxito por el receptor.
- *timeToReadAcknowledgement: String*. Cuando este atributo tiene valor, significa que se ha acordado que el receptor del mensaje tiene que enviar un acuse de lectura cuando procesa el mensaje, dentro del tiempo indicado en el atributo. Esto indicará al remitente que el mensaje ha sido entendido y procesado por el receptor, y de esta manera el remitente estará completamente seguro de que podrá continuar con la lógica del protocolo.

Asociaciones

- *sender: PartnerRole [1]*. Referencia el rol que envía el mensaje de negocio.
- *receiver: PartnerRole [1]*. Referencia el rol que recibe el mensaje de negocio.
- *intention: SpeechAct [1]*. Referencia el acto de comunicación asociado al mensaje de negocio, el cual representa la intención del remitente con respecto al documento de negocio intercambiado en el mensaje.
- *information: BusinessDocument [1]*. Referencia el documento de negocio a ser transportado por el mensaje de negocio, representando el intercambio de información entre los roles.
- *guard:Condition [0..1]*. Referencia la expresión condicional asociada a un mensaje, la cual es utilizada para restringir el envío del mensaje y/o para indicar que un mensaje puede ser enviado N veces. Cuando la evaluación de dicha expresión retorna *True*, el mensaje puede ser enviado. La expresión puede ser expresada en lenguaje natural o en un lenguaje interpretable por software.

Restricciones

[1] Un mensaje de negocio no puede tener como remitente y receptor al mismo rol.

```
context BusinessMessage
inv: C5
  self.sender <> self.receiver
```

[2] El nombre de un mensaje de negocio debe ser el mismo al del acto de comunicación asociado.

```
context BusinessMessage
inv: C6
   self.name = self.intention.name
```

[3] El documento de negocio asociado al mensaje debe corresponderse con uno de los documentos de negocio definidos en el proceso colaborativo que el protocolo que contiene al mensaje está materializando.

```
context BusinessMessage
inv: C7
self.information->forAll(b |
self.interactionProtocol.realizes.document-> exists(d|d.name =
b.name)
```

SPEECH ACT (ACTO DE COMUNICACIÓN)

Semántica

Un acto de comunicación representa la actitud o intención (tales como aceptar, proponer, rechazar, informar, etc.) que un rol tiene con respecto a la información que es trasportada en un mensaje de negocio. Un acto de comunicación permite que un mensaje no sólo sea entendido por la información que es transportada en el mismo, sino también por la intención que el rol remitente tiene con respecto a la información enviada en el mensaje. Además, el acto de comunicación asociado a un mensaje representa la acción que el remitente del mensaje ha realizado, la cual es comunicada al receptor y a través de la cual se crea un compromiso entre las partes. Por ejemplo, un mensaje con el acto *propose* representa una propuesta con el compromiso del remitente de realizar una determinada acción, la cual, para que posteriormente se haga efectiva deberá ser aceptada por el receptor. Para ello, el receptor puede enviar un acto *accept-proposal* indicando la aceptación de la propuesta. Esto finalmente representa el compromiso establecido entre ambos roles para que se realice la acción propuesta inicialmente.

Un acto de comunicación es independiente del mensaje, ya que el mismo puede estar asociado a varios mensajes con contenido diferentes.

Atributos

- *name: String*. El nombre del acto de comunicación.

PROTOCOL REFERENCE (REFERENCIA DE PROTOCOLO)

Semántica

Una referencia de protocolo hace referencia a otro protocolo de interacción, denominado el subprotocolo o protocolo anidado. Una referencia de protocolo indica que un protocolo de interacción contiene una llamada a un subprotocolo, la cual representa la inserción del comportamiento del subprotocolo dentro del protocolo. Cuando el subprotocolo es ejecutado, el protocolo que lo invoca queda esperando hasta que el subprotocolo finalice.

Un subprotocolo puede ser utilizado por varias razones: para el caso en que se necesite repetir un patrón de mensajes dentro de uno o varios protocolos, o bien para hacer un diseño más modular de los protocolos. La semántica interna de la referencia de protocolo está dada por el protocolo al que hace referencia.

El uso de subprotocolos debe ser realizado en forma precisa. Esto significa que los roles del subprotocolo deben ser los mismos del protocolo de interacción que lo invoca, o bien se debe establecer una correspondencia entre los mismos. Esta correspondencia puede ser definida a través de una correlación, en donde el nombre de un rol del subprotocolo se corresponde con el nombre de uno de los roles del protocolo que lo invoca.

Además, también es posible establecer una correlación entre los documentos de negocio utilizados en el protocolo y su subprotocolo. Un mismo documento de negocio puede ser utilizado en un protocolo y en sus subprotocolos, o bien dos documentos del mismo tipo pueden estar siendo utilizados, uno en el protocolo y el otro en el subprotocolo. Cuando esto ocurre, en algunos casos es conveniente indicar que la instancia del documento de negocio que está siendo manejada por la instancia de un protocolo, debe ser la misma que la manejada en la instancia de los subprotocolos. Esto es como consecuencia de que pueden existir en un mismo momento diferentes instancias del protocolo que se están ejecutando al mismo tiempo.

La correlación de documentos de negocio no significa que el documento de un protocolo es transferido al subprotocolo, es decir, esto no significa un pasaje por referencia o por valor a un parámetro, como en un lenguaje de programación. Pasajes por valor o por referencia de objetos en procesos colaborativos débilmente acoplados no

es apropiado. En su lugar, una correlación de documentos indica que en un subprotocolo, la instancia de un documento debe ser la misma instancia del documento del protocolo que lo invoca. No obstante, cómo se identifican las instancias de los documentos en las instancias de los protocolos no es un detalle que deba ser definido en este nivel de abstracción, sino que debe ser manejado por el lenguaje dependiente de la tecnología en el que se implemente el protocolo.

Asociaciones

- *protocol: InteractionProtocol [1]*. Referencia el protocolo anidado.
- *correlation: Correlation [0..*]*. Referencia las correlaciones entre elementos del protocolo y del subprotocolo.

CORRELATION (CORRELACIÓN)

Semántica

Una correlación de una referencia de protocolo expresa la correlación entre un rol o un documento de negocio de un protocolo y su subprotocolo. El valor del atributo *subProtocolElement* debe corresponderse con el nombre de un rol o un documento de negocio utilizado en el subprotocolo. El valor del atributo *protocolElement* debe corresponderse con el nombre de un rol o un documento de negocio utilizado en el protocolo que contiene la referencia de protocolo.

Atributos

- *subProtocolElement: String*. Indica el nombre del rol o documento de negocio del subprotocolo.
- *protocolElement: String*. Indica el nombre del rol o documento de negocio del protocolo.

CONTROL FLOW SEGMENT (SEGMENTO DE CONTROL)

Semántica

Un segmento de flujo de control representa una secuencia de mensajes compleja. Aunque un protocolo de interacción puede tener un único camino que consista de una secuencia de mensajes ordenados, generalmente esto no es común en las interacciones que ocurren en un proceso colaborativo. Los procesos colaborativos pueden consistir de

negociaciones complejas, en donde los socios deben considerar diferentes caminos alternativos, de acuerdo a su lógica de negocio interna y a sus intenciones. Por lo tanto, los segmentos de control permiten representar varios caminos de interacción posibles dentro de la coreografía de un protocolo. Un segmento de control está compuesto de uno o varios caminos de interacción, los cuales a su vez pueden contener otros segmentos de control, subprotocolos, secuencia de mensajes, etc., es decir, cualquier otro elemento de un protocolo de interacción.

La semántica del segmento de flujo de control depende del operador de flujo de control utilizado. La enumeración *ControlFlowOperator* especifica los diferentes tipos de operadores que pueden ser utilizados para definir la coreografía de un protocolo. Con el objetivo de proveer operadores bien conocidos, la mayoría de los operadores definidos están basados en los operadores utilizados en los lenguajes de modelación de procesos de negocio tradicionales. Además, estos operadores soportan los patrones de workflow básicos utilizados para definir el flujo de control en un proceso (van der Aalst y otros, 2003). Muchos de los lenguajes de modelado de procesos de negocio han sido comparados con respecto a estos patrones. A continuación se describen cada uno de los operadores de control.

And

El operador *And* representa la ejecución en paralelo de diferentes caminos, los cuales pueden ser ejecutados en cualquier orden. Esto permite la representación de interacciones concurrentes. La ejecución de los mensajes de los diferentes caminos puede ser intercalada en cualquier orden mientras se respete el orden de los mensajes en cada camino. Este operador es equivalente al patrón *Parallel* utilizado en lenguajes de workflow.

Xor

El operador *Xor* define que varios caminos alternativos son posibles y sólo uno de ellos será ejecutado. Esto significa que un rol tendrá que determinar a lo sumo un camino a ejecutar para continuar la interacción. Cada camino alternativo posee una condición de control. La elección de un camino está basada en la condición de control, es decir, que la misma sea evaluada y retorne *True*. Por lo tanto, a lo sumo una de las condiciones debe retornar *True*. Si ninguna condición evaluada retorna *True*, entonces el segmento de control no será ejecutado. En el caso que al menos un camino siempre

deba ser ejecutado, es posible agregar otro camino alternativo con la condición *else*, que represente la continuación del protocolo en caso que las condiciones asociadas a los caminos alternativos retornen *False*. También, es posible definir caminos alternativos con condiciones implícitas, es decir caminos que no tienen condición asociada. Esto podría ser requerido en un proceso colaborativo como consecuencia de que los socios no desean expresar las condiciones o eventos explícitamente en el proceso, o bien no desean definir la información compartida sobre la cual deberá definirse las condiciones. Este operador es equivalente al patrón *Exclusive-Choice* utilizado en lenguajes de workflow. Cuando se utiliza este operador con condiciones implícitas, éste es equivalente al patrón *Deferred Choice* utilizado en lenguajes de workflow.

Or

El operador *Or* representa dos o más caminos alternativos que pueden ser ejecutados y al menos uno de ellos deberá ser ejecutado. No obstante, es posible que todos los caminos puedan ser ejecutados. Los caminos a ser ejecutados deben tener su condición evaluada en *True*. Este operador es equivalente al patrón *Multi-Choice* utilizado en lenguajes de workflow.

If

El operador *If* representa un camino que puede ser ejecutado en el caso que su condición sea evaluada en *True* o bien puede que no sea ejecutado nunca debido a que la evaluación de su condición retorna *False*. Es posible agregar un camino con la condición *else* que represente la negación del anterior, y el cual será ejecutado en el caso que la condición del primero retorne *False*. En este caso, este operador se comporta de manera similar a un segmento con el operador *Xor* con dos caminos, en donde uno de ellos es el camino alternativo *else*.

Loop

El operador *Loop* representa un camino de interacción que puede ser ejecutado varias veces mientras la condición asociada al loop sea satisfecha. Dos tipos de loop pueden ser realizados: un loop “For”, en donde los caminos y los mensajes del loop deben ser ejecutados al menos una vez; y un loop “While”, en donde los caminos pueden ser ejecutados cero o N veces. Para el primer caso, la condición del camino debe comenzar con la expresión “(1,n)”. En el segundo caso, la condición del camino debe

comenzar con la expresión "(0,n)". Este operador es equivalente al patrón *Arbitrary Cycles* utilizado en lenguajes de workflow.

Transaction

El operador *Transaction* representa una transacción de negocio, en donde los mensajes y caminos dentro del segmento de control deben ser ejecutados en forma atómica y los mismos no pueden ser intercalados con otros mensajes de otros caminos. Transacciones de negocio pueden ser usadas para asegurar que el intercambio de información sea realizado en forma atómica. Por ejemplo, una transacción podría involucrar tanto el envío de información acerca de la entrega de productos o de pagos de los mismos como así también la respuesta del receptor, con el objetivo de que estos mensajes se realicen en forma atómica. Si una falla ocurriera, la transacción podría ser reanudada o bien finalizar con una falla, la cual indica que los mensajes que ya se realizaron (posibles compromisos ya asumidos) dentro de la transacción deben ser descartados. Los roles involucrados en la transacción son los encargados de asegurar que la misma se realice en forma atómica.

Exception

El operador *Exception* representa el camino a seguir como consecuencia de una excepción ocurrida, la cual es identificada en tiempo de diseño como una situación anormal en el protocolo. El camino debe tener asociada la condición de excepción, la cual debe ser evaluada en *True* para que éste se ejecute. El objetivo de un segmento de control con este operador es gestionar una excepción que puede ocurrir en un punto específico dentro del protocolo. Luego de la ejecución del camino de excepción, la ejecución del protocolo continúa.

Stop

El operador *Stop* representa el camino a seguir como consecuencia de una excepción ocurrida, la cual requiere la abrupta terminación del protocolo. El camino debe tener asociada una condición que debe ser evaluada en *True* para que éste se ejecute. Después de la ejecución de este camino la instancia del protocolo no podrá ser reiniciada. Al igual que el operador *Exception*, éste es utilizado para gestionar una excepción que puede ocurrir en un punto específico dentro del protocolo. No obstante,

difiere del operador *Exception* en que el protocolo es finalizado después de la ejecución del camino de excepción.

Cancel

El operador *Cancel* representa el camino a seguir como consecuencia de una excepción ocurrida, que a diferencia de los operadores *Stop* y *Exception* dicha excepción puede ocurrir en cualquier punto del protocolo de interacción. Este operador significa que después de la ejecución del camino de interacción que maneja la excepción, el protocolo finaliza. Un segmento de control con el operador *Cancel* es utilizado para que un protocolo pueda terminar en una forma coherente entre las partes. Las excepciones definidas en el mismo son aplicables a todos los mensajes y elementos del protocolo. Por lo tanto, un protocolo de interacción sólo puede tener definido un único segmento de control con el operador *Cancel*, y el mismo, debe ser definido al final del protocolo.

Un segmento de control con el operador *Cancel* puede consistir de varios caminos, cada uno definido para gestionar una excepción específica. Cada camino debe tener asociado la condición de excepción a ser evaluada, la cual determina la activación de la excepción. Dos tipos de excepciones son posibles gestionar: excepciones de tiempo y excepciones con respecto a la lógica del protocolo.

La excepción de tiempo debe ser definida en un camino que contenga una condición con el valor *TimeException*. Éste camino define la secuencia de mensajes a ser realizada en el caso que una restricción de tiempo, asociada a cualquier elemento de un protocolo, no sea satisfecha. Este camino es aplicable a todas las restricciones de tiempo definidas en el protocolo. Por lo tanto, sólo es posible definir una excepción de tiempo para un protocolo. Cualquier restricción de tiempo definida que no se cumpla en la ejecución del protocolo será manejada por este camino de interacción. Además, este camino también puede ser utilizado para gestionar excepciones de tiempo que ocurran con respecto al tiempo definido para los acuses de recibo de los mensajes.

El otro tipo de excepción es representado en un camino que contiene la condición a ser evaluada para activar la excepción. El camino define la secuencia de mensajes a ser realizada en el caso que la condición evaluada sea verdadera. Dicho camino representa la lógica de compensación del protocolo como consecuencia de la ocurrencia de una determinada excepción. La excepción es aplicable a todos los mensajes y elementos del

protocolo, es decir, puede ocurrir en cualquier punto del protocolo. Por ejemplo, es posible definir una excepción a ser gestionada en el caso que el contenido de un mensaje no fuera entendido sintáctica o semánticamente por el receptor. También es posible definir una excepción a ser gestionada cuando el receptor no comprenda el acto de comunicación asociado al mensaje o cuando éste esperaba otro acto de comunicación. Este tipo de excepciones también puede enfocarse en excepciones acerca de la lógica de negocio del protocolo. Aquellas excepciones que puedan ocurrir como consecuencia de fallas en las aplicaciones o en la red, como por ejemplo la pérdida de mensajes, no son consideradas en este nivel y las mismas deben ser gestionadas por los sistemas de los socios.

Atributos

- *operator: ControlFlowOperator*. Especifica el operador de control que define la semántica de un segmento de control.

Asociaciones

- *path:InteractionPath [1..*]*. Referencia el conjunto de caminos de interacción que forman parte del segmento de control.

Restricciones

[1] Un segmento de flujo de control con el operador *Loop, Exception* o *Stop* sólo puede contener un solo camino de interacción.

```
context ControlFlowSegment
inv: C4
self.operator = ControlFlowSegment::Loop or self.operator =
ControlFlowSegment::Exception or self.operator =
ControlFlowSegment::Stop implies self.path->size() = 1
```

INTERACTION PATH (CAMINO DE INTERACCIÓN)

Semántica

Un camino de interacción representa un camino posible de ejecución en un segmento de control. Un camino de interacción siempre está contenido en un segmento de control. Opcionalmente, un camino de interacción puede tener una condición, la cual debe ser evaluada en *True* para que el camino se ejecute. No obstante, la ejecución de un camino también depende del operador de control definido en el segmento de control al que pertenece.

Un camino de interacción, al igual que un protocolo, está compuesto de un conjunto ordenado de elementos. Esto representa que dentro del camino pueden ocurrir secuencias de mensajes, llamada a subprotocolos, otro segmento de control anidado, o terminaciones.

Asociaciones

- *guard:Condition [0..1]*. Referencia la expresión booleana que restringe un camino de interacción. La expresión puede ser expresada en lenguaje natural o en un lenguaje formal.
- *element:InteractionProtocolElement [0..*]*. Referencia los elementos que forman parte del camino.

CONDITION (CONDICIÓN)

Semántica

Una condición representa una expresión booleana que restringe un camino de interacción o un mensaje de negocio, es decir, define cuándo el camino o mensaje puede ser ejecutado. La expresión puede ser expresada en lenguaje natural o en un lenguaje formal.

Atributos

- *conditionExpression*. Indica la expresión de la condición.

TIME EXPRESSION (EXPRESIÓN DE TIEMPO)

Semántica

Una expresión de tiempo está asociada a un elemento del protocolo de interacción, tal como un mensaje, un segmento de control, un subprotocolo o el protocolo mismo. La semántica concreta de la expresión está dada por sus subtipos.

Asociaciones

- *element: InteractionProtocolElement*. Referencia el elemento del protocolo al que la expresión de tiempo es aplicada.

TIME OBSERVATION (OBSERVACIÓN DE TIEMPO)

Semántica

Una observación de tiempo es una acción que retorna el valor actual del tiempo en el contexto en el cual un protocolo se está ejecutando. El valor es asignado a una variable, la cual podrá ser utilizada posteriormente en diferentes restricciones de tiempo. Por ejemplo, para capturar el tiempo actual en una variable "t", el valor puede ser "now", que representa la hora y fecha actual.

Atributos

- *variable: String.* Indica el nombre de la variable que contiene el valor observado en el tiempo.
- *value: String*. Contiene el valor observado en el tiempo.

TIME CONSTRAINT (RESTRICCIÓN DE TIEMPO)

Semántica

Una restricción de tiempo denota una duración en el tiempo. La duración está dada por la diferencia entre un punto de inicio y un punto de fin en el tiempo. La restricción de tiempo asociada a un elemento del protocolo representa el tiempo límite disponible para la ejecución de ese elemento. Para un mensaje, éste indica el tiempo a considerar para la recepción del mensaje. Para un subprotocolo, éste indica el tiempo a considerar para la finalización del mismo. Para un segmento de control, éste indica el tiempo a considerar para la finalización del segmento.

Una restricción de tiempo puede ser definida usando tiempos absolutos o tiempos relativos, es decir, que la misma se referirá al tiempo actual o a un evento específico, tal como el último mensaje enviado. Por ejemplo, una restricción de tiempo sobre un mensaje, la cual tiene como fecha de comienzo "0" y como fecha de fin "4d", expresa que el mensaje tiene que ser recibido entre el tiempo actual y cuatro días. Como ejemplo de tiempos relativos, una restricción sobre un mensaje, con la fecha de comienzo "t" y la fecha de fin "t+4d" expresa que el mensaje tiene que ser recibido entre el tiempo de observación "t" y el tiempo de observación más cuatro días.

Atributos

- *startTime.* Indica el comienzo en el tiempo de la duración que representa la restricción de tiempo.
- *endTime*. Indica el fin en el tiempo de la duración que representa la restricción de tiempo.

TERMINATION (TERMINACIÓN)

Semántica

Representa la finalización del protocolo. La semántica concreta de la terminación está dada por el atributo *endState*, el cual puede tomar dos valores: "Success" o "Failure". En el caso que el valor de éste atributo sea "Success", significa que el protocolo ha finalizado con éxito. En caso contrario, significa que el protocolo ha finalizado sin éxito o con una falla. El éxito o falla de un protocolo es sólo una indicación lógica de la terminación del protocolo, la cual puede ser utilizada posteriormente para iniciar otros protocolos. Una terminación con falla del protocolo no significa que todos los documentos de negocio y actos de comunicación intercambiados deban descartarse, sino que sólo es una indicación de que el protocolo no ha seguido la lógica de negocio deseada.

Un protocolo siempre tiene una terminación exitosa implícita, como consecuencia de la ejecución del último mensaje, el cual también puede estar definido dentro de un segmento de control. No obstante, el uso de terminaciones hace posible indicar la finalización del protocolo como consecuencia de la ejecución de un determinado camino de interacción dentro de un segmento de control, luego del cual no se requiere la continuación del protocolo.

Atributos

- *endState: {Success | Failure}*. Indica si la terminación de un protocolo es exitosa o no.

PROTOCOL TEMPLATE (PLANTILLA DE PROTOCOLO)

Semántica

Una plantilla de protocolo representa un patrón reutilizable de interacciones utilizado para definir otros protocolos de interacción. Una plantilla de protocolo no es

un protocolo que pueda ser usado directamente, debido a que éste es un protocolo parametrizable, en donde los roles y el contenido de los mensajes constituyen los parámetros del mismo. En una plantilla de protocolo, los roles y el contenido de los mensajes deben ser ficticios, es decir, no deben referirse a los roles y documentos de negocio definidos en una colaboración B2B.

Una plantilla de protocolo sirve precisamente como un patrón estándar que puede ser personalizado para crear protocolos de interacción que soporten varios procesos colaborativos. Para ello, los roles y documentos de negocio ficticios deben corresponderse con los definidos en una colaboración B2B. Esto es indicado asignando valor a los parámetros de la plantilla en tiempo de diseño, de acuerdo a los roles y documentos definidos en la colaboración B2B.

Debido a que es una plantilla es un tipo específico de protocolo, la semántica adicional de la misma es idéntica a la semántica de un protocolo de interacción.

Restricciones

[1] Una plantilla de protocolo no puede estar asociada a un proceso colaborativo, debido a que la misma no representa un protocolo de interacción específico.

```
context ProtocolTemplate
inv: C9
self.realizes->isEmpty()
```

[2] Los roles definidos en una plantilla deben ser diferentes.

```
context ProtocolTemplate
inv: C2
self.role->forAll(r1,r2:PartnerRole | r1 <> r2)
```

PROTOCOL PARAMETER (PARÁMETRO DEL PROTOCOLO)

Semántica

Un parámetro del protocolo es utilizado para definir un protocolo de interacción en base a una plantilla de protocolo. Los parámetros asociados a la plantilla se corresponden con los roles y/o documentos de negocio ficticios definidos en la plantilla. El valor de los parámetros indica los roles o documentos de negocio definidos en una colaboración B2B.

Atributos

- *name: String*. Indica el nombre del parámetro.
- *value: String*. Indica el nombre de un rol o documento de negocio de una colaboración B2B asignado al parámetro.

Asociaciones

- *protocol: ProtocolTemplate*. Referencia la plantilla de protocolo en la cual éste es un parámetro.

4.2.1.4. El paquete Business Interfaces

Este paquete especifica los elementos conceptuales para la definición de la Vista de las Interfaces de Negocio de los Socios, la cual define las interfaces asociadas a cada rol que los socios deben implementar. La Figura 4-7 muestra la sintaxis abstracta del metamodelo descrita por este paquete. Los elementos conceptuales provistos por el mismo se describen a continuación.

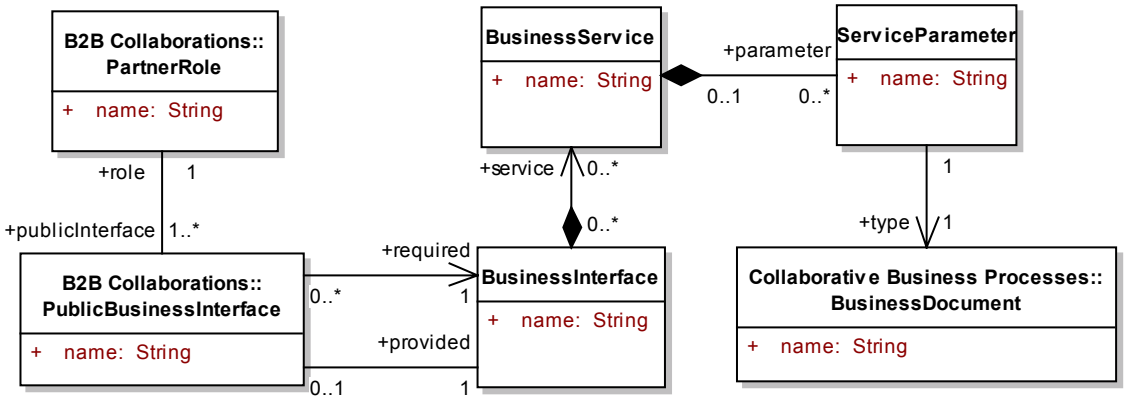


Figura 4-7. Sintaxis abstracta del paquete Business Interfaces

Las anteriores vistas dan soporte al análisis y diseño de los procesos colaborativos. No obstante, es necesario definir las interfaces de negocio de los roles, las cuales proveen los servicios requeridos para soportar el intercambio de mensajes en los protocolos de interacción. Las interfaces con sus servicios son derivadas a partir de las definiciones de los protocolos de interacción.

BUSINESS INTERFACE (INTEFAZ DE NEGOCIO)

Semántica

Una interfaz de negocio forma parte de la interfaz de negocio pública que el rol expone para interactuar con otro rol. El socio que desempeña un rol debe implementar sus interfaces de negocio para poder ejecutar los protocolos de interacción.

Una interfaz de negocio está compuesta por servicios, con el objetivo de dar soporte a las interacciones entre dos roles definidas en los protocolos de interacción. Una interfaz de negocio siempre esta asociada a dos roles, en donde en un rol es definida como su interfaz provista y en el otro es definida como su interfaz requerida. De esta manera se provee una vista estática de los protocolos de interacción mirada desde el punto de vista del rol que tiene asociada la interfaz de negocio. Cuando ésta es una interfaz provista en un rol, la interfaz dará una vista de los mensajes de negocio que el rol puede recibir en los protocolos en los cuales interactúa con el otro rol. Cuando ésta es una interfaz requerida, la interfaz dará una vista de los mensajes de negocio que el rol puede enviar desde otro rol.

Las interfaces de negocio provistas y requeridas entre dos roles son derivadas a partir de los protocolos de interacción en los cuales los roles están involucrados. Los servicios de la interfaz son definidos a partir de los mensajes de negocio que el rol puede enviar o recibir en los protocolos en los cuales está involucrado. Aunque la interfaz de los roles podría ser definida directamente con la tecnología de implementación a usar, la representación conceptual de la misma permite a los socios tener una idea cabal de lo que deben proveer y lo que requieren de los otros socios.

El concepto de interfaz de negocio es similar al concepto de interfaces provisto por UML para el modelado conceptual orientado a objetos o a componentes.

Atributos

- *name: String*. El nombre dado a la interfaz.

Asociaciones

- *service: BusinessService [0..*]*. Referencia el conjunto de servicios que contiene la interfaz de negocio.

BUSINESS SERVICE (SERVICIO DE NEGOCIO)

Semántica

Un servicio de negocio maneja la recepción de mensajes de negocio que son enviados por un rol a partir de lo definido en un protocolo de interacción, como así también es el encargado de responder los acuses de recibo que requiera dicho mensaje.

Un servicio de una interfaz puede ser reutilizado para dar soporte a diferentes protocolos los cuales contienen el mensaje que maneja el servicio.

Atributos

- *name: String*. Indica el nombre del servicio.

Asociaciones

- *parameter: ServiceParameter [0..*]*. Referencia los parámetros del servicio.

SERVICE PARAMETER (PARÁMETRO DE UN SERVICIO)

Semántica

Un parámetro de un servicio es una especificación de un argumento utilizado para pasar la información asociada al mensaje a ser recibido y manejado por el servicio. La información hace referencia al documento de negocio trasportado por el mensaje de negocio a ser procesado por el servicio.

Atributos

- *name: String*. Indica el nombre del parámetro.

Asociaciones

- *type: BusinessDocument[1]*. Referencia el tipo de documento de negocio del parámetro.

4.2.2.. Definición del Lenguaje UP-ColBPIP como un Perfil UML

El *Perfil UML de Procesos de Negocio Colaborativos basados en Protocolos de Interacción (UP-ColBPIP)* especifica cómo los elementos conceptuales definidos en el metamodelo de UP-ColBPIP descrito (Sección 4.2.1) se relacionan y son representados en UML a través de estereotipos, valores etiquetados y restricciones. Cada clase del metamodelo es definida como un estereotipo en el perfil o bien se establece una correspondencia con una metaclase UML. Los atributos de las clases del metamodelo

son expresados como valores etiquetados en los estereotipos correspondientes. Las restricciones son definidas a partir de las restricciones del metamodelo. En algunos casos también se agregan restricciones para representar las multiplicidades de las asociaciones que aparecen en el metamodelo. UP-ColBPIP está basado en el metamodelo de UML2 (OMG, 2003b).

UP-ColBPIP (Villarreal y otros, 2004b) ha sido definido con el objetivo de proveer la mejor correspondencia posible con UML, de tal manera que la notación sea natural para un diseñador de procesos colaborativos que conoce UML. Esto significa que la semántica de los elementos de UML que se han extendido se asemeja o se corresponde de alguna manera con la semántica de los conceptos provistos en el metamodelo. Este principio de diseño permite que los usuarios del perfil puedan capturar en forma más rápida la semántica de los conceptos provistos por UP-ColBPIP.

UP-ColBPIP está organizado en cuatro paquetes que corresponden a los paquetes definidos en el metamodelo, los cuales proveen los elementos del perfil para modelar las vistas requeridas en el diseño de procesos colaborativos (Figura 4-8). El perfil es definido en un paquete UML estereotipado <<*profile*>>. La notación utilizada es la siguiente. Dentro de cada paquete, los estereotipos se definen con la notación de clase, con el agregado de la expresión <<*stereotype*>> antes o después de su nombre. Las metaclases UML que están siendo extendidas por los estereotipos también se definen con la notación de clase, con el agregado de la expresión <<*metaclass*>>. La notación de la extensión es una flecha apuntando desde el estereotipo hacia la metaclase, en donde la punta de la flecha es un triángulo relleno. Los *valores etiquetados* son representados como atributos de cada clase que define el estereotipo.

A continuación se describen cada uno de los paquetes de UP-ColBPIP.

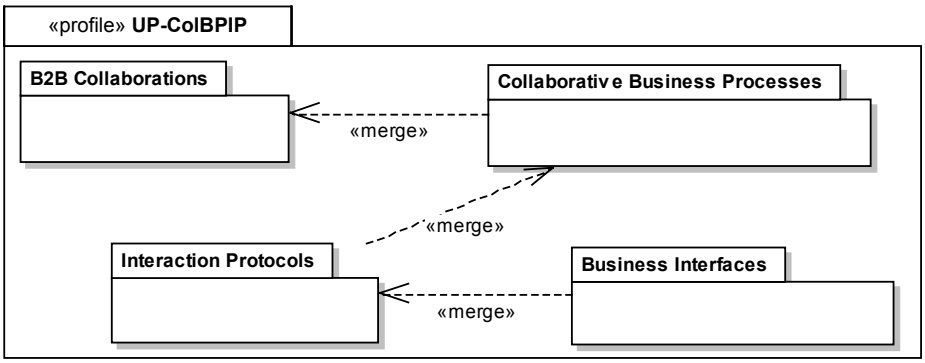


Figura 4-8. Paquetes del perfil UML UP-ColBPIP

4.2.2.1. El paquete B2B Collaborations

Este paquete especifica los elementos conceptuales para la definición de la *Vista de la Colaboración B2B*. Este paquete extiende la semántica de las colaboraciones de UML como así también la semántica de clases de UML, con el objetivo de modelar las colaboraciones B2B y los requerimientos y metas del acuerdo de colaboración. La Figura 4-9 muestra los estereotipos de este paquete junto con las metaclases de UML que están extendiendo y sus valores etiquetados correspondientes.

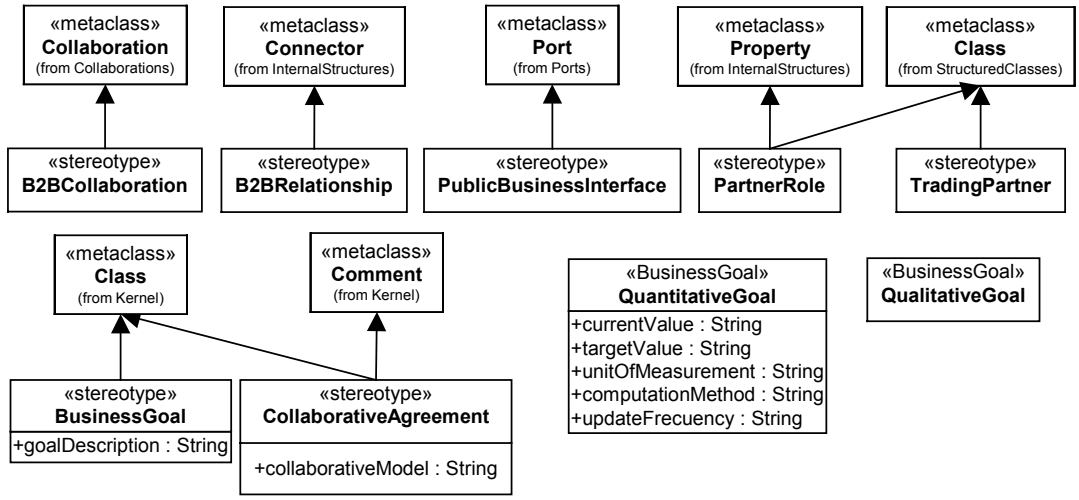


Figura 4-9. Paquete *B2B Collaborations* de UP-ColBPIP

El concepto de *colaboración* de UML es utilizado para representar un conjunto de instancias cooperando entre sí, las cuales son especificadas por clasificadores. Dichas instancias cooperan enviando señales o invocando operaciones. Este concepto es similar al concepto de colaboración B2B en el metamodelo. Dicho concepto representa un conjunto de roles que cooperan entre sí, los cuales son desempeñados por socios de negocio. Los roles cooperan enviando mensajes que representan las interacciones dentro de un proceso colaborativo. De esta manera, es posible extender los constructores de UML que definen colaboraciones para representar colaboraciones B2B.

En UML, las clases estructuradas son utilizadas para representar que un clasificador puede contener una estructura interna y puertos en donde se definen las interfaces requeridas y provistas por dicho clasificador. Por lo tanto, los estereotipos *TradingPartner* y *PartnerRole* extienden una clase estructurada de UML para indicar que el primero puede contener una estructura interna compuesta de procesos privados y el segundo puede contener un comportamiento público y visible hacia los otros roles.

Además, en UML la metaclase *Property* representa una parte o participante en una colaboración. Por lo tanto, el estereotipo *PartnerRole* extiende la metaclase *Property* para representar los participantes en una colaboración B2B.

Una clase estructurada de UML puede expresar su comportamiento público a través de puertos, los cuales poseen las interfaces requeridas y provistas del clasificador al que está asociado. En el metamodelo, esto es representado por el concepto *PublicBusinessInterface* asociado a los roles. Por lo tanto, el estereotipo *PublicBusinessInterface* extiende la metaclase *Port*. Los puertos en UML están vinculados a través de conectores, por lo tanto el estereotipo *B2BRelationship* extiende la metaclass *Connector*. A través de la extensión de la semántica de las colaboraciones de UML para representar colaboraciones B2B en UP-ColBPIP, es posible definir estas últimas usando los Diagramas de Estructura Compuesta de UML.

La notación para una colaboración B2B es la misma que para una colaboración de UML, es decir, una elipse estereotipada <<B2Bcollaboration>>. Cada participante de la colaboración B2B es indicado usando la notación de parte de UML, en donde se indica primero el nombre del socio seguido por el nombre del rol que el socio desempeña. Antes de asociar los socios de negocio a los roles y definir la participación de los mismos en una colaboración B2B, ambos deben ser definidos.

La Figura 4-10, muestra un diagrama de estructuras compuestas que describe la colaboración B2B “Planificación Jerárquica Colaborativa” a llevar a cabo entre la empresa “Imperio Autopartes” y su socio de negocio en la colaboración, la empresa “WW Automotores”. La primera juega el rol de “Proveedor” y la segunda el rol de “Distribuidor”. La relación entre ambos roles es indicada por las interfaces de negocio públicas que cada uno provee y por el conector *B2BRelationship* que las vincula. Las interfaces de negocio públicas son mostradas con la notación de puertos. El conector *B2BRelationship* es indicado usando la notación de asociaciones de UML. Un comentario con el estereotipo <<CollaborativeAgreement>> indica el nombre del acuerdo de colaboración que gobierna la colaboración B2B.

Los estereotipos *CollaborativeAgreement* y *BusinessGoal* extienden la semántica de una clase de UML. Por lo tanto, los atributos de un acuerdo de colaboración junto con las metas de negocio son definidos en Diagramas de Clases. Sólo la meta de negocio es definida como un estereotipo. Los subtipos de metas no son definidos como

estereotipos sino que son provistos en el perfil directamente como clases adicionales estereotipadas <<*BusinessGoal*>>, las cuales representan un tipo de meta. De esta manera es posible definir las metas de negocio cuantitativas y cualitativas como instancias de metas de negocios en un diagrama de clase. Esto permite que los diseñadores puedan definir no sólo metas cuantitativas y cualitativas, sino que también puedan definir otros tipos de metas o bien especializar las anteriores. Las relaciones de dependencia entre las mismas se representan usando el concepto de asociación de dependencia de UML.

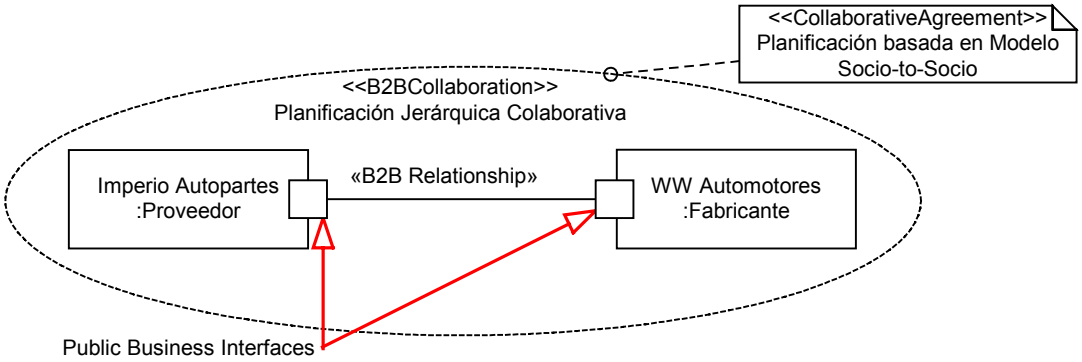


Figura 4-10. Diagrama de estructuras compuestas que describe una Colaboración B2B

En un diagrama de clases, un acuerdo de colaboración es mostrado con la notación de clase estereotipada <<CollaborativeAgreement>>. Los atributos del acuerdo son indicados usando la notación de atributos de clase de UML. Las metas de negocio asociadas al acuerdo son indicadas usando la notación de instancia de una clase estereotipada <<BusinessGoal>>, en donde la clase de la instancia representa el tipo de meta de negocio, por ejemplo si es una meta cuantitativa o cualitativa. La meta principal está asociada a un acuerdo usando la notación de asociaciones de UML. Las dependencias entre las metas son mostradas usando la notación de asociaciones de dependencias de UML.

La Figura 4-11, muestra el diagrama de clases que define el acuerdo de colaboración "Planificación basada en Modelo P2P". En éste se define que el modelo de colaboración a utilizar es el *Modelo de Colaboración Socio-a-Socio* (Villarreal y otros, 2003a). El acuerdo posee atributos que definen: la fecha de comienzo y fin de la colaboración, los productos sobre los que se va a colaborar (en este caso se colabora sobre un único producto), y los períodos de congelamiento para los planes maestros de la producción (PMP) y los planes de aprovisionamiento (MRP), dentro de los cuales un

plan no debería ser modificado salvo excepciones por imprevistos. El acuerdo tiene asociado una meta de negocio cualitativa. Dicha meta tiene una dependencia con tres submetas, las cuales deben ser alcanzadas para cumplir con la meta padre. Las tres submetas son cuantitativas y estas han sido definidas en base a indicadores de rendimientos claves o medidas de rendimientos. Por ejemplo, la meta "Precisión en la Planificación Agregada" expresa el deseo de definir planes agregados de producción los más precisos posibles, en el sentido que no requieran de demasiados cambios durante el período de congelamiento. Dicha meta define que el método de cálculo es el número de cambios en un plan agregado durante el período de congelamiento definido en el acuerdo, y por lo tanto, la unidad de medida es número de cambios. La frecuencia de actualización o de evaluación de la meta es de dos meses. El valor actual indica que actualmente se realizan cinco cambios en promedio cada dos meses y el valor a alcanzar indica que se desea que el número de cambios no sea mayor a dos.

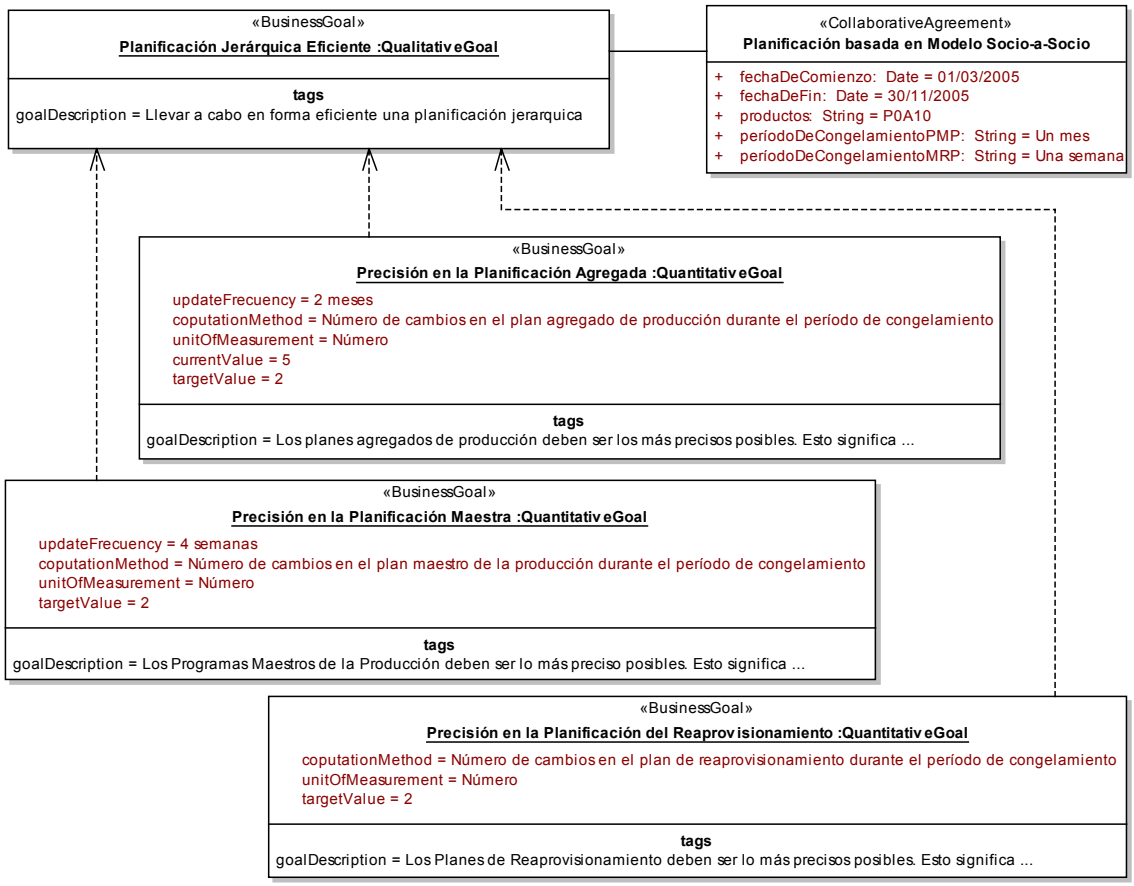


Figura 4-11. Diagrama de clases del acuerdo de colaboración y las metas de negocio

4.2.2.2. El Paquete Collaborative Business Processes

Este paquete especifica los elementos conceptuales para la definición de la *Vista de Procesos de Negocio Colaborativos*. Este paquete extiende la semántica de los casos de uso de UML para identificar y definir de modo informal los procesos de negocio colaborativos a llevar a cabo en una colaboración B2B. La Figura 4-12 muestra los estereotipos de este paquete junto con las metaclases de UML que están extendiendo y sus valores etiquetados.

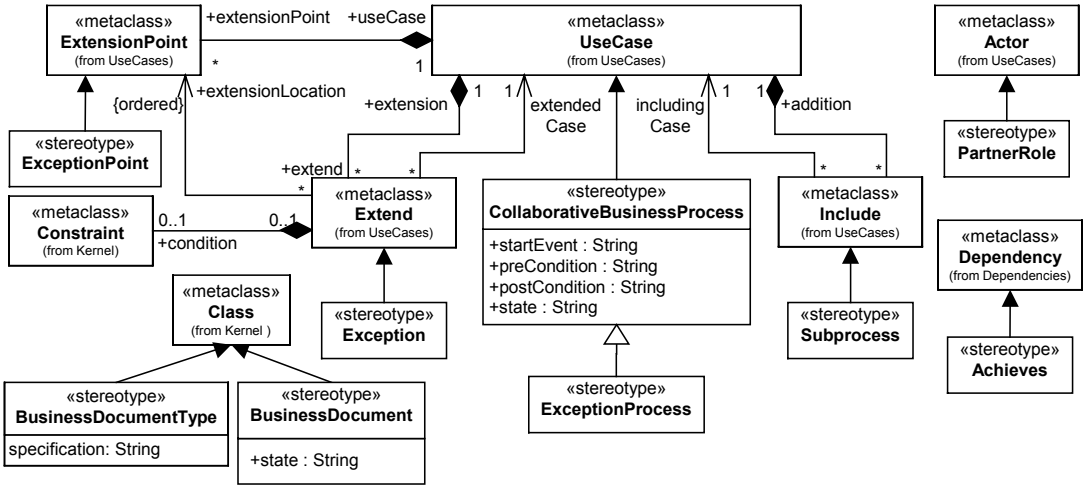


Figura 4-12. Paquete *Collaborative Business Processes* de UP-ColBPIP

En UML, el concepto de *caso de uso* es utilizado para capturar los requerimientos de un sistema, es decir, lo que se supone que el sistema va a realizar. En UP-ColBPIP, la semántica del *caso de uso* es extendida para representar un proceso colaborativo, en el cual se captura qué es lo que se supone que los socios realizarán como parte de una colaboración B2B. Por lo tanto, UP-ColBPIP extiende los constructores de UML que permiten definir casos de uso para posibilitar la definición de procesos colaborativos. De esta manera, los procesos colaborativos son definidos usando *diagramas de casos de uso de UML.*

En un diagrama de casos de uso, un proceso colaborativo es mostrado con una elipse estereotipada <<CollaborativeBusinessProcess>>. La Figura 4-13 muestra un diagrama de casos de uso que describe cuatro procesos colaborativos.

En este paquete, el estereotipo *PartnerRole* extiende la metaclase *actor* para representar los roles involucrados en un proceso colaborativo. El rol es asociado a un proceso a través de una relación de asociación. En un diagrama de casos de uso, un rol

que toma parte de un proceso colaborativo es indicado con la misma notación de actor de UML, con el nombre del socio seguido del nombre del rol que éste desempeña en el proceso. Por ejemplo, la Figura 4-13 muestra que la empresa "Imperio Autopares" desempeña el rol de proveedor y la empresa "WW Automotores" desempeña el rol de fabricante en el proceso "Planificación Jerárquica de la Producción", como así también en los subprocesos de éste.

El estereotipo *Achieves* extiende la asociación de dependencia de UML para representar la relación de asociación establecida en el metamodelo entre una meta de negocio y el proceso colaborativo que pretende cumplir con dicha meta. La relación de asociación es indicada a través de la notación de asociación de dependencia de UML estereotipada <<Achieves>>. Como ejemplo, en la Figura 4-13 el proceso "Planificación Jerárquica de la Producción" tiene definido que la meta a alcanzar es "Planificación Jerárquica Eficiente".

La metaclase *Include* de UML representa la inclusión, en el comportamiento de un caso de uso, del comportamiento definido en otro caso de uso. Por lo tanto, el estereotipo *Subprocess* extiende la metaclase *Include* para expresar que el comportamiento de un subproceso está incluido en el comportamiento del proceso que lo invoca. Este estereotipo representa el atributo de asociación *subprocess* del elemento conceptual *CollaborativeBusinessProcess* del metamodelo.

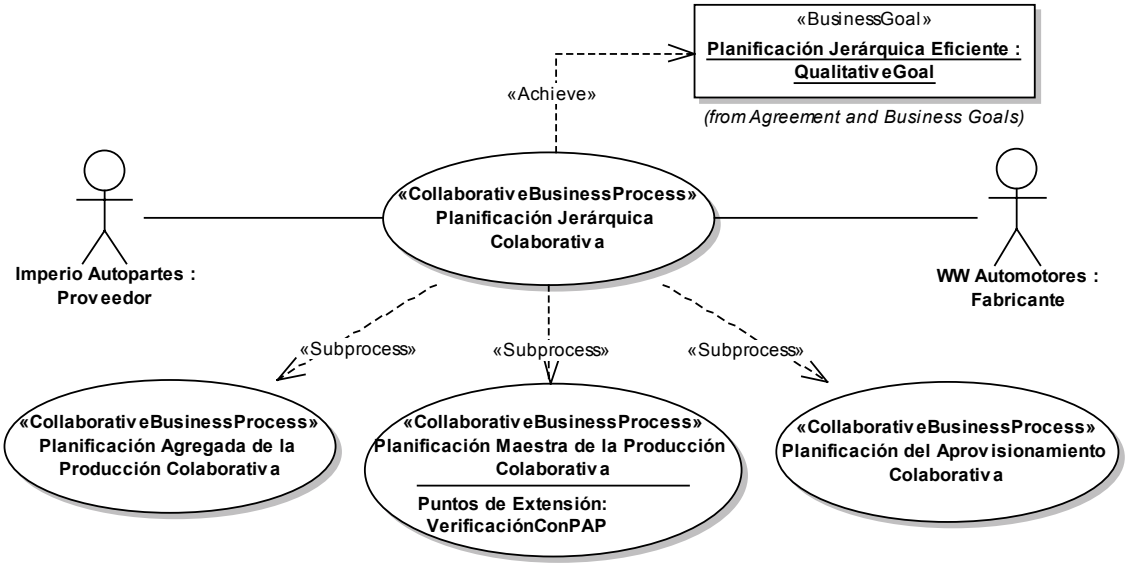


Figura 4-13. Diagrama de casos de uso describiendo procesos colaborativos

Una relación de subproceso es indicada usando la notación de asociaciones entre casos de uso estereotipada <<Subprocess>>, en donde la asociación es dirigida desde el proceso padre al subproceso. Por ejemplo, la Figura 4-13 describe que el proceso "Planificación Jerárquica de la Producción" posee tres subprocesos.

El estereotipo *Exception* representa el atributo de asociación *exception* del elemento conceptual *CollaborativeBusinessProcess* del metamodelo. Por lo tanto, este estereotipo es utilizado para establecer una relación entre un proceso colaborativo y un proceso de excepción, en donde este último manejará alguna excepción definida en el primero. El estereotipo *Exception* extiende la metaclase *Extend*, debido a que en UML, la misma representa cuándo y cómo el comportamiento definido en un caso de uso puede ser insertado en el comportamiento definido en un caso de uso extendido. Por lo tanto, el estereotipo *Exception* es utilizado para indicar cuándo y cómo el comportamiento de un proceso de excepción es incluido en el comportamiento de un proceso colaborativo para manejar una excepción en este último. La asociación de excepción entre dos procesos es indicada usando la notación de asociaciones entre casos de uso estereotipada <<Exception>>.

El estereotipo *ExceptionPoint* extiende la semántica de la metaclase *ExtensionPoint*, debido a que en UML esta última permite definir el lugar en donde un caso de uso puede ser extendido por el comportamiento de otro. En UP-ColBPIP, un *ExceptionPoint* no sólo representa el lugar donde una excepción puede ocurrir sino también la condición de excepción a ser evaluada para determinar la ocurrencia de la excepción. Esta condición es definida en el atributo *condition* del *ExceptionPoint* del metamodelo. En el perfil esta condición es representada por la metaclase *Constraint* asociada a la metaclase *Extend*.

Un punto de excepción es definido dentro de la elipse que representa a un proceso colaborativo usando la siguiente sintaxis: <puntos de extensión> ::= <nombre del punto de excepción> [:<explicación>]. La condición de excepción es indicada en una nota asociada a la relación de excepción entre un proceso colaborativo y un proceso de excepción.

Como ejemplo de la definición de procesos de excepciones, la Figura 4-14 muestra el proceso de excepción "Resolver Excepción en PMP Colaborativo", el cual está asociado a través del estereotipo *Exception* al proceso colaborativo "Planificación

Maestra de la Producción Colaborativa". Esta asociación indica que el primero será el encargado de gestionar la excepción que puede ocurrir en el punto de excepción "VerficiaciónConPAP" del proceso colaborativo. La condición de excepción a evaluar como así también el punto de excepción relacionado con dicha excepción son visualizados con una nota asociada a la relación de excepción. El proceso de excepción se activará en el caso que el proceso colaborativo se encuentre en el punto de excepción "VerficiaciónConPAP" y la condición de excepción sea verdadera, es decir, que la diferencia entre el plan agregado de producción y el plan maestro de la producción sea mayor a la tolerancia de cambio acordada entre los socios.

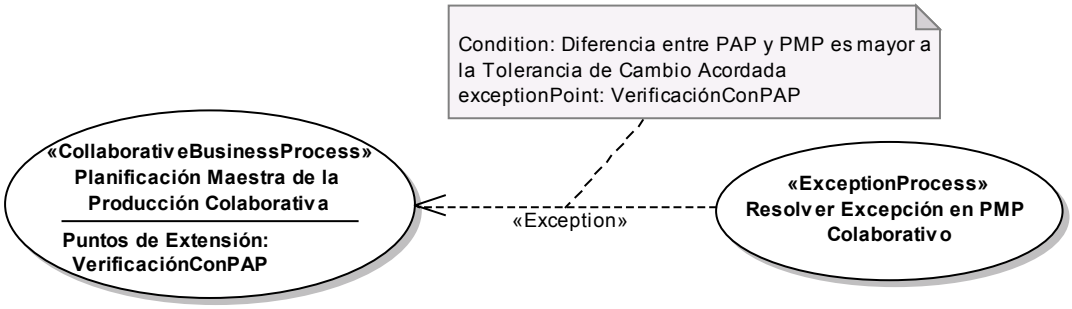


Figura 4-14. Ejemplo de proceso de excepción

Al igual que un caso de uso, un proceso colaborativo también puede ser indicado en un diagrama de clases usando una notación de clase con una elipse en la esquina superior derecha, como lo indica la Figura 4-15. Esta última notación posibilita la visualización de los atributos de un proceso, como ser el evento de inicio, las precondiciones y poscondiciones, y los documentos de negocio que serán intercambiados entre los socios en la ejecución del proceso.

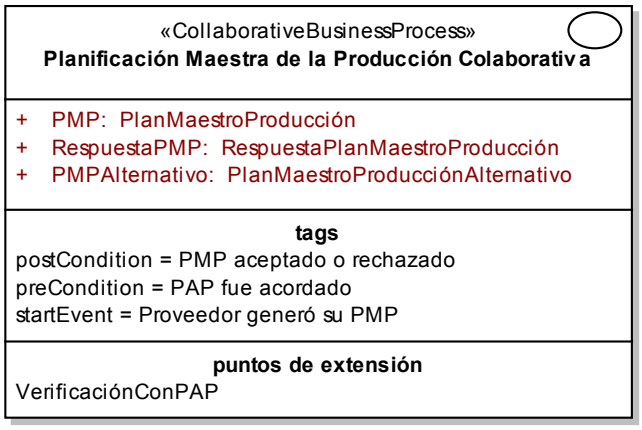


Figura 4-15. Ejemplo de un proceso colaborativo con notación de clase

Los estereotipos *BusinessDocument* y *BusinessDocumentoType* extienden la metaclase *Class*. Tanto los documentos de negocio como los tipos de documentos de negocio son definidos en diagramas de clase. La relación entre el documento de negocio y su tipo es indicada a través de una relación de asociación. Por ejemplo, la Figura 4-16 muestra tres documentos de negocio definidos con sus respectivos tipos de documentos.

El estereotipo *BusinessDocument* además extiende la metaclase *property*, para representar los documentos de negocio que son intercambiados en un proceso colaborativo. De esta manera, en un proceso colaborativo, los documentos de negocio son definidos como atributos del proceso, cuyo tipo es uno de los tipos de documentos definidos. Por ejemplo, la Figura 4-15 muestra que el proceso "Planificación Maestra de la Producción Colaborativa" tiene definido tres documentos de negocio, con sus respectivos tipos.

Un documento de negocio es indicado como una clase estereotipada <<*BusinessDocument*>>. Un documento de negocio es definido como parte de una colaboración B2B y puede ser reusado en diferentes procesos de negocio. Por lo tanto, todos los documentos de negocio que pueden ser usados en la colaboración se definen en un diagrama de clases. En los procesos colaborativos se indica como un atributo el nombre del documento a usar en el proceso, cuyo tipo es uno de los documentos definidos en la colaboración B2B. El nombre de ambos debe ser el mismo. Por ejemplo, la Figura 4-16 muestra tres documentos de negocio utilizados en el proceso colaborativo "Planificación Maestra de la Producción Colaborativa".

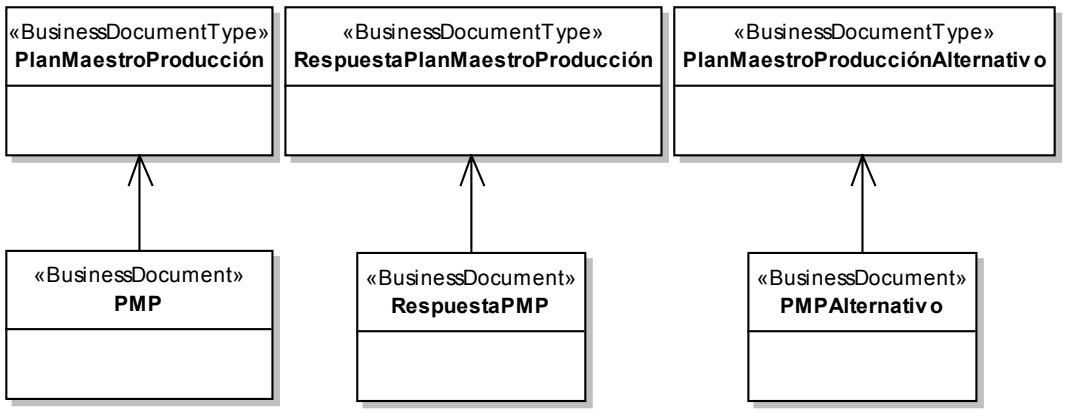


Figura 4-16. Ejemplo de documentos de negocio

4.2.2.3. El Paquete Interaction Protocols

Este paquete especifica los elementos conceptuales para la definición de la *Vista de Protocolos de Interacción*. Este paquete extiende la semántica de las interacciones de UML con el objetivo de modelar protocolos de interacción que definan el comportamiento de los procesos de negocio colaborativos. La Figura 4-17 muestra los estereotipos de este paquete junto con las metaclases de UML que extienden y sus valores etiquetados.

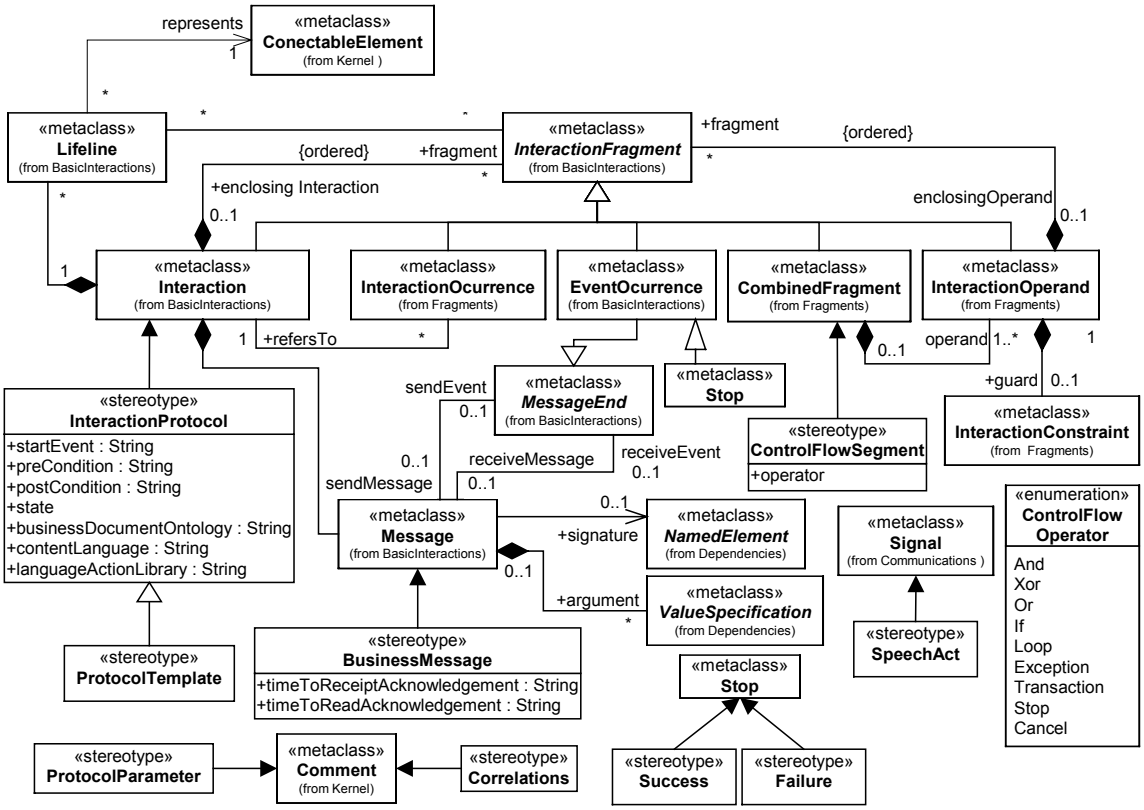


Figura 4-17. El Paquete *Interaction Protocols* de UP-ColBPIP

En UML, el concepto de *interacción* es utilizado para describir una unidad de comportamiento que se centra en el intercambio de mensajes entre elementos que están conectados, tales como actores de casos de uso, clasificadores, instancias de clasificadores, etc. En UP-ColBPIP, la semántica de las interacciones es extendida para definir protocolos de interacción que describen el comportamiento de un proceso colaborativo a través de una coreografía de mensajes, en donde los mensajes son intercambiados entre los roles que los socios de negocio desempeñan en un proceso. Con este propósito, el estereotipo *InteractionProtocol* extiende la metaclase *Interaction*.

Las interacciones en UML pueden ser visualizadas usando los diferentes tipos de *diagramas de interacciones*: *diagramas de secuencia*, *diagramas de comunicaciones*, *diagramas global de interacciones* (Overview) y *diagramas de tiempo*. Cada tipo de diagrama permite la visualización de diferentes vistas de las interacciones UML. Para definir un protocolo de interacción se utilizará principalmente el diagrama de secuencia, dado que es el más expresivo y permite visualizar en un solo diagrama la mayoría de las propiedades de los protocolos. No obstante, opcionalmente también es posible visualizar un protocolo a través de diagramas globales de interacciones y diagramas de tiempo.

En un diagrama de secuencia, la notación utilizada para un protocolo de interacción es la misma que la utilizada para las interacciones UML, en donde el rectángulo del diagrama de secuencia es estereotipado <<protocol>>. Como ejemplo, la Figura 4-18 muestra el diagrama de secuencia del protocolo *Solicitud Pronóstico de Ventas*. Opcionalmente, los atributos del protocolo, tales como la librería de actos de comunicación (Anexo A) o el lenguaje de contenido podrían ser definidos con un comentario asociado al protocolo, con el estereotipo <<ProtocolProperties>>.

Una interacción de UML está compuesta de un conjunto ordenado de *InteractionFragments*. Un *InteractionFragment* representa un elemento de una interacción. Por lo tanto, el concepto *InteractionProtocolElement* tiene la misma semántica que un *InteractionFragment* y de este manera este concepto no es adicionado como un estereotipo en el perfil, sino que se corresponde con el concepto de *InteractionFragment* provisto por UML.

En las interacciones de UML, los mensajes son intercambiados entre elementos que pueden ser conectados (ej: clasificadores, instancias de clasificadores, actores de casos de uso, etc.), los cuales son indicados por las *lifelines* que posee la interacción. Una *lifeline* representa un participante en la interacción, ya que referencia a un elemento que puede ser conectado. En UP-ColBPIP, *lifelines* permiten representar los roles involucrados en un protocolo de interacción. Una *lifeline* sólo puede hacer referencia a una clase o actor con el estereotipo *PartnerRole*.

En un diagrama de secuencia (Figura 4.18), la notación del estereotipo *PartnerRole* junto con el socio de negocio que desempeña el rol es indicada de la misma manera que los elementos que pueden ser conectados a una *lifeline* en UML. Es decir, un *PartnerRole* es mostrado por dos elementos: un rectángulo en la parte superior del

diagrama con el nombre de la *lifeline* que sigue la siguiente sintaxis: <nombre del socio de negocio>: <nombre del rol>; y una línea vertical de puntos unida al rectángulo. El nombre del socio es opcional, ya que un protocolo puede definir roles sin determinar los socios que cumplen dichos roles. Sobre la línea vertical de la *lifeline* se indican los mensajes que un rol puede enviar o recibir. De esta manera es posible visualizar las responsabilidades de los roles en los protocolos, es decir, los mensajes que pueden enviar o recibir. Además, a través de las *lifelines* es posible visualizar el orden en el tiempo de los elementos del protocolo de interacción. Como ejemplo, la Figura 4-18 muestra dos *lifelines* que describen a los roles "proveedor" y "minorista", los cuales son desempeñados respectivamente por el "socio A" y el "socio B".

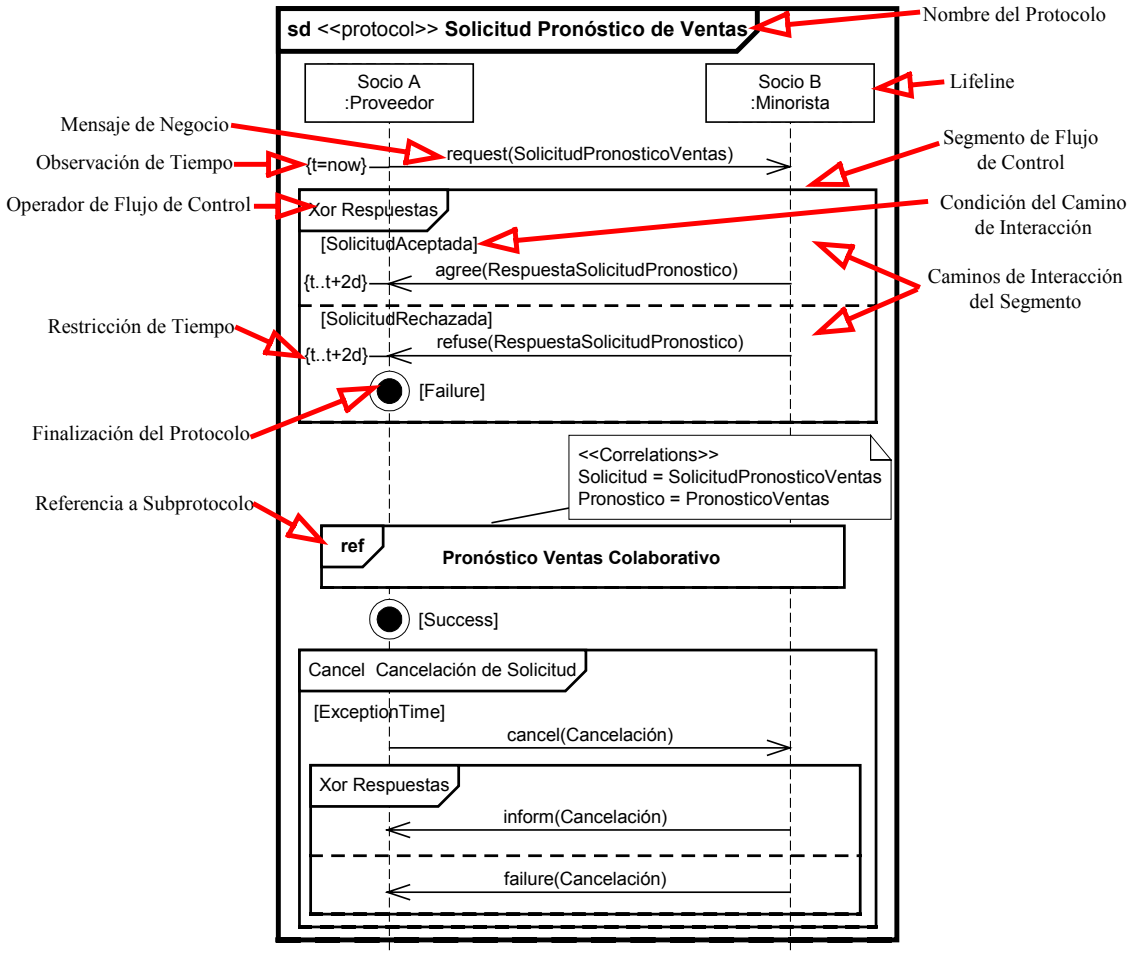


Figura 4-18. Ejemplo de protocolo de interacción

Continuando con la descripción del paquete Interaction Protocol (Figura 4-17) se debe considerar que en las interacciones de UML, una comunicación entre *lifelines* es definida a través de mensajes. Por lo tanto, el estereotipo *BusinessMessage* extiende la metaclase *Message* para representar la comunicación entre dos roles. Además, en UML,

el atributo de asociación *signature* de un mensaje puede referirse a una operación o a una señal. En el primer caso, el mensaje representa una llamada a una operación de un clasificador. En el segundo caso, un mensaje representa el envío y recepción de una señal, la cual dispara una reacción en el receptor en una manera asincrónica. No obstante, en el dominio de modelado de procesos colaborativos, los socios no pueden invocar comportamientos privados de otros socios a través de operaciones. Ellos sólo deberían poder gestionar las recepciones de señales y reaccionar ante ellas a través de sus procesos de decisiones privados. Por lo tanto, para indicar que un mensaje de negocio representa un acto de comunicación que indica la intención del remitente del mensaje, en donde el receptor del acto debe reaccionar de acuerdo a la semántica del mismo, el estereotipo *SpeechAct* extiende la metaclase *Signal*. De esta manera, en UP-ColBPIP el atributo de asociación *signature* de un mensaje de negocio sólo debe referirse a una señal estereotipada <<SpeechAct>> y no a una operación.

El documento de negocio asociado a un mensaje de negocio es representado por el atributo de asociación *argument* del mensaje, el cual debe hacer referencia a uno de los documentos de negocio definidos en el proceso colaborativo que el protocolo está materializando.

Un mensaje de negocio es indicado usando la notación de mensajes de UML en diagramas de secuencias. Un mensaje de negocio es mostrado con una línea dirigida "→" desde un rol remitente a un rol receptor. La sintaxis del nombre del mensaje es la siguiente: <nombre del acto de comunicación> (<nombre del documento de negocio >). Por ejemplo, en la Figura 4-18, el mensaje *request(SolicitudPronosticoVentas)* indica que *request* es el acto de comunicación que le da un significado al mensaje y *SolicitudPronosticoVentas* es el documento de negocio que es enviado con el mensaje.

Una condición que restringe el envío del mensaje es indicada antes del nombre del mensaje con la sintaxis: [<Expresión de Condición>]: <nombre del mensaje>. La posible repetición de un mensaje se indica antes del nombre del mensaje. Un "*" indica que el mensaje puede ser enviado un número no determinado de veces. De lo contrario, el número de veces que un mensaje puede ser enviado es definido con un número natural.

Una interacción en UML contiene los mensajes utilizados en la misma pero no define explícitamente el orden de los mensajes en la coreografía de la interacción. El

ordenamiento de un mensaje de negocio es definido por un par de *EventOccurrences*, una representando el evento de envío del mensaje y otra representando el evento de recepción del mensaje. Estas son definidas en los atributos de asociación *sendEvent* y *receiveEvent* de un mensaje. Las *EventOccurrences* forman parte de los elementos ordenados que definen la coreografía del protocolo. Una *EventOccurrence* con el atributo de asociación *sendMessage* haciendo referencia a un mensaje, indica el envío de dicho mensaje. Una *EventOccurrence* con el atributo de asociación *receiveMessage* haciendo referencia a un mensaje, indica la recepción de dicho mensaje.

Además, las *EventOccurrences* en una interacción están asociados a una *lifeline*, para indicar quien envía y recibe un mensaje. De esta manera, a través de las *EventOcurrences* asociadas a una *lifeline* es posible conocer los mensajes de negocio que un rol envía o recibe. El orden de las *EventOccurrences* en las *lifelines* es significante ya que denota el orden en que los mensajes son enviados y recibidos en el protocolo.

Para establecer secuencias complejas de mensajes, en UML el concepto de *CombinedFragment* es utilizado para definir los caminos posibles de una interacción. Este concepto podría ser utilizado directamente en UP-ColBPIP, dado que la semántica del mismo se corresponde con la del concepto *ControlFlowSegment*. No obstante, algunos operadores de flujo de control de un *CombinedFragment* no son apropiados para el modelado de procesos. Por lo tanto, con el objetivo de proveer operadores de flujo de control bien conocidos, la metaclase *CombinedFragment* es extendida con el estereotipo *ControlFlowSegment* (Figura 4.17), el cual tiene un valor etiquetado que indica el operador de flujo de control a utilizar. Al igual que en el metamodelo, los posibles operadores a utilizar son provistos en la enumeración *ControlFlowOperator*.

En interacciones de UML, los caminos dentro de un *CombinedFragment* son representados por la metaclase *InteractionOperand*. La expresión condicional que determina la ejecución del camino es representada por la metaclase *InteractionConstraint*. Por lo tanto, para representar un camino de interacción en UP-ColBPIP no se adiciona ningún estereotipo. En su lugar, el concepto *InteractionPath* del metamodelo se corresponde con el concepto *InteractionOperand* de UML, dado que ambos tienen la misma semántica. El concepto *Condition* del metamodelo se corresponde con el concepto *InteractionConstraint*. El atributo de asociación *condition*

del concepto *InteractionPath* es representado por el atributo de asociación *guard* del concepto *InteractionConstraint*.

La notación de un segmento de flujo de control es un rectángulo similar al del protocolo, en donde dentro del pentágono de la esquina superior izquierda se indica el operador de flujo de control utilizado, seguido por el nombre del segmento. Los diferentes caminos de un segmento están indicados como regiones dentro del rectángulo del segmento de control, las cuales están separadas por una línea de puntos. En la Figura 4-18 se muestra un ejemplo de un segmento de flujo de control, el cual tiene definido el operador XOR y el nombre del segmento es "Respuestas". Este segmento posee dos caminos de interacción, uno representando el camino a seguir si el minorista acepta la solicitud, y el otro representando el camino a seguir en el caso que el minorista rechaza la solicitud. Las condiciones de los caminos son indicadas entre corchetes al comienzo de cada camino, antes del primer elemento del camino.

El concepto *ProtocolReference* del metamodelo tampoco es adicionado como un estereotipo en UP-ColBPIP, dado que la semántica de éste se corresponde con la semántica de la metaclase *InteractionOccurrence* de UML. En una interacción de UML, esta metaclase es utilizada para representar una llamada a otra interacción anidada. De esta manera, el concepto de *InteractionOccurrence* provisto por UML es utilizado para representar la invocación dentro de un protocolo a un subprotocolo o protocolo anidado. El estereotipo *Correlations* extiende la metaclase *Comment* para representar las correlaciones de roles y documentos entre el protocolo y el subprotocolo.

La notación de una referencia de protocolo es mostrada con un rectángulo similar al utilizado para los segmentos de control, en donde la palabra clave *ref* es colocada en el pentágono superior del rectángulo. El contenido del rectángulo contiene el nombre del protocolo de interacción que está siendo invocado. Las correlaciones son indicadas con un comentario asociado al rectángulo de la referencia de protocolo, el cual es estereotipado <<Correlations>>. Cada correlación se define con la siguiente sintaxis: <nombre del elemento del subprotocolo> = <nombre del elemento del protocolo>. En el caso que la referencia de protocolo haga referencia a una plantilla de protocolo, entonces la instanciación de los parámetros de la plantilla son indicados con un comentario estereotipado <<ProtocolParameters>> asociada a la referencia de protocolo. Como ejemplo, la Figura 4-18 muestra que el protocolo *Solicitud Pronóstico*

Ventas tiene una llamada o referencia al protocolo *Pronóstico Ventas Colaborativo*. En dicha referencia se indican dos correlaciones de documentos de negocio. Por ejemplo, la primera correlación indica que el documento *Solicitud* del subprotocolo se corresponde con el documento *SolicitudPronosticoVentas* del protocolo que contiene la referencia.

Para representar una finalización exitosa o una finalización con falla de un protocolo, dos estereotipos son adicionados en UP-ColBPIP: *Success* y *Failure*. Ambos representan la semántica de los posibles estados finales de un protocolo, definidos por el concepto *Termination* del metamodelo. En UML, la metaclase *stop* es una especialización de una *EventOccurrence* que indica la finalización de la instancia especificada por la *lifeline*. No obstante, en UP-ColBPIP, la semántica de la metaclase *stop* es extendida por los estereotipos *Success* y *Failure* para indicar una finalización exitosa o con falla del protocolo, independientemente de la *lifeline* a la que están asociados estos estereotipos.

En un diagrama de secuencia, la finalización exitosa es indicada con un círculo relleno junto con la expressión “Success”, la cual es colocada sobre una *lifeline* de un rol. De la misma manera, una finalización con falla es indicada con un círculo relleno junto con la expresión “Failure” colocada sobre una *lifeline* de un rol. La Figura 4-18 muestra que el protocolo finaliza con falla después que el minorista envió el mensaje *refuse(RespuestaSolicitudPronostico)*, cuando éste rechaza la solicitud del proveedor. En esta caso el protocolo finaliza y no se podrá continuar con los siguientes mensajes o elementos del protocolo. En el ejemplo, la llamada al subprotocolo no será realizada.

Los conceptos definidos en el metamodelo para representar expresiones de tiempo no son adicionados como estereotipos en UP-ColBPIP, dado que UML ya provee un conjunto de constructores para expresar la semántica de estas expresiones de tiempo.

Restricciones de tiempo son mostradas en un diagrama de secuencia como un intervalo con la siguiente sintaxis: *{TiempoInicial..TiempoFinal}*. Observaciones de tiempo son indicadas de la siguiente manera: *{t=Tiempo Actual}*. Las restricciones de tiempo y las observaciones de tiempo son indicadas con una línea horizontal asociada al mensaje o al elemento al cual ésta se aplica. Como ejemplo, la Figura 4-18 muestra que el primer mensaje tiene una observación de tiempo en donde la fecha y hora actual es asociada a la variable *t*. Además, los restantes mensajes tienen asociados una restricción

de tiempo que indica que los mismos deben ser realizados dentro del tiempo de la variable *t* más dos días.

La semántica del protocolo mostrado en la Figura 4-18 es la siguiente. Dicho protocolo tiene dos roles, un rol iniciador y otro receptor. Tres mensajes de negocio forman parte del protocolo. El protocolo comienza con el *iniciador* enviando el mensaje *request(SolicitudPronosticoVentas)*, el cual consiste del acto de comunicación *request* y un documento de negocio *SolicitudPronosticoVentas*. Este mensaje indica que el iniciador solicita al receptor que realice un pronóstico de ventas con respecto a uno o varios productos. El receptor puede responder con dos mensajes alternativos. Esto es definido por el segmento de control *Respuestas* que tiene el operador *XOR*. El mensaje *refuse(RespuestaSolicitudPronostico)* indica que el receptor rechaza la acción solicitada por el iniciador e informa los detalles del rechazo en el documento *RespuestaSolicitudPronostico*. El mensaje *agree(RespuestaSolicitudPronostico)* indica que el receptor acepta la solicitud de pronóstico del iniciador y luego se compromete a generar y enviar posteriormente el pronóstico de ventas solicitado a través del subprotocolo *Pronóstico de Ventas Colaborativo*. Luego de realizado este subprotocolo el protocolo finaliza con éxito. Al final del protocolo se ha definido un segmento de control *Cancelación de Solicitud*, el cual contiene el operador *cancel*, para gestionar excepciones que pueden ocurrir en cualquier punto del protocolo. Este segmento tiene un único camino con la condición *TimeException*, indicando el curso de acción a seguir en el caso que no se cumplan las restricciones de tiempo definidas a los mensajes *agree* y *refuse*. En este caso, el proveedor cancelará la solicitud realizada previamente, expresando que no está más interesado en que el cliente se comprometa a realizar un pronóstico de ventas. Luego el cliente informa al proveedor que la cancelación fue realizada o bien que la cancelación ha fallado. Esto es representado en el segmento de control anidado con el operador *Xor*.

Finalmente, en forma similar a lo definido previamente en el metamodelo, el estereotipo *ProtocolTemplate* especializa un protocolo de interacción. Una plantilla de protocolo se representa en un diagrama de secuencia con el estereotipo <<template>>. Los parámetros de la plantilla de protocolo que deben tomar un valor específico para posibilitar la definición del protocolo son indicados en un comentario asociado a la plantilla, el cual es estereotipado <<ProtocolParameters>>. Cada parámetro tiene la siguiente sintaxis: <nombre del parámetro> = <valor del parámetro>.

La Figura 4-19 define una plantilla de protocolo *Request* basada en la idea de que el protocolo *Solicitud Pronostico Ventas* definido en la Figura 4-18, puede ser genérico y aplicable a diferentes situaciones en donde se requiere la solicitud de una determinada información. Los roles y documentos de negocio definidos en la plantilla son ficticios.

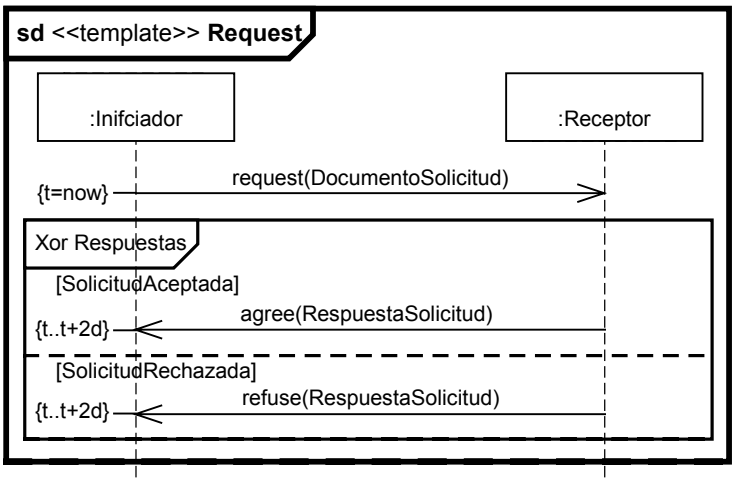


Figura 4-19. Ejemplo de una plantilla de protocolo

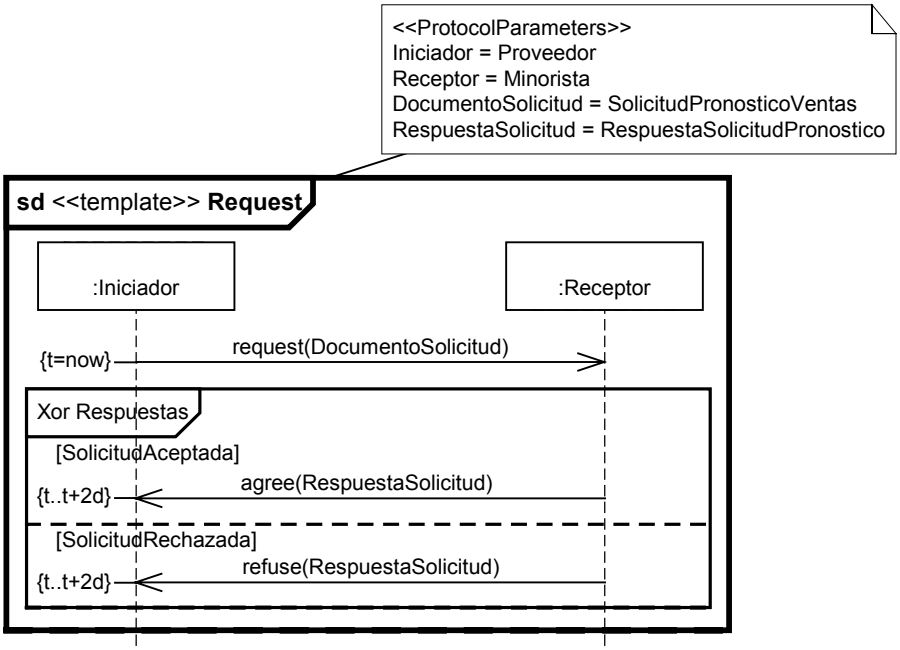


Figura 4-20. Ejemplo de protocolo definido en base a una plantilla de protocolo

Para definir un protocolo de interacción en base a esta plantilla es necesario dar valor a los parámetros de la plantilla. Por ejemplo, la Figura 4-20 muestra la definición del protocolo *Solicitud Pronostico Ventas* en base a la plantilla *Request*. En este caso, el rol iniciador es el *Proveedor*, el rol receptor es el *Minorista*, y los documentos a intercambiar en los mensajes son *SolicitudPronosticoVentas* y

RespuestaSolicitudPronostico. No obstante, la semántica del protocolo no será igual al de la Figura 4-18, dado que en la definición anterior el protocolo invoca a un subprotocolo cuando la solicitud es aceptada y además tiene definido el manejo de excepciones de tiempo.

4.2.2.4. El Paquete *Business Interfaces*

Este paquete especifica los elementos conceptuales para la definición de la *Vista de Interfaces de Negocio*. Este paquete extiende la semántica de las interfaces y recepciones de UML para modelar las interfaces de negocio de los roles con sus servicios, las cuales indican los mensajes que cada rol puede enviar y recibir para dar soporte a los procesos colaborativos. La Figura 4-21 muestra los estereotipos de este paquete junto con las metaclases de UML que están extendiendo y sus valores etiquetados.

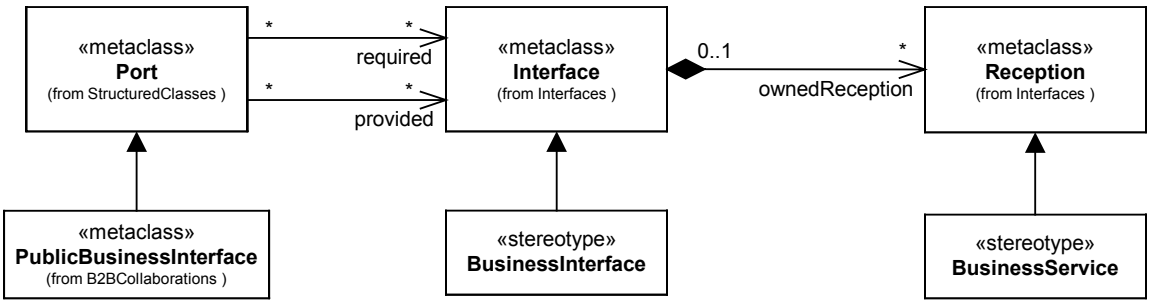


Figura 4-21. Paquete *Business Interfaces* de UP-ColBPIP

En UML, el concepto de *Interface* es utilizado para declarar un conjunto de responsabilidades y obligaciones públicas ofrecidas por un clasificador. Un clasificador puede poseer un conjunto de interfaces provistas e interfaces requeridas, las cuales pueden estar contenidas en un puerto del clasificador. Las primeras representan las obligaciones que las instancias del clasificador tienen con respecto a sus clientes. Las segundas representan los servicios que un clasificador necesita para ejecutar sus procesos internos y cumplir con sus propias obligaciones para con sus clientes. Además, una interfaz puede contener operaciones o recepciones de señales.

El estereotipo *BusinessInterface* extiende la metaclase *Interface* con el objetivo de representar las interfaces de los roles, las cuales siempre deben estar asociadas a un puerto estereotipado *PublicBusinessInterface* de un *PartnerRole*.

El estereotipo *BusinessService* extiende la metaclase *Reception* para representar el manejo y recepción de los mensajes basados en actos de comunicación.

La notación de interfaces y servicios es la misma que la de UML. Interfaces de negocio son definidas como clases con el estereotipo <<BusinessInterface>>. Servicios son definidos dentro de las interfaces con la misma sintaxis que una operación en UML: *<nombre del servicio> (<nombre del documento>:<tipo de documento>)*. Las interfaces provistas son asociadas a una interfaz pública del rol a través del símbolo —○. Las interfaces requeridas son asociadas a una interfaz pública del rol a través del símbolo —⊂ .

Como ejemplo, la Figura 4-22 muestra las interfaces de los roles proveedor y minorista. Dos interfaces son definidas: *InterfazProveedorAMinorista* e *InterfazMinoristaAProveedor*. La primera define los servicios que manejan, en el rol *Proveedor*, las recepciones de mensajes desde el rol *Minorista*. La segunda define los servicios que manejan, en el rol *Minorista*, las recepciones de mensajes desde el rol *Proveedor*. Por lo tanto, la interfaz pública del *Proveedor* tiene como interfaz provista *InterfazProveedorAMinorista*, mientras que en la interfaz pública del *Minorista* ésta es una interfaz requerida. De manera similar, la *InterfazMinoristaAProveedor* es la interfaz que el *Minorista* ofrece al *Proveedor* y es una interfaz requerida en el *Proveedor*. Los servicios definidos dan soporte al intercambio de mensajes en los protocolos en los que los roles están involucrados. Por ejemplo, el servicio *request(SolicitudPronosticoVentas: SolicitudPronosticoVentas)* permite la recepción del mensaje con el acto de comunicación *request* y con un documento de negocio cuyo tipo es *SolicitudPronosticoVentas*.

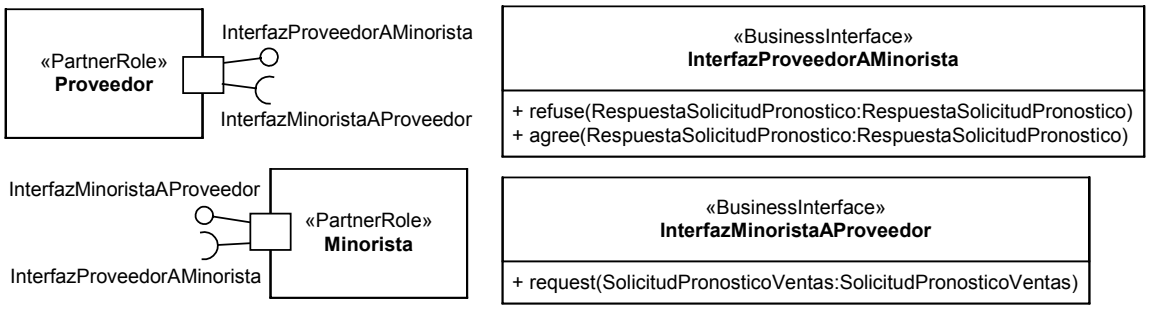


Figura 4-22. Ejemplo de interfaces de negocio

4.2.2.5. Organización de las Vistas y Diagramas en un Modelo UP-ColBPIP

Cuando se define un modelo de colaboraciones B2B y procesos colaborativos en UP-ColBPIP, los submodelos de cada una de las vistas deben ser organizados de una forma lógica expresando las relaciones establecidas en el metamodelo entre los principales conceptos de los diferentes paquetes. Además, es necesario establecer en forma clara los diferentes diagramas que pueden ser utilizados para definir y visualizar los elementos y sus relaciones. La organización de las vistas y elementos de un modelo UP-ColBPIP junto con sus diagramas, es organizada en forma jerárquica, como se indica en la Figura 4-23. El paquete *B2B Collaboration Views* es el paquete principal, en el cual se definen las colaboraciones B2B y todos los elementos de la misma, los cuales son visualizados en un diagrama de estructura compuesta.

Una colaboración B2B es definida en un paquete estereotipado *B2BCollaboration*, en donde el nombre del paquete corresponde al nombre de la colaboración B2B definida en el mismo.

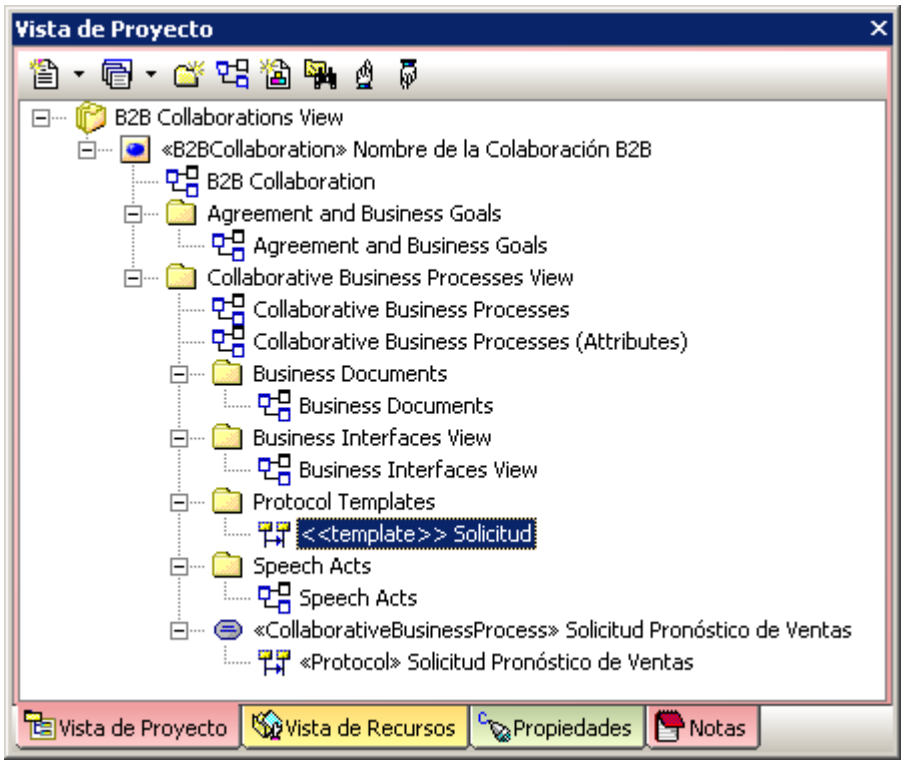


Figura 4-23. Organización de los modelos y diagramas en UP-ColBPIP

El acuerdo de colaboración y las metas de negocio son definidas en el paquete *Agreement and Business Goals*, el cual es un subpaquete del paquete que contiene la

colaboración B2B. El acuerdo de colaboración y las metas de negocio son visualizadas en uno o varios diagramas de clases.

Para representar que una colaboración B2B está compuesta de un conjunto de procesos colaborativos, el paquete que define la colaboración B2B contiene otro paquete denominado *Collaborative Business Processes View*. En este paquete se definen todos los procesos colaborativos que forman parte de la colaboración B2B. Los procesos colaborativos son definidos usando diagramas de caso de uso. También es posible utilizar diagramas de clase para visualizar los atributos de los mismos.

En UML, el comportamiento de un caso de uso puede ser definido a través de interacciones visualizadas en diagramas de secuencias. El atributo heredado *realizedBy* de un caso de uso referencia a la interacción que describe su comportamiento y el atributo heredado *realizes* de una interacción referencia el caso de uso que está describiendo. Por lo tanto, la relación entre un proceso colaborativo y el protocolo de interacción que describe su comportamiento en el metamodelo, se representa en UP-ColBPIP a través de los anteriores atributos mencionados ya provistos por UML. Por lo tanto, cada proceso colaborativo definido puede poseer un protocolo de interacción el cual es visualizado y definido usando diagramas de secuencias. También es posible visualizar los protocolos en diagramas globales de interacciones y diagramas de tiempo.

El paquete *Collaborative Processes View* contiene tres paquetes adicionales: el paquete *Business Documents*, el paquete *Speech Acts* y el paquete *Protocol Templates*. El primero es utilizado para definir los documentos de negocio que forman parte de una colaboración B2B y que serán utilizados en los procesos colaborativos. Estos son definidos en forma separada de los procesos dado que un documento de negocio puede ser utilizado en diferentes procesos colaborativos. Los documentos de negocio con sus tipos pueden ser visualizados en diagramas de clases.

El paquete *Speech Acts* define los actos de comunicación a ser utilizados en los protocolos de interacción. Los actos de comunicación pueden ser definidos ad-hoc o pueden ser tomados de alguna librería de actos de comunicación estándar, como por ejemplo FIPA ACL (FIPA, 2002). Los actos de comunicación pueden ser visualizados en diagramas de clases.

El paquete *Protocol Templates* contiene las plantillas de protocolo que pueden ser utilizadas para definir nuevos protocolos de interacción. Al igual que los protocolos de

interacción, las plantillas de protocolo son visualizadas en diagramas de secuencia y opcionalmente también es posible utilizar diagramas globales de interacciones o diagramas de tiempo.

El paquete *Business Interfaces View* contiene las interfaces de negocio de los roles con sus servicios de negocio. Las interfaces de los socios son visualizadas en un diagrama de clases.

4.3. Aplicación del Lenguaje UP-ColBPIP a un Caso de Estudio

En esta sección se describe la utilización del lenguaje de modelado de procesos colaborativos UP-ColBPIP a través de un caso de estudio, el cual consiste en el establecimiento de una alianza de colaboración entre las empresas de producción "Pincelap S.A." y "Tractores Puma S.A". "Pincelap S.A." elabora pinturas de diferentes tipos (látex, sintéticas, acrílicas, etc) y provee pintura acrílica a "Tractores Puma S.A." quien las utiliza en su proceso de fabricación. El objetivo de las empresas es establecer una colaboración B2B para llevar a cabo un aprovisionamiento colaborativo con respecto a dos pinturas específicas. A través de esta colaboración, "Pincelap S.A." podrá contar con información más precisa acerca de los requerimientos futuros de pinturas por parte de su cliente, lo que le permitirá mejorar sus procesos de planificación de producción para asegurar la entrega de las pinturas solicitadas en el lugar y en el momento requeridos. Por su parte, esto le permite a "Tractores Puma S.A." planificar con menos incertidumbre su producción. En términos globales el proceso de gestión del flujo de materiales en la cadena de suministro en cuestión, se verá sustancialmente mejorado debido fundamentalmente a la reducción del nivel de inventarios necesarios, pudiendo esto mejorar la competitividad de ambas empresas.

Para implementar el aprovisionamiento colaborativo, las empresas han decidido aplicar un modelo de colaboración Vendor-Managed Inventory (VMI). En un modelo VMI, el proveedor determina cuándo enviar los materiales y cuánto enviar para mantener el inventario del cliente en el nivel acordado. El cálculo del aprovisionamiento puede estar basado en un pronóstico o sólo en información acerca del consumo de los materiales. Particularmente, el modelo VMI adoptado sigue los lineamientos provistos por el *Modelo VMI basado en Pronóstico* propuesto por EIDX (Electronics Industry Data Exchange Association) (CompTIA EIDX, 2005). EIDX es un consorcio de

empresas que provee estándares para el intercambio de información como así también guías generales de procesos de negocio colaborativos para la cadena de suministro de la industria electrónica. No obstante, aunque los productos objetos de colaboración por parte de las empresas "Pincelap S.A." y "Tractores Puma S.A." pertenecen a una cadena de suministro de la industria química, éstas han detectado útil el modelo VMI basado en pronóstico de EIDX.

Para el intercambio de datos, las empresas han decido utilizar el estándar CIDX (Chemical Industry Data Exchange Association) (CIDX, 2004). CIDX es un consorcio de empresas que también provee estándares para el intercambio de información pero para la cadena de suministro de la industria química.

Los modelos VMI propuestos por los consorcios de empresas citados no son directamente aplicables a cualquier relación entre empresas. Estos sólo constituyen guías generales acerca de los procesos colaborativos que deberían formar parte de la colaboración y el comportamiento general que deberían tener los mismos. Además, las condiciones del acuerdo y las metas de negocio de los socios son particulares a cada relación. Por lo tanto, en este caso de estudio se describen los diferentes pasos en la definición de los procesos colaborativos que constituyen el modelo VMI a aplicar a través del uso del lenguaje UP-ColBPIP.

Para definir los procesos colaborativos con UP-ColBPIP, es necesario utilizar una herramienta Case basada en UML que posibilite el uso de perfiles UML. Por lo cual, en este caso de estudio el perfil UML UP-ColBPIP ha sido implementado en la herramienta Case UML *Enterprise Architect* [5].

Siguiendo el caso de estudio, a continuación se describe paso por paso la definición de las cuatro vistas soportadas por UP-ColBPIP.

4.3.1. Definiendo la Vista de la Colaboración B2B

El nombre dado a la colaboración B2B definida para el caso de estudio es *VMI-based Collaborative Replenishment*. El Diagrama de Estructura Compuesta de la Figura 4-24 muestra esta colaboración la cual indica que "Pincelap S.A." desempeña el rol de

[5] Enterprise Architect (EA), http://www.sparxsystems.com

proveedor y "Tractores Puma S.A." desempeña el rol de cliente. Ambos se comunican entre sí a través de una relación B2B, para lo cual se ha definido una interfaz de negocio pública en ambos roles. La colaboración B2B tiene asociado el nombre del acuerdo de colaboración que los socios definen para gobernar la colaboración.

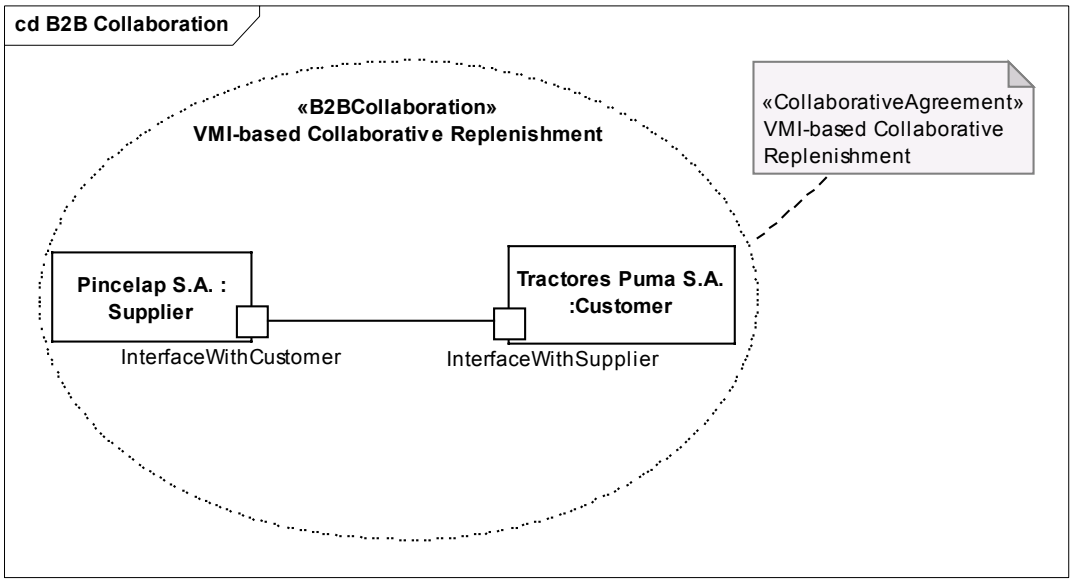


Figura 4-24. Diagrama de estructura compuesta que describe la Colaboración B2B

En la Figura 4-25 se muestra el Diagrama de Clases que describe el acuerdo de colaboración y la jerarquía de metas de negocio que los socios han acordado. El acuerdo de colaboración establecido define varios atributos. Los dos primeros definen el período de tiempo que abarca el acuerdo. Los tres siguientes definen algunos términos comerciales del acuerdo, como ser los productos objetos de colaboración, el precio acordado para dichos productos, y la moneda tomada como referencia. El atributo *leadTime* indica los tiempos de provisión normales acordados para cada producto. Los atributos *minInventoryLevels* y *maxInventoryLevels* definen los niveles de inventario mínimos y máximos acordados para cada producto. El atributo *acceptableToleranceForPlans* representa los niveles porcentuales de tolerancia aceptables ante cambios entre un plan actual y el plan previo correspondiente.

Las metas de negocio han sido definidas de acuerdo a los indicadores de rendimientos claves provistos por CIDX para medir los resultados de la colaboración con un modelo VMI. Esto es de suma importancia, debido a que una de las principales fallas identificadas por muchas empresas en la implementación de estos modelos, es que

los objetivos de la relación de colaboración no son claramente definidos y por lo tanto tampoco es posible medir el éxito de la colaboración.

La meta principal indica el objetivo principal de un modelo VMI: *Disminuir los Niveles Promedio de Inventario del Cliente*. Esta meta es cuantitativa y la misma indica que: el valor actual del inventario promedio d el cliente es de 2500 unidades, el valor a alcanzar acordado entre los socios es de 1500 unidades, la unidad de medida utilizada es unidades de producto, el método de cálculo consiste en la suma del valor promedio del inventario para cada producto para todo el período que dure el acuerdo, y la frecuencia de medición o actualización de la meta es de dos meses.

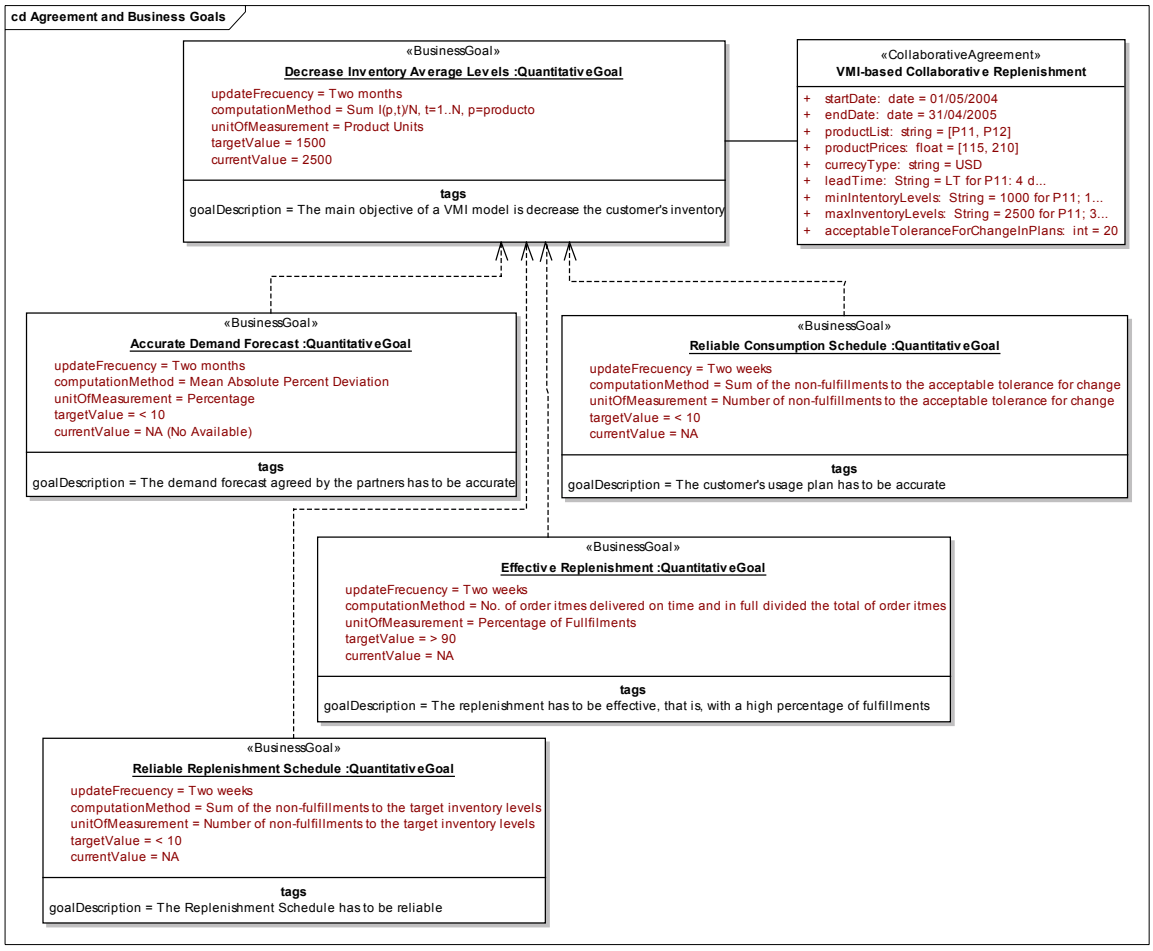


Figura 4-25. Diagrama de clase que describe el acuerdo de colaboración y las metas de negocio

El cumplimiento de la meta anterior depende del cumplimiento de cuatro submetas, las cuales indican objetivos más específicos a cumplir. Una de ellas es la meta de negocio *Accurate Demand Forecast*. Esta meta significa que los socios deben definir en

forma conjunta un pronóstico de demanda lo más preciso o exacto posible. En el modelo VMI basado en pronósticos, existirá una única orden para todo el período del acuerdo, denominada *Blanket Purchase Order (BPO)*, la cual contiene inicialmente las cantidades pronosticadas a solicitar por el cliente para cada producto. Para determinar si el pronóstico es adecuado, la meta define que: la unidad de medida del pronóstico es un porcentaje de la precisión del pronóstico, el valor a alcanzar debe ser menor al 10%, el método de cálculo (la medida de error del pronóstico) utilizado es la Desviación Porcentual Media Absoluta, y la frecuencia de actualización es de dos meses.

Debido a que el modelo VMI utilizado está basado en pronósticos de consumo, también se define la meta *Reliable Consumption Schedule*, la cual representa el compromiso de un cliente para generar planes de consumo que estén dentro de la tolerancia de cambio establecida entre un plan de consumo actual y su plan previo. Esta meta define que el número de incumplimientos a las tolerancias de cambio definidas debe ser menor a 10 y el método de cálculo es la suma de los incumplimientos a las tolerancias de cambio durante todo el período del acuerdo. La frecuencia de actualización es de dos semanas.

La meta *Reliable Replenishment Schedule* representa el compromiso del proveedor de generar planes de aprovisionamiento que se ajusten a los niveles de inventarios acordados (mínimos y máximos). Esta meta es definida en forma similar a la anterior. La misma define que el número de incumplimientos a los niveles de inventarios acordados debe ser menor a 10 y el método de cálculo es la suma de los incumplimientos a los niveles de inventarios acordados. La frecuencia de actualización es de dos semanas.

Finalmente, se define la meta *Effective Replenishment*, la cual representa el compromiso del proveedor de aprovisionar en tiempo y en cantidad el inventario del cliente de acuerdo a lo definido en la BPO. Esta meta define que la unidad de medida será el porcentaje de cumplimientos, y el valor a alcanzar debe ser mayor a un 90%. El método de cálculo es el número de ítems entregados en tiempo y en forma completa, dividido por el número total de ítems de la BPO. La frecuencia de actualización es de dos semanas.

4.3.2. Definiendo la Vista de los Procesos de Negocio Colaborativos

Los procesos de negocio colaborativos identificados son descriptos en el Diagrama de Casos de Uso mostrado en la Figura 4-26. Estos son los procesos que los socios han acordado realizar. Los mismos han sido identificados a partir de las metas de negocio definidas previamente. Por lo tanto, el proceso colaborativo *Forecast-based VMI* es el principal y tiene asociada la meta de negocio *Decrease Inventory Average Levels*. Siguiendo las metas definidas, este proceso está compuesto de cuatro subprocesos, los cuales permiten alcanzar las restantes submetas. Estos subprocesos son: *Blanket Purchase Order*, *Consumption Schedule*, *Release Against Blanket Purchase Order* y *Replenishment Schedule*. De esta manera cada proceso colaborativo tiene asignado la meta de negocio que debe cumplir. El diagrama también indica los roles involucrados en el proceso principal, los cuales también están involucrados en los subprocesos.

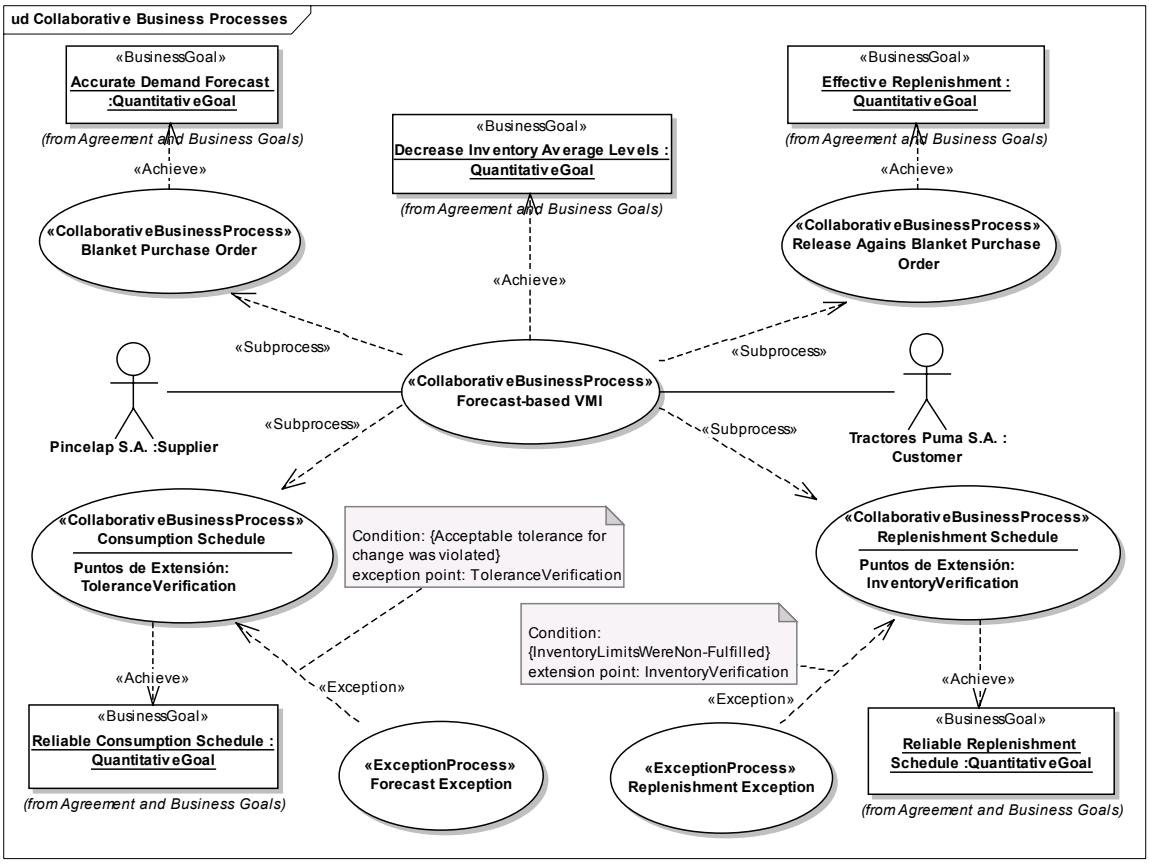


Figura 4-26. Diagrama de casos de uso que describe los procesos colaborativos

Cada uno de los procesos ha sido definido de manera informal siguiendo los lineamientos utilizados para la definición de casos de uso. Como ejemplo de una definición informal de un proceso colaborativo, la Figura 4-27 muestra el escenario

principal definido para el proceso *Forecast-based VMI*. Existirá una única instancia de dicho proceso para todo el período que dure la colaboración B2B. El escenario principal de este proceso comienza con el cliente, quien inicia una instancia del subproceso *Blanket Purchase Order* con el objetivo de acordar con el proveedor una *BPO* para todo el período del acuerdo, la cual básicamente define un pronóstico de órdenes y representa un compromiso de mediano plazo del proveedor de suministrar los materiales a través de múltiples emisiones de corto plazo generadas para satisfacer los requerimientos del cliente. Una vez definida la BPO, ambos inician un ciclo de planificación, en donde cada ciclo involucra los restantes pasos.

El segundo paso del proceso consiste en el envío por parte del cliente del plan de consumo y la posible respuesta del proveedor si una excepción ha ocurrido. Este paso es realizado invocando el subproceso *Consumption Schedule*.

El tercer paso consiste en la determinación por parte del proveedor de si es necesario el aprovisionamiento durante el ciclo actual de planificación. Si es así, el proveedor realiza el envío del material y envía una notificación del mismo. Esto es realizado a través del subproceso *Release Against Blanket Purchase Order*.

El cuarto paso consiste en el envío por parte del proveedor del plan de aprovisionamiento. Esto es realizado a través del subproceso *Replenishment Schedule*.

Finalmente, el último paso consiste en realizar un nuevo ciclo de aprovisionamiento y comenzar nuevamente con el segundo paso, es decir, con el subproceso *Consumption Schedule*.

El comportamiento detallado de cada uno de los procesos identificados es descrito en la vista de los protocolos de interacción.

Como se indica en la Figura 4-26, además de los procesos colaborativos dos procesos de excepción han sido definidos: *Forecast Exception* y *Replenishment Exception*. El primero tiene como objetivo gestionar las excepciones que pueden ocurrir en el proceso *Consumption Schedule* con respecto a las tolerancias de cambio establecidas en el acuerdo de colaboración, como lo indica la asociación estereotipada *Exception*. El lugar dentro del proceso colaborativo donde esta excepción puede ocurrir es indicado por el punto de excepción *ToleranceVerification* definido en el proceso colaborativo. Este punto indica que la excepción puede ocurrir después que el proveedor

recibe el plan de consumo enviado por el cliente y verifica que esté dentro de las tolerancias para el cambio. Este paso es indicado como una extensión al escenario principal del proceso en la definición informal. El comentario asociado a la relación entre los procesos establece la condición o criterio de excepción a evaluar para determinar la ocurrencia de una excepción. Esta condición define que el proceso de excepción será disparado cuando el cambio entre un plan de consumo actual y el plan de consumo previo supera las tolerancias acordadas. El proceso de excepción *Forecast Exception* describe la forma en que dicha excepción será resuelta.

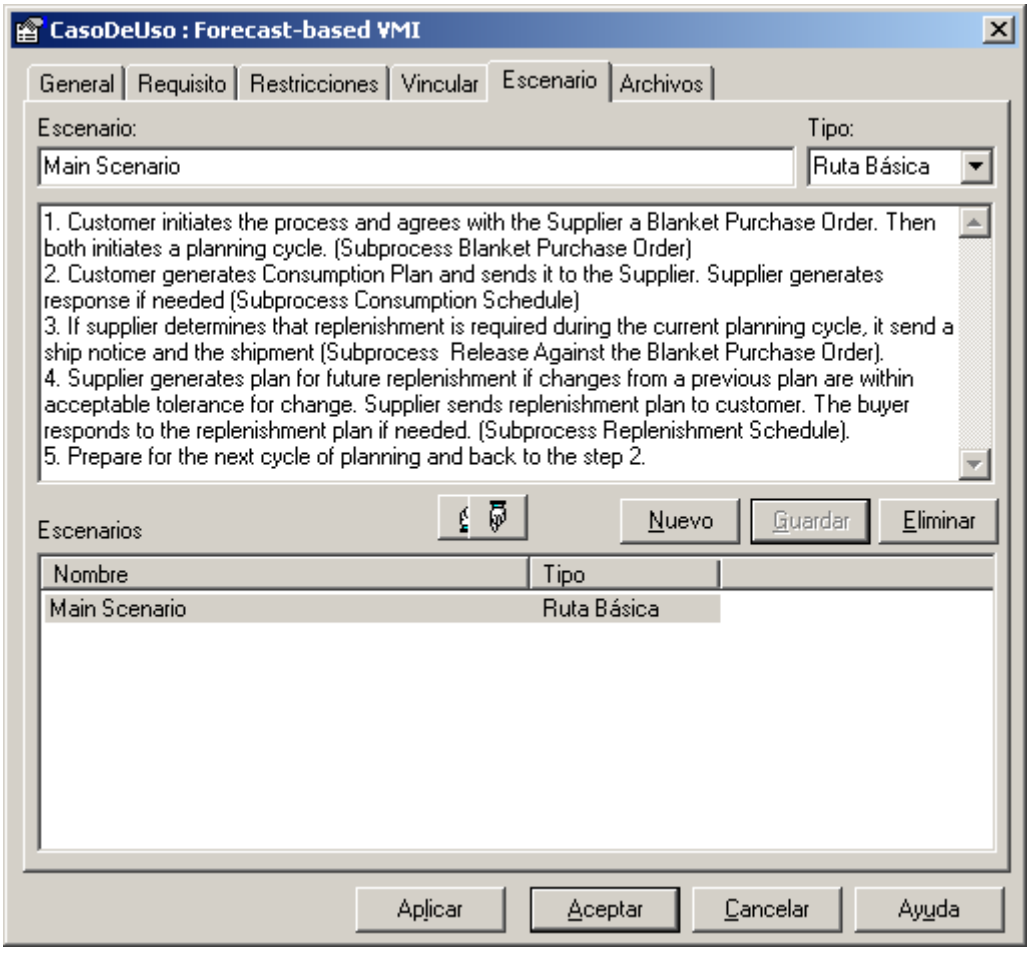


Figura 4-27. Escenario principal del proceso *Forecast-based VMI*

El proceso de excepción *Replenishment Exception* tiene como objetivo gestionar las excepciones que pueden ocurrir en el proceso *Replenishment Schedule* con respecto a los niveles mínimos y máximos establecidos para el inventario. La condición de excepción define que una excepción ocurre y debe ser resuelta a través de este proceso cuando los niveles mínimos y máximos establecidos para el inventario no son satisfechos por el plan de aprovisionamiento que envió el proveedor. La condición de

excepción es evaluada cuando el cliente recibe el plan de aprovisionamiento y verifica si conduce a los niveles de inventarios acordados.

La Figura 4-28 muestra un Diagrama de Clases que describe los atributos definidos en cada uno de los procesos colaborativos y los documentos de negocio a ser intercambiados en los mismos. Como ejemplo, el proceso *Consumption Schedule* tiene definido que una instancia del proceso se inicia cuando el cliente generó el plan de consumo. La precondición indica que antes del comienzo de este proceso, la BPO debe estar acordada y definida. La poscondición indica que el proceso debe finalizar con un plan de consumo que esté dentro de las tolerancias de cambios establecidas en el acuerdo. Finalmente, dos documentos de negocio son definidos en el proceso: el plan de consumo que será enviado por el cliente y la respuesta al plan de consumo que será enviada por el proveedor.

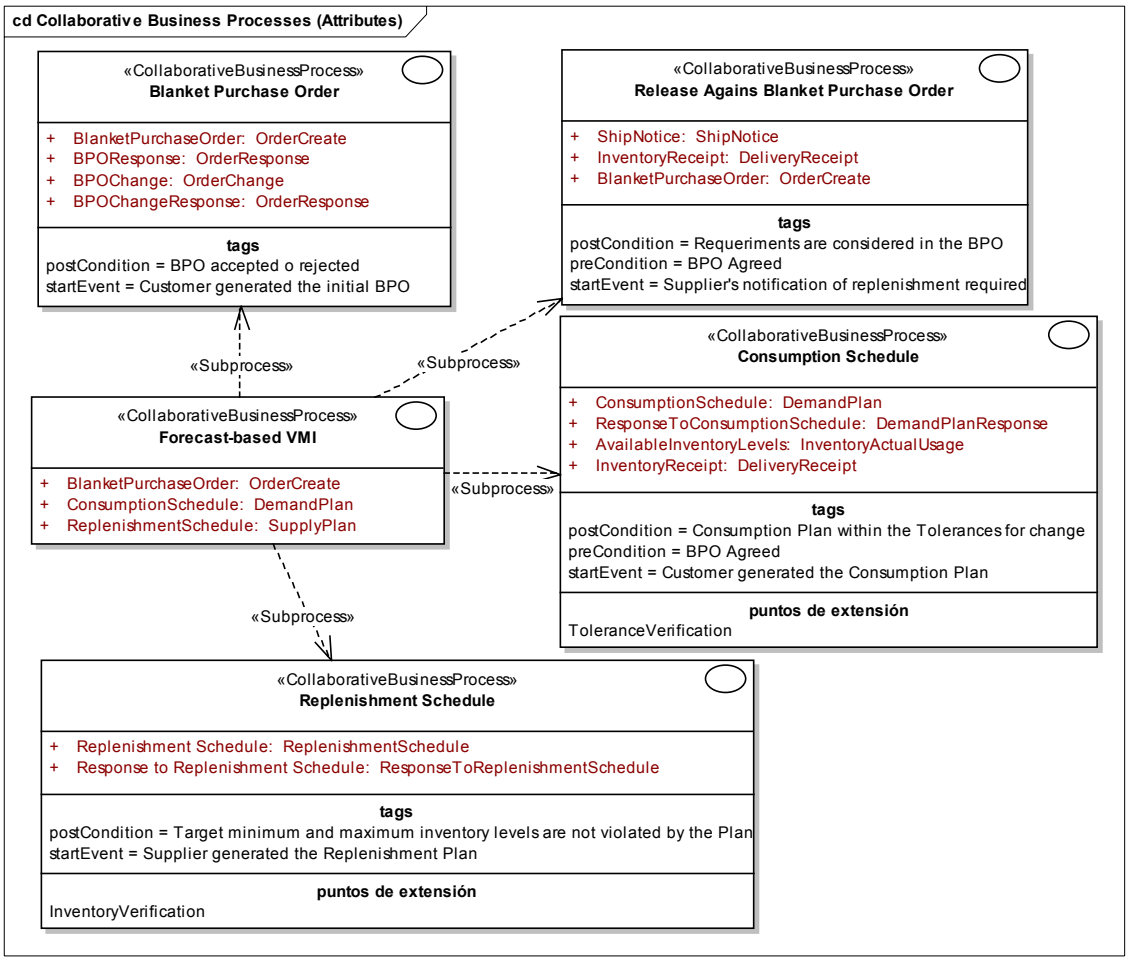


Figura 4-28. Diagrama de clases que describe los atributos de los procesos

Los documentos de negocio definidos en la colaboración B2B y la relación a sus tipos son mostrados en el diagrama de clase de la Figura 4-29. Los tipos de documentos se corresponden con los provistos por el estándar CIDX. Para los tipos de documentos de negocio *OrderCreate* y *OrderResponse*, el atributo *specification* es mostrado, el cual referencia el esquema XML, provisto por el estándar, que describe la estructura de este tipo de documento.

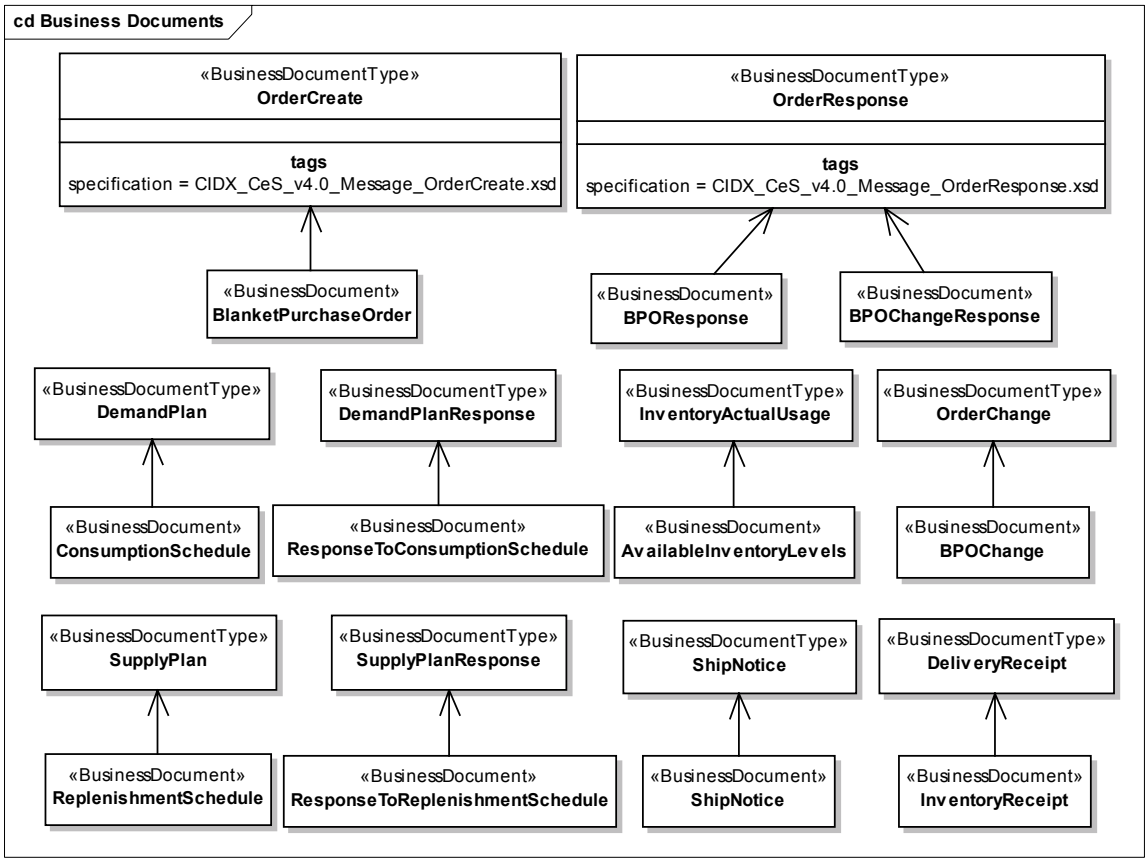


Figura 4-29. Diagrama de clases que describe los documentos de negocio

4.3.3. Definiendo la Vista de los Protocolos de Interacción

Habiendo realizado el análisis de los procesos colaborativos, el próximo paso consiste en especificar el comportamiento formal de cada uno de ellos a través de un protocolo de interacción. El nombre de cada protocolo es el mismo que el nombre del proceso que está materializando.

4.3.3.1 Protocolo de Interacción Forecast-based VMI

Comenzamos describiendo el protocolo de interacción *Forecast-based VMI*, el cual especifica el comportamiento del proceso principal (Figura 4-30). Este protocolo

comienza con una referencia al protocolo *Blanket Purchase Order*, la cual representa que una instancia del protocolo *Blanket Purchase Order* será iniciada con el objetivo de que los socios acuerden una *BPO*. Una vez instanciado el protocolo, varios ciclos de planificación y reaprovisionamiento pueden ocurrir. Esto es representado por el segmento de control *Planning and Replenishment Cycles* con el operador *Loop* [6]. La condición asociada al camino del segmento especifica que al menos un ciclo tiene que ocurrir y que *n* ciclos pueden ocurrir mientras la fecha actual más el tiempo estimado del ciclo sea menor a la fecha de fin definida en el acuerdo de colaboración.

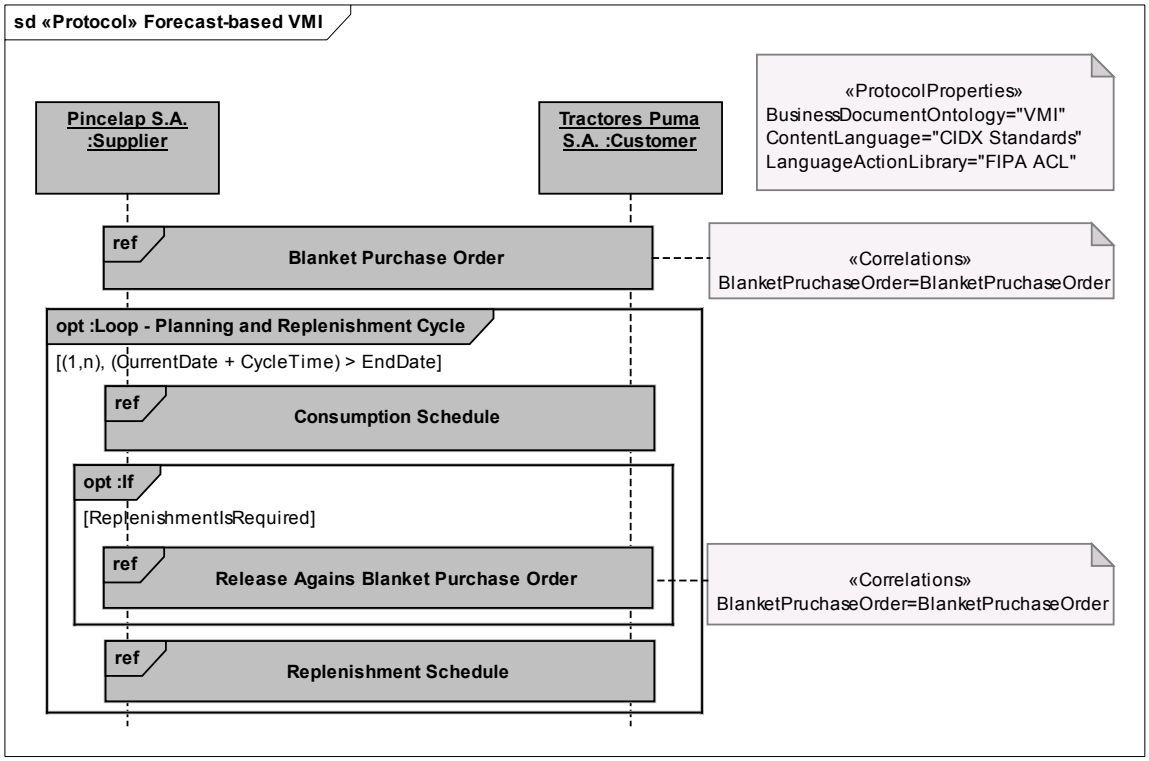


Figura 4-30. Diagrama de secuencia del protocolo *Forecast-based VMI*

El ciclo comienza con una invocación al protocolo *Consumption Schedule* con el objetivo de que el cliente informe al proveedor el plan de consumo de los productos para que éste último pueda calcular los requerimientos de aprovisionamiento. Una vez finalizada la instancia de este proceso, si el proveedor determina que es necesario realizar un aprovisionamiento en este ciclo, entonces una instancia del protocolo

[6] La herramienta utilizada no permite visualizar el valor etiquetado que define el operador del estereotipo *ControlFlowSegment*. Por lo tanto, en los protocolos aquí definidos, este operador es visualizado con la sintaxis: opt: <operator> [-<nombre segmento>].

Release Against Blanket Purchase Order es generada. Esto es representado por un segmento de control con el operador *If* con la condición *ReplenishmetIsRequired* asociada. Luego se continúa con el protocolo *Replenishment Schedule*. Si el aprovisionamiento no es requerido se continúa directamente con este último protocolo con el objetivo de que el proveedor informe al cliente de los planes de aprovisionamiento. Cuando la instancia de este protocolo finaliza, se evalúa nuevamente la condición del segmento de control *Planning and Replenishment Cycles* para determinar si un nuevo ciclo debe ser realizado. Si esto es así, el protocolo continúa nuevamente con el comienzo de este segmento de control. De lo contrario, si la fecha actual más el tiempo estimado del ciclo es mayor o igual a la fecha de fin del acuerdo de colaboración, el proceso *Forecast-based VMI* finaliza dado que no existe otro elemento a evaluar.

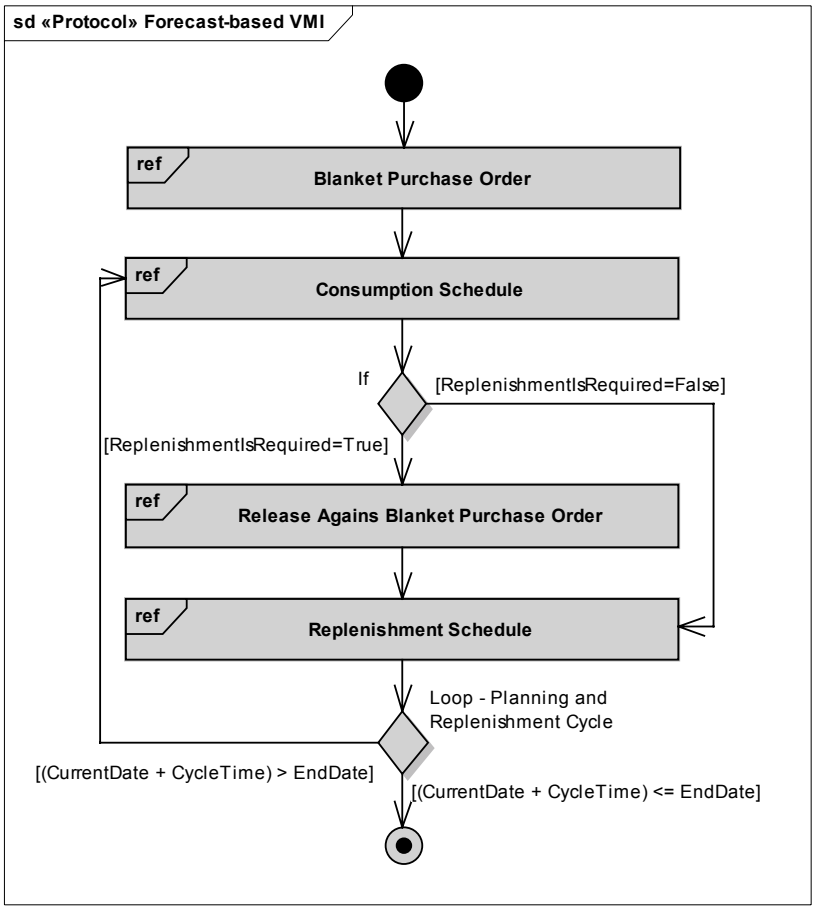


Figura 4-31. Diagrama global de interacción del protocolo *Forecast-based VMI*

El protocolo *Forecast-based VMI* tiene definido tres atributos, como se indica en el comentario <<ProtocolProperties>>. El primer atributo indica que la ontología que

describe la semántica de los documentos es la ontología *VMI*. Esta ontología no es un estándar sino que ha sido definida previamente por los socios para entender la semántica de los documentos de negocio. El lenguaje de contenido a utilizar es el estándar CIDX. Finalmente, la librería de actos de comunicación a utilizar es FIPA ACL (FIPA, 2002), descrita en el Anexo A de esta tesis.

Todos los protocolos definidos para el caso de estudio tienen especificados estos atributos con los mismos valores que los del protocolo *Forecast-based VMI*.

La Figura 4-31 muestra otra vista del protocolo *Forecast-based VMI* a través de un diagrama global de interacciones.

4.3.3.2. Protocolo de Interacción Blanket Purchase Order

La Figura 4-32 muestra el diagrama de secuencia del protocolo *Blanket Purchase Order*. Este protocolo representa el proceso de negociación entre el proveedor y el cliente para acordar una BPO, orden de compra para todo el período que dure el acuerdo. En la BPO se define la cantidad total, y otros detalles sobre las condiciones y términos de precios que no hayan sido definidos en el acuerdo.

El protocolo comienza con el cliente, quien en base a sus procesos internos de planificación de los requerimientos de materiales determina las cantidades necesarias de los materiales y propone una BPO inicial. Esto es indicado por el primer mensaje del protocolo[7] que es dirigido desde el cliente hacia el proveedor. El acto de comunicación *propose* asociado al mensaje indica que el cliente propone al proveedor comprarle una cierta cantidad de los diferentes materiales y a un precio determinado, lo cual es especificado en la BPO. Luego, varios ciclos pueden ocurrir como consecuencia de la negociación acerca de la BPO. Esto es representado por el segmento de control *BPO Negotiation* con el operador *Loop*. Este segmento indica que los mensajes y caminos que lo componen deben ejecutarse al menos una vez y que un nuevo ciclo de negociación deberá realizarse mientras la condición *NegotiationIsRequired* sea verdadera, es decir, mientras los roles estén interesados en seguir negociando.

[7] Un mensaje asincrónico debería ser indicado con la flecha "→". No obstante, la herramienta utilizada no permite distinguir en los diagramas de secuencias entre mensajes sincrónicos y asincrónicos.

Dentro de este bucle, el proveedor evalúa la propuesta del cliente y sólo uno, de los dos caminos alternativos y mutuamente excluyentes, puede ocurrir. Esto es representado por el segmento de control *BPO Changes* definido con el operador *Xor*.

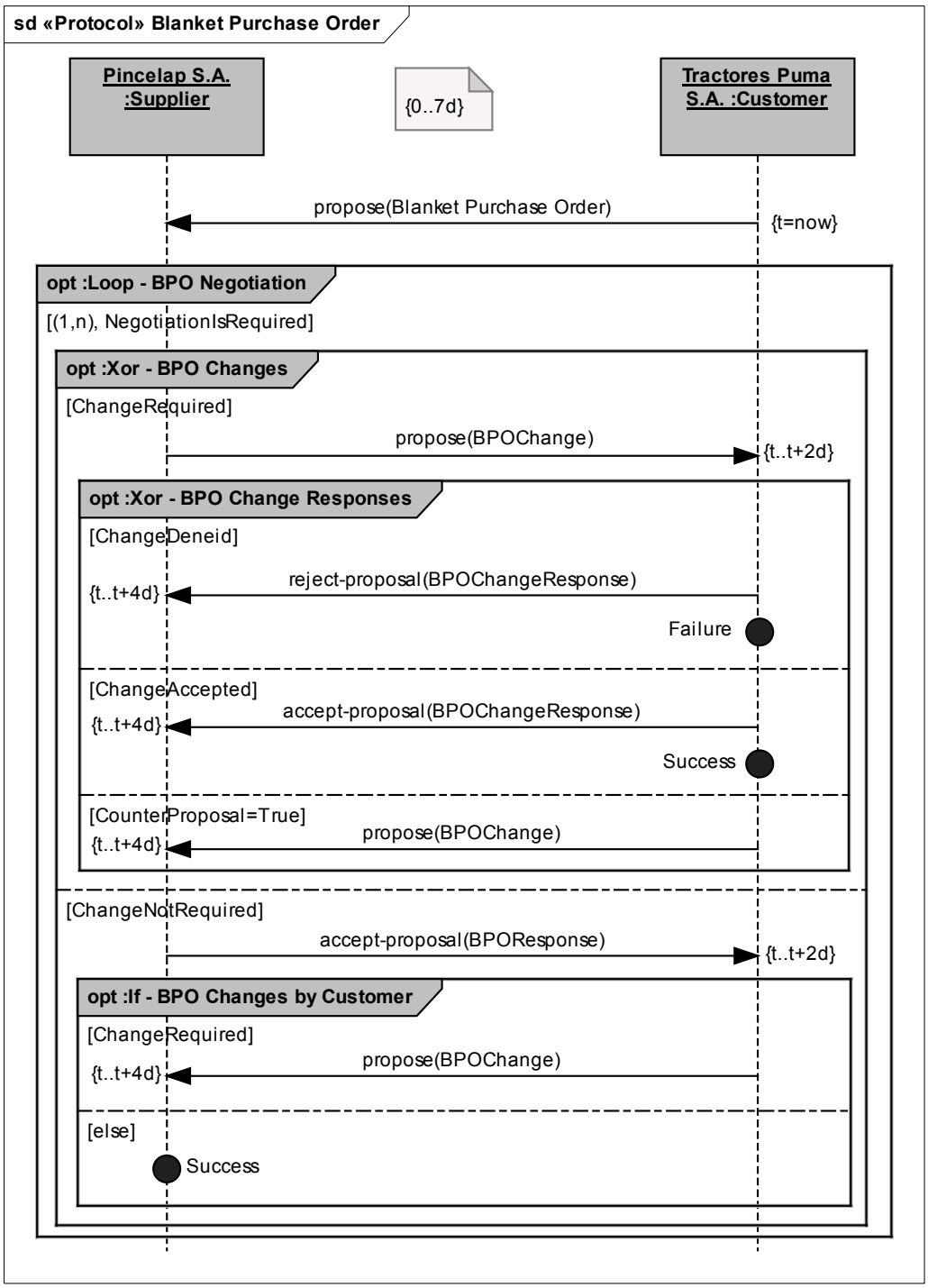


Figura 4-32. Diagrama de secuencia del protocolo *Blanket Purchase Order*

Si el proveedor requiere cambios en la BPO, se verifica la condición *Change Required*, entonces el primer camino del segmento *BPO Changes* es ejecutado.

Mediante un mensaje *propose(BPOChange)* el proveedor realiza una contrapropuesta a lo propuesto previamente por el cliente. El cliente evalúa la propuesta del proveedor y puede responder de tres formas diferentes, como lo expresa el segmento de control *BPO Change Responses* con el operador *Xor*. Una alternativa consiste en el rechazo de los cambios propuestos por el proveedor (se verifica la condición *Change Denied* asociada), lo cual es indicado por el primer camino del segmento con el mensaje *reject-proposal(BPO ChangeResponse)*. El acto de comunicación *reject-proposal* asociado al mensaje indica que el cliente rechaza la propuesta. El documento transportado por este mensaje indica las razones del rechazo. En este camino, el protocolo termina con el evento *Failure* para indicar que el protocolo no ha finalizado con éxito. Otra alternativa es que el cliente envíe el mensaje *accept-proposal(BPOChangeResponse).* El acto de comunicación *accept-proposal* indica que acepta los cambios propuestos por este último. En este caso el protocolo finaliza en forma exitosa con el evento *Success*. La tercera alternativa es que el cliente realice una contrapropuesta como lo expresa el mensaje *propose(BPOChange)*. En este caso, tanto el segmento *BPO Change Responses* como el segmento *BPO Changes* finalizan y el protocolo continúa con un nuevo ciclo de negociación.

En el caso que el proveedor no requiera cambios a la BPO propuesta inicialmente por el cliente (se verifica la condición *ChangeNoRequired*), se ejecutará el segundo camino del segmento *BPO Changes*. Mediante el mensaje *accept-proposal(BPOResponse)*, el proveedor acepta la BPO propuesta por el cliente, y se compromete a cumplir lo establecido en la orden en cuanto a cantidades y precios.

El cliente también puede requerir cambios en la BPO ya aceptada por el proveedor. Esto es indicado por el segmento *BPO Changes by Customer*. Si el cliente requiere cambios en la BPO (se verifica la condición *ChangeRequired*), entonces éste envía el mensaje *propose(BPOChange)* realizando una nueva propuesta al proveedor. Luego, el protocolo continuará con un nuevo ciclo de negociación, si así es requerido por uno de los socios. En el caso que el cliente no requiera cambios, el protocolo finaliza en forma exitosa, como lo indica la terminación *Success* del camino que tiene asociado la condición *else*.

La negociación de la BPO puede llevar a realizar *n* iteraciones del camino del segmento *BPO Negotiation*. No obstante, la restricción de duración *{0..7}* indicada

arriba del protocolo define una fecha límite para el mismo, es decir, que el protocolo no puede ser ejecutado por más de siete días. Además, los mensajes de negocio del protocolo también tienen restricciones de tiempo definidas utilizando tiempos relativos. El primer mensaje tiene asociada la observación de tiempo *{t=now}* que indica la asignación de la fecha actual a la variable *t* cuando se recibe el mensaje. Los restantes mensajes tienen restricciones de tiempo que representan fechas límites para su realización computada desde la fecha en que se envió el primer mensaje.

4.3.3.3. Protocolo de Interacción Comsuption Schedule

La Figura 4-33 muestra el diagrama de secuencia del protocolo *Consumption Schedule*. Este protocolo comienza con el cliente enviando al proveedor la notificación de recepciones de materiales, los niveles de disponibilidad de inventario de los materiales y el plan de consumo de los materiales. Esto es indicado por el segmento de control *Inventory Information* con el operador *And*. Este segmento indica que tres mensajes deben ser enviados, y que los mismos pueden ser realizados en forma concurrente y en cualquier orden. Los mensajes están basados en el acto de comunicación *inform* para representar el envío de información del cliente, con el objetivo que el proveedor tome conocimiento de la misma. La restricción de tiempo *{0..2}* asociada al segmento indica que los mensajes deben ser ejecutados en el término de dos días, a partir del comienzo de la instancia del protocolo.

Luego, el proveedor determina si la diferencia entre el plan de consumo actual informado por el cliente y el acordado supera las tolerancias establecidas en el acuerdo. Esto es indicado por el segmento con el operador *Exception* que contiene la condición de excepción *ToleranceForChangeWasViolated*. En el caso que no exista una excepción, el protocolo finaliza con éxito en forma implícita. De lo contrario, la excepción debe ser gestionada por un proceso de excepción. Esto es representado por la referencia al protocolo *Plan Exception Management*. Este último ha sido definido como una plantilla de protocolo para materializar los procesos de excepción definidos en la vista previa (Sección 4.3.2). Por lo tanto, este protocolo es utilizado aquí para materializar el proceso *Forecast Exception*.

Debido a que este protocolo es una plantilla, para su uso como protocolo de interacción es necesario asignar valor a los parámetros de la misma. Estos valores son indicados con el comentario <<ProtocolParameters>> asociado al protocolo

referenciado. En este caso, el rol iniciador será el proveedor y el rol que responde será el cliente. El documento de negocio que indica la notificación de la excepción es el *ResponseToConsumptionSchedule*. El documento utilizado para dar una respuesta a la excepción es el *ConsumptionSchedule*. Los detalles de cómo la excepción es resuelta son indicados por la definición de la plantilla *Plan Exception Management*, la cual es explicada posteriormente. Una vez finalizado el subprotocolo, el protocolo finaliza.

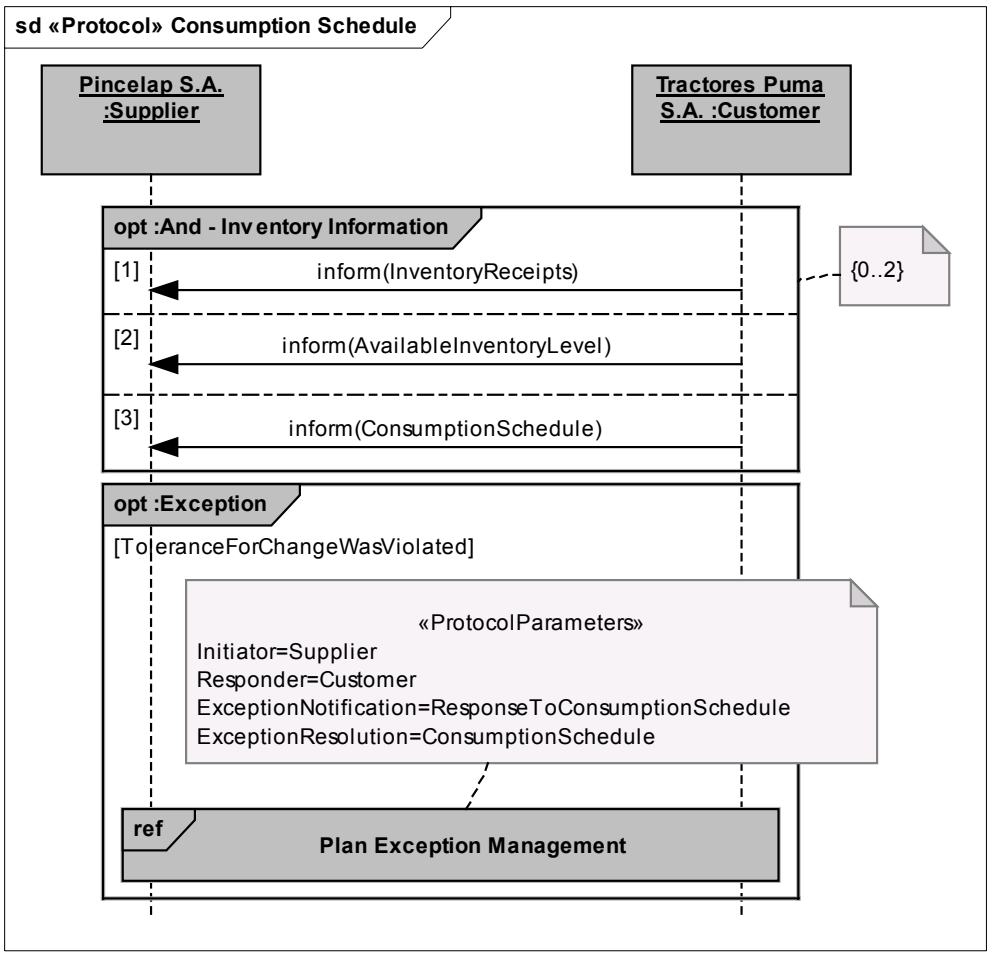


Figura 4-33. Diagrama de secuencia del protocolo *Consumption Schedule*

4.3.3.4 Protocolo de Interacción Release Against Blanket Purchase Order

El diagrama de secuencia que describe el protocolo *Release Against Blanket Purchase Order* se muestra en la Figura 4-34. En este protocolo, el proveedor determina si uno o más aprovisionamientos son necesarios en el presente ciclo. La decisión la realiza en base a sus procesos internos y teniendo en cuenta el plan de consumo informado por el cliente, la notificación de recepciones de materiales y los niveles de disponibilidades de inventarios. Además, el proveedor debe determinar si la BPO

cubrirá los requerimientos de aprovisionamiento del cliente dentro del ciclo. Por lo tanto, si estos requerimientos no son cubiertos, los socios deberán ejecutar nuevamente el protocolo *Blanket Purchase Order* para acordar una nueva BPO que se ajuste a los nuevos requerimientos. Esto es indicado por el segmento de control con el operador *If*.

Independientemente de si se ha acordado una nueva BPO o no, el protocolo continúa con el proveedor enviando un mensaje *inform(ShipNotice)*. Dicho mensaje significa que el proveedor notifica al cliente de los envíos de materiales ya despachados. Puede ocurrir que exista más de una notificación de envíos dentro del tiempo que dura el protocolo, lo cual significa que habrá más de un mensaje de notificación de envíos, como lo indica el asterisco y la condición asociada al mensaje. El protocolo finalizará cuando los aprovisionamientos no sean más requeridos o bien cuando se cumpla la fecha límite de 7 días establecida en el protocolo.

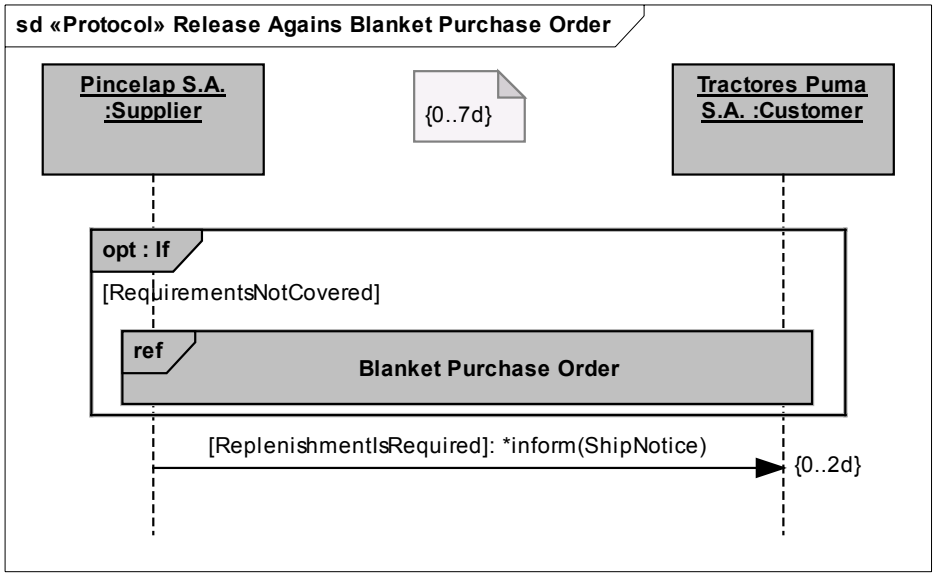


Figura 4-34. Diagrama de secuencia del protocolo *Release Against Blanket Purchase Order*

4.3.3.5. Protocolo de Interacción Replenishment Schedule

El diagrama de secuencia del protocolo *Replenishment Schedule* es mostrado en la Figura 4-35. Luego de haber realizado sus procesos internos de planificación de corto plazo y habiendo considerado los datos de la BPO y del plan de consumo enviado por el cliente, el proveedor inicia el protocolo informando al cliente el plan de aprovisionamiento. Este plan contiene los despachos futuros de materiales. Con esta información, el cliente valida el plan del proveedor contra sus propios datos y determina

si el inventario proyectado está dentro de los valores acordados. Si esto no ocurre, una excepción debe ser generada y resuelta. Esto es representado por el segmento con el operador *Exception*. La condición de excepción indica que cuando los límites mínimos y máximos del inventario no son satisfechos se debe ejecutar el camino contenido en este segmento. Dicho camino contiene una referencia a la plantilla de protocolo *Plan Exception Management*. Para su uso como protocolo de interacción es necesario asignar valor a los parámetros de la misma para que represente el proceso *Replenishment Exception*. Los valores de los parámetros son indicados en el comentario <<ProtocolParameters>> asociado al protocolo referenciado. En este caso se define que el rol iniciador será el cliente y el rol que responde será el proveedor. El documento de negocio que representa la notificación de la excepción es el *ResponseToReplenishmentSchedule*. El documento utilizado para dar una respuesta a la excepción es el *ReplenishmentSchedule*. Los detalles de cómo la excepción es resuelta son indicados por la definición de la plantilla *Plan Exception Management*, la cual se explica a continuación. Una vez finalizado este subprotocolo, el protocolo finaliza implícitamente con éxito.

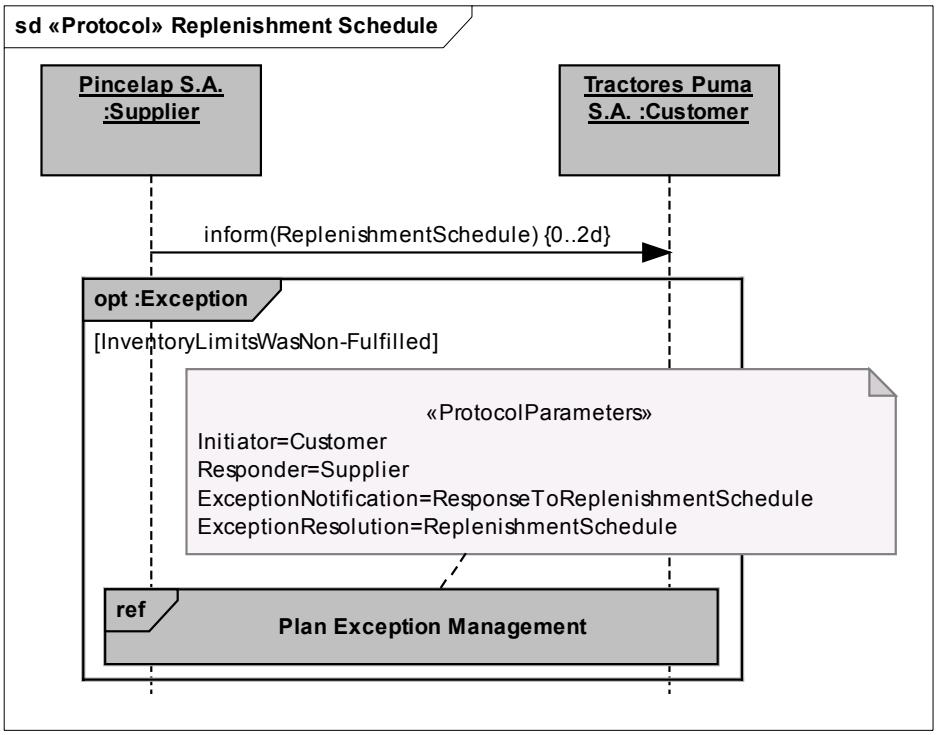


Figura 4-35. Diagrama de secuencia protocolo *Replenishment Schedule*

4.3.3.6 Plantilla de protocolo Plan Exception Management

En esta sección se describe la plantilla de protocolo *Plan Exception Management*, utilizada para materializar los procesos *Forecast Exception* y *Replenishment Exception* y gestionar las excepciones en los protocolos *Consumption Schedule* (Figura 4-33) y *Replenishment Schedule* (Figura 4-35). El diagrama de secuencia que describe esta plantilla de protocolo es mostrado en la Figura 4-36. El objetivo es reutilizarla para definir protocolos de interacción que administren de igual manera todo tipo de excepciones que puedan ocurrir sobre información de planificación, como ha sido utilizada anteriormente sobre el plan de consumo o el plan de aprovisionamiento. Debido a que este protocolo es una plantilla, los roles y el contenido de los mensajes no se corresponden con los definidos en los procesos colaborativos debido a que estos representan los parámetros de la plantilla. Estos parámetros toman valor cuando se define un protocolo en base a la plantilla, como se ha hecho en las figuras 4-33 y 4-35.

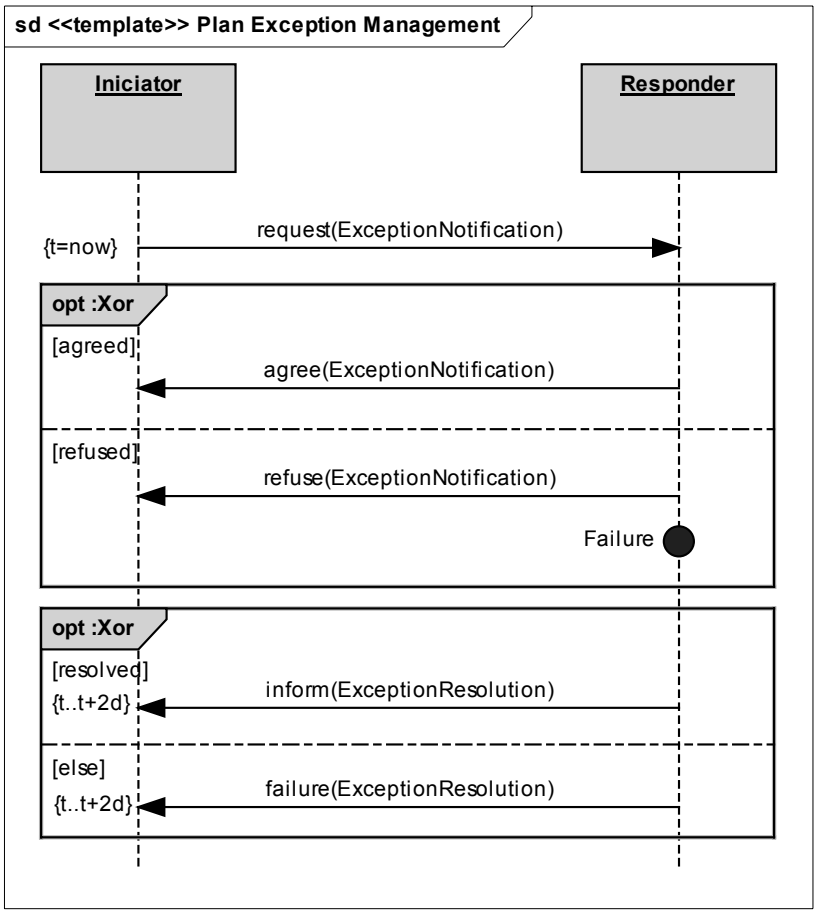


Figura 4-36. Diagrama de secuencia de la plantilla *Plan Exception Management*

Este protocolo comienza con el rol iniciador quien es el encargado de enviar un mensaje con la notificación de la excepción, solicitando su resolución. Esto es indicado por el acto de comunicación *request* asociado al mensaje. Luego, el rol receptor tiene dos alternativas: enviar un mensaje *agree(ExceptionNotification)*, representando el compromiso de éste a resolver la excepción en el futuro próximo; o enviar un mensaje *refuse(ExceptionNotification)* representando el rechazo a lo solicitado por el iniciador, lo cual significa la finalización del protocolo. Estas alternativas son expresadas en el primer segmento con el operador *Xor*.

Si la primer alternativa fue ejecutada, el protocolo continúa con el segundo segmento de control con el operador *Xor*. Este segmento representa dos alternativas que el rol que responde puede realizar para finalizar el protocolo de acuerdo a sus decisiones internas. Si este rol resuelve la excepción, es decir, puede cumplir con su compromiso asumido, informa los resultados al iniciador. De lo contrario, si el rol no pudo resolver la excepción, lo notifica al iniciador enviando el mensaje *failure(ExceptionNotification)*.

4.3.4. Definiendo la Vista de las Interfaces de Negocio de los Socios

Una vez definidos los protocolos de interacción que definen el comportamiento de los procesos, a partir de los mismos se derivan las interfaces de los roles involucrados en la colaboración B2B. Las interfaces definidas para los roles son mostradas en el diagrama de clases de la Figura 4-37.

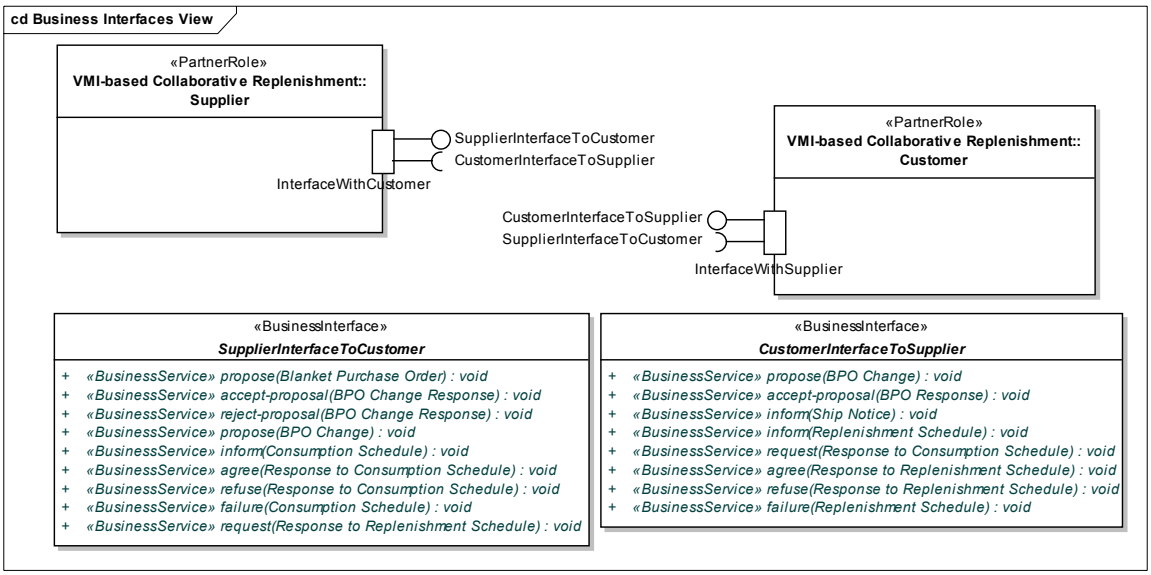


Figura 4-37. Diagrama de clases con las interfaces de negocio de los roles

Dos interfaces han sido definidas. La interfaz *SupplierInterfaceToCustomer* representa los servicios provistos por el rol proveedor al rol cliente. Esta interfaz está asociada al rol proveedor como una interfaz provista por el mismo. En el rol cliente, esta interfaz está asociada como una interfaz requerida por el mismo.

De manera similar, la interfaz *CustomerInterfaceToSupplier* representa los servicios provistos por el rol cliente al rol proveedor. Esta interfaz está asociada al rol cliente como la interfaz que éste provee al rol proveedor. En el rol proveedor, esta interfaz está asociada como la interfaz que éste rol requiere del cliente.

Los servicios de cada interfaz han sido definidos a partir de los mensajes que los roles envían y reciben en los protocolos de interacción definidos. Para el caso de los protocolos basados en plantillas, los roles a quien asignar los servicios identificados y los documentos a utilizar en cada servicio son obtenidos desde los parámetros de instanciación del protocolo.

4.4. Trabajos Relacionados

Un trabajo que extiende UML, el cual puede ser comparado con el lenguaje UP-ColBPIP propuesto en este capítulo, es UMM (UN/CEFACT Modelling Methodology) (UN/CEFACT, 2001). Presenta una metodología para el análisis y diseño de procesos de negocio colaborativos, y es el único que permite representar la vista global de los procesos colaborativos, definida como uno de los requerimientos para el modelado de dichos procesos. UMM provee un perfil UML, el cual también soporta un proceso de diseño top-down a través de la definición de varias vistas. En UMM, un proceso colaborativo es representado por una *colaboración de negocio*, la cual involucra uno o más socios de negocio y consiste de una coreografía de transacciones de negocio y colaboraciones de negocio anidadas. Una *transacción de negocio* es el concepto atómico dentro de una colaboración de negocio, y es el principal concepto para modelar procesos de negocio. Una transacción de negocio describe el intercambio de documentos de negocio también a través de una coreografía, la cual involucra solamente dos actividades, una de solicitud y otra de respuesta, llevadas a cabo por dos roles desempeñados por diferentes socios de negocio. Para definir las transacciones, UMM provee un conjunto de patrones de transacciones de negocio.

En UMM, el modelado de procesos colaborativos es realizado en forma jerárquica. Primero se define la coreografía de las colaboraciones de negocio y luego la coreografía de cada una de las transacciones de negocio involucradas en dichas colaboraciones de negocio. La consecuencia de este enfoque jerárquico es que tanto las interacciones entre las partes como las responsabilidades de las mismas no son definidas en la colaboración de negocio, sino que son definidas en cada una de las transacciones de negocio. Por lo tanto, estas propiedades de un proceso colaborativo no son expresadas en un alto nivel de abstracción cuando se describe el proceso completo a través de una colaboración de negocio. En cambio, en UP-ColBPIP, los procesos colaborativos basados en protocolos de interacción son definidos en términos de los mensajes de negocio intercambiados entre los roles, los cuales constituyen el concepto atómico y principal de un protocolo. Los mensajes de negocio de un protocolo no necesitan ser descompuestos en otros conceptos. De esta manera, en un protocolo de interacción es posible visualizar todas las interacciones y responsabilidades de los roles, sin depender de otra vista para el entendimiento de las mismas.

Otra desventaja importante de UMM es que los patrones sugeridos para definir transacciones de negocio no permiten dar soporte a negociaciones complejas. Por ejemplo, el patrón *Comercial (Oferta-Aceptación)* permite describir una negociación en donde una oferta sólo puede ser aceptada o rechazada. Sin embargo, como se ha visto en algunos ejemplos, un proceso de negociación involucra más que la aceptación o rechazo de una oferta, es decir, pueden existir contrapropuestas ante un rechazo o aceptación. Por lo tanto, la definición de un proceso de negociación complejo no puede ser expresada en una misma transacción o colaboración de negocio sin un esfuerzo adicional. Este esfuerzo resulta en una complejidad mayor al momento de modelar procesos colaborativos que involucran procesos de negociación. En (Rebstock y Thun, 2003), este problema es resuelto en parte, indicando las diferentes alternativas de la negociación dentro de los documentos de negocio intercambiados en una transacción de negocio. No obstante, la desventaja es que la intención de los socios en los mensajes no está separada de la información que es intercambiada. Esto conduce a una pérdida de poder descriptivo dentro de los procesos, ya que para entender las negociaciones hay que entender la estructura de los documentos. En cambio, en UP-ColbPIP, los procesos de negociaciones pueden ser expresados completamente en un mismo protocolo de

interacción, en donde la intención de cada una de las partes en los mensajes está separada del documento de negocio transportado en el mensaje.

Por otra parte, la definición de la coreografía de las colaboraciones de negocio a través de diagramas de actividades de UML no es intuitiva y hace que las mismas sean difíciles de entender. En UML, una actividad describe una acción a ser ejecutada por un único rol. Esto es diferente a la semántica de una colaboración de negocio de UMM, en donde una actividad representa una transacción de negocio o una colaboración de negocio anidada, las cuales representan las acciones ejecutadas por dos o más roles. Esto provee confusión al modelado de procesos colaborativos a través de diagramas de actividades de UML. Además, la noción del tiempo tampoco es visualizada en estos diagramas que describen las colaboraciones y las transacciones de negocio. En cambio, la noción del tiempo sí puede ser expresada en UP-ColBPIP, a través de los diagramas de secuencia que describen los protocolos de interacción. Los diagramas de secuencia proveen suficiente expresividad gráfica para describir protocolos de interacción, facilitando la lectura y diseño de los mismos. Estos diagramas permiten que todos los elementos de un protocolo sean representados explícitamente. Además, permiten visualizar el orden de los mensajes y elementos de un protocolo en el tiempo, como así también las responsabilidades de cada uno de los roles.

Una propuesta que extiende UMM es el enfoque llamado *Unified Framework for Business Models and Process Models* (Bergholtz y otros, 2003). Este enfoque incorpora el concepto de acciones pragmáticas para definir patrones de transacciones de negocio. Dichas acciones representan actos de comunicación que pueden ser utilizados. De esta manera, este enfoque adiciona aspectos de comunicación para modelar transacciones de negocio. El uso de acciones pragmáticas permite un mejor entendimiento de las transacciones de negocio. No obstante, esta propuesta hereda las mismas desventajas de UMM que se han mencionado anteriormente. Además, al igual que otros enfoques basados en LAP, los actos de comunicación no son considerados el principal bloque de construcción; ellos son utilizados en un bajo nivel de abstracción, dentro de las transacciones de negocio. En cambio, en UP-ColBPIP, los mensajes basados en actos de comunicación son el bloque de construcción principal y atómico de los procesos colaborativos, lo cual da gran poder expresivo al lenguaje para definir procesos de negocio complejos.

Con respecto al modelado de protocolos de interacción, UP-ColBPIP puede ser visto como una especialización de AUML2 (Huget y Odell, 2004). Éste es un lenguaje cuya especificación aún está en progreso y el mismo pretende ser un estándar para el modelado de sistemas multiagentes. AUML2 soporta el modelado de protocolos de interacción para sistemas multiagentes. No obstante, en UP-ColBPIP los protocolos de interacción han sido extendidos para proveer conceptos más específicos al dominio de procesos colaborativos. Por ejemplo, se han incluido operadores de flujo de control utilizados en lenguajes de workflows, como así también para el manejo de transacciones y de excepciones, tanto en cuanto a la lógica del proceso como así también excepciones de tiempo. Se han adicionado además conceptos más apropiados al dominio de procesos colaborativos, como los conceptos de segmento de flujo de control, mensajes de negocio y documentos de negocio. Además, en UP-ColBPIP, la vista de los protocolos de interacción está integrada con las restantes vistas, particulares al dominio de procesos colaborativos, con el objetivo de materializar estos procesos y derivar luego las interfaces de negocio requeridas por cada empresa para la implementación de los mismos.

4.5. Conclusiones

Como se ha destacado anteriormente, uno de los principales desafíos para el desarrollo de colaboraciones B2B es la necesidad de un lenguaje de modelado visual independiente de la tecnología para el diseño de procesos colaborativos. El lenguaje UP-ColBPIP definido en este capítulo es una solución hacia esa dirección. Este lenguaje provee un alto nivel de abstracción para el diseño de estos procesos, dejando de lado la definición de los detalles técnicos requeridos para la implementación de los mismos.

UP-ColBPIP soporta la definición de cuatro vistas de los procesos colaborativos involucrados en una colaboración B2B. Cada vista ofrece una perspectiva diferente de los procesos. A través de la definición de las vistas, UP-ColBPIP da soporte a las etapas de análisis y diseño de los procesos colaborativos, como así también a la derivación, a partir de dichos procesos, de las interfaces de negocio que los socios deben implementar. Con UP-ColBPIP, el modelado conceptual de los procesos colaborativos sigue un proceso de diseño top-down, en donde cada vista es un refinamiento de la anterior. No obstante, UP-ColBPIP también podría ser aplicado a través de un proceso de diseño bottom-up, con el objetivo de analizar los procesos ya definidos en una

colaboración B2B de acuerdo al enfoque propuesto. En este caso, debería comenzarse preferentemente definiendo la vista de protocolos de interacción, para luego generar la vista de los procesos colaborativos y la vista de la colaboración B2B. Esto obligará a las partes a repensar sus procesos no sólo desde el punto de vista de las interacciones, sino también desde el punto de vista de las metas que cada proceso debe alcanzar y cómo éstas deberían ser medidas. Finalmente, la derivación de las interfaces de negocio desde los protocolos debería indicar si las interfaces de sistemas ya existentes pueden satisfacer lo definido en los protocolos.

La base teórica principal de UP-ColBPIP es la aplicación del enfoque de diseño de procesos colaborativos basados en protocolos de interacción. Por un lado, este enfoque posibilita satisfacer los diferentes requerimientos identificados con respecto al modelado de procesos colaborativos. Por otro lado, con este enfoque, UP-ColBPIP puede ser considerado como un tipo de lenguaje de procesos colaborativos orientado a la comunicación. A través de la definición de protocolos de interacción, los diseñadores se enfocan en modelar no sólo el intercambio de información, sino también los aspectos de comunicación requeridos en los procesos colaborativos, permitiendo un mejor entendimiento de las dinámicas de las colaboraciones B2B. El uso de actos de comunicación para describir los mensajes o interacciones entre las partes posibilita expresar las intenciones de las partes en un proceso colaborativo. De esta manera, un proceso puede ser entendido como un conjunto de interacciones entre dos o más partes, a través de las cuales las partes crean, modifican, cancelan o cumplimentan los compromisos. Estos compromisos son derivados a partir de los actos de comunicación, de acuerdo a la semántica de los mismos.

El uso de actos de comunicación para describir las interacciones entre las partes simplifica el diseño de los procesos colaborativos. Los diseñadores pueden utilizar conceptos intuitivos más cercanos al lenguaje natural, pudiendo hacer analogías con las interacciones sociales entre humanos. Un punto importante a tener en cuenta es que, para que las partes entiendan de igual manera un proceso colaborativo basado en protocolos de interacción, la semántica de los actos de comunicación debe ser también entendida de igual manera por ambas partes. El entendimiento de la semántica del acto de comunicación es importante dado que cuando una empresa recibe un mensaje de negocio, debe procesarlo y responder en forma consistente con la semántica del acto de comunicación asociado al mensaje. Además, el conjunto de actos de comunicación

posibles a utilizar también debería ser conocido por las partes. Por lo tanto, el uso de una librería que provea de actos de comunicación bien definidos es importante, y permite que los protocolos puedan ser reutilizados. Como ejemplo, en esta tesis hemos utilizado la librería de actos de comunicación de FIPA, tanto para los protocolos de ejemplos como para los protocolos definidos en el caso de estudio.

Para la definición del lenguaje, se han seguido los lineamientos propuestos por MDA (OMG, 2003a) para diseñar lenguajes independientes de las plataformas. En particular, UP-ColBPIP ha sido definido como un perfil UML, el cual está basado en la nueva versión del estándar UML. En este perfil se ha reutilizado la mayor parte de la notación de UML2, con el propósito de ofrecer a los analistas de negocio y desarrolladores de sistemas una notación intuitiva para modelar procesos colaborativos. Esto permite a los usuarios del lenguaje acelerar el tiempo de aprendizaje del mismo, debido a que UML es extensamente conocido tanto en el dominio de sistemas de información como en el de modelado de procesos de negocio. Además, el uso de un perfil UML para modelar procesos colaborativos permite: proveer un vocabulario más adecuado que el original de UML para modelar dichos procesos; adicionar semántica y restricciones al metamodelo de UML desde el dominio de modelado de procesos colaborativos; y reutilizar las herramientas Case UML existentes para modelar estos procesos, sin la necesidad de contar con una herramienta específica para ello.

Otra ventaja de definir el lenguaje UP-ColBPIP como un perfil UML, es que el mismo también puede ser extendido y personalizado por los usuarios finales. Esto puede ser de utilidad en aquellos casos en donde varias empresas se enfocan en un tipo específico de cadena de suministro, en el cual es necesario capturar otros conceptos de ese dominio. Dichos conceptos pueden ser adicionados al lenguaje siempre que se respete la semántica del mismo.

Se han detectado algunas dificultades en el uso del perfil con las herramientas Case UML existentes en el mercado. Por ejemplo, algunas herramientas no permiten extender todos los conceptos de UML o bien no soportan la nueva versión del estándar. Más aún, existe una carencia en el soporte para definir y chequear restricciones definidas con OCL (OMG, 2003e). Sólo muy pocas herramientas posibilitan esto. Con lo cual, algunas restricciones definidas en el perfil no pueden ser chequeadas automáticamente y deben ser los usuarios quienes tienen que preocuparse en respetar dichas restricciones.

Además, con un perfil UML tampoco es posible automatizar algunas tareas de diseño, como por ejemplo la derivación automática de las interfaces de negocio desde los protocolos. Estas dificultades podrían ser resueltas si se implementa una herramienta específica para el lenguaje UP-ColPIP. En este caso, la herramienta podría ser desarrollada teniendo en cuenta la sintaxis y semántica del metamodelo que se ha definido para el lenguaje.

Este metamodelo también puede ser utilizado como una ayuda de referencia para entender el perfil definido, es decir, para entender la semántica y restricciones de cada uno de los conceptos y de las relaciones entre los mismos. Esto evita que los usuarios del perfil deban conocer los detalles de la sintaxis del metamodelo de UML para entender el lenguaje UP-ColBPIP.

Capítulo

5

MÉTODO Y HERRAMIENTA PARA TRANSFORMACIONES DE MODELOS

En este capítulo, en primer lugar se describen los conceptos, las principales características a ser tenidas en cuenta en un método y los diferentes enfoques propuestos, en el dominio de transformaciones de modelos. Luego se discuten los requerimientos particulares y se establecen los objetivos para llevar a cabo transformaciones de modelos de procesos colaborativos. Posteriormente, se describen el método y la herramienta propuestos para dar soporte a la definición y ejecución de dichas transformaciones. Finalmente, se presentan las conclusiones.

5.1. Transformaciones de Modelos

5.1.1. Conceptos de Transformación de Modelos

Según se describió en el capítulo 3, en el desarrollo conducido por modelos, las transformaciones de modelos son tan importantes como la especificación de los modelos que describen a un sistema desde diferentes niveles de abstracción. Los métodos de transformación constituyen el soporte principal para transformar los modelos independientes de la plataforma (MIPs) en modelos específicos de la plataforma (MEPs), y a partir de estos últimos generar el código correspondiente (Sendall y Kozaczynski, 2003). Las transformaciones de modelos posibilitan mantener sincronizados a los modelos conceptuales, con los modelos específicos de la solución, y finalmente a los anteriores con la implementación. Esta sincronización permite asegurar que la solución generada en el espacio tecnológico (la implementación) sea consistente con lo que se ha definido en el espacio del dominio del problema.

Debido a que el dominio de transformaciones de modelos es relativamente nuevo, no existe un consenso acerca del significado de los principales términos o conceptos utilizados. Para clarificar estos conceptos, en esta tesis se utilizan las definiciones dadas en el marco de la iniciativa MDA. Siguiendo la filosofía propuesta por MDA para el

desarrollo conducido por modelos (Kleppe y otros, 2003), una *transformación de modelos* es la generación automática de un modelo (modelo de destino o de salida) a partir de otro modelo (modelo de origen o de entrada). Una *definición de transformación* es un conjunto de reglas que describen cómo un modelo de origen basado en un lenguaje (denominado el lenguaje de origen), puede ser transformado en un modelo de destino basado en otro lenguaje (denominado el lenguaje de destino). Una *regla de transformación* es una descripción de cómo uno o más conceptos en el lenguaje de origen pueden ser transformados en uno o más conceptos en el lenguaje de destino. El *lenguaje de definición de transformaciones* es el lenguaje utilizado para especificar transformaciones de modelos. Una transformación de modelos también podría ser aplicable a múltiples modelos de origen y/o múltiples modelos de destino.

El patrón básico de transformación de modelos de MDA puede ser generalizado como se muestra en la Figura 5-1 (OMG, 2003a). El *transformador* es la herramienta o aplicación que soporta la ejecución automática de las transformaciones de modelos. Éste toma como entrada un modelo de origen definido en un lenguaje de origen determinado, aplica las reglas de transformación especificadas en la definición de la transformación, y finalmente produce la salida, es decir el modelo de destino basado en un lenguaje de destino determinado.

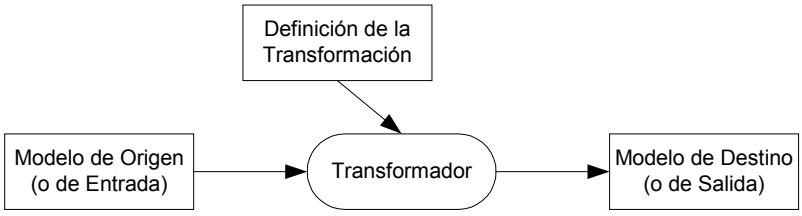


Figura 5-1. El patrón genérico de MDA

Estos conceptos de transformación de modelos están definidos desde el punto de vista de MDA, en donde el principal objetivo es transformar MIPs a MEPs. En estos casos, los lenguajes de los modelos de origen y destino son diferentes. Dichos conceptos también son aplicables a otros escenarios. Por ejemplo, se pueden utilizar en varias actividades de un proceso de desarrollo de software (Sendall y Kozaczynski, 2003) para: el refinamiento de modelos, realizar ingeniería inversa, generar nuevas vistas desde modelos existentes, aplicar patrones de diseño y arquitectónicos, y reorganizar modelos en donde la estructura interna de un modelo es cambiada pero se preserva su comportamiento externo. En algunos de estos escenarios, los modelos de origen y de

destino están basados en el mismo lenguaje. En estos casos, la transformación implica la modificación o actualización de un modelo. Esto último se conoce como transformación o sincronización horizontal, en donde los modelos de origen y de destino residen en el mismo nivel de abstracción o plataforma (Mens y otros, 2004). El otro tipo de transformación, como la transformación de un MIP a un MEP, se conoce como transformación o sincronización vertical, en donde los modelos de origen y destino pertenecen a diferentes niveles de abstracción o se corresponden con diferentes plataformas.

En el contexto del desarrollo conducido por modelos basado en MDA, la técnica de metamodelación es muy importante. Un lenguaje de modelado visual es definido a través de un metamodelo que describe la sintaxis abstracta del lenguaje, en donde el metamodelo puede ser visto como el modelo del lenguaje de modelado (Mellor y otros, 2004). Por lo tanto, la definición y ejecución de una transformación de modelos requiere un claro entendimiento de la sintaxis abstracta (el metamodelo) y de la semántica de los modelos de origen y de destino (Sendall y Kozaczynski, 2003).

Los diferentes niveles de metamodelos, modelos e instancias de los mismos que intervienen en el modelado de un sistema son expresados a través de la arquitectura clásica de cuatro niveles definida por el OMG (Mellor y otros, 2004). A estos niveles se los denomina comúnmente con las iniciales M0, M1, M2 y M3 (Figura 5-2).

El nivel M0 modela el sistema real y sus elementos son las instancias que componen dicho sistema. Ejemplos de estos elementos son, el empleado “Juan” que pertenece al departamento de “Ventas”.

El nivel M1 se refiere a los modelos del sistema. Aquí se definen, por ejemplo, los conceptos de “Empleado” y “Departamento”, cada uno con sus correspondientes atributos. Los conceptos de este nivel definen las clasificaciones de los elementos del nivel M0, mientras que los elementos del nivel M0 son las instancias de los elementos del nivel M1.

El nivel M2 refiere al modelo del modelo, es decir, el metamodelo que describe un lenguaje. Los elementos del nivel M2 son los conceptos de los lenguajes de modelado. Define los elementos conceptuales que intervienen a la hora de definir un modelo del nivel M1. En el caso de un modelo UML de un sistema, los conceptos propios del nivel M2 son “Clase”, “Atributo”, “Asociación”, etc. Luego, los elementos del nivel M2

definen las clases de los elementos válidos en un determinado modelo de nivel M1, mientras que los elementos del nivel M1 pueden ser considerados como instancias de los elementos del nivel M2.

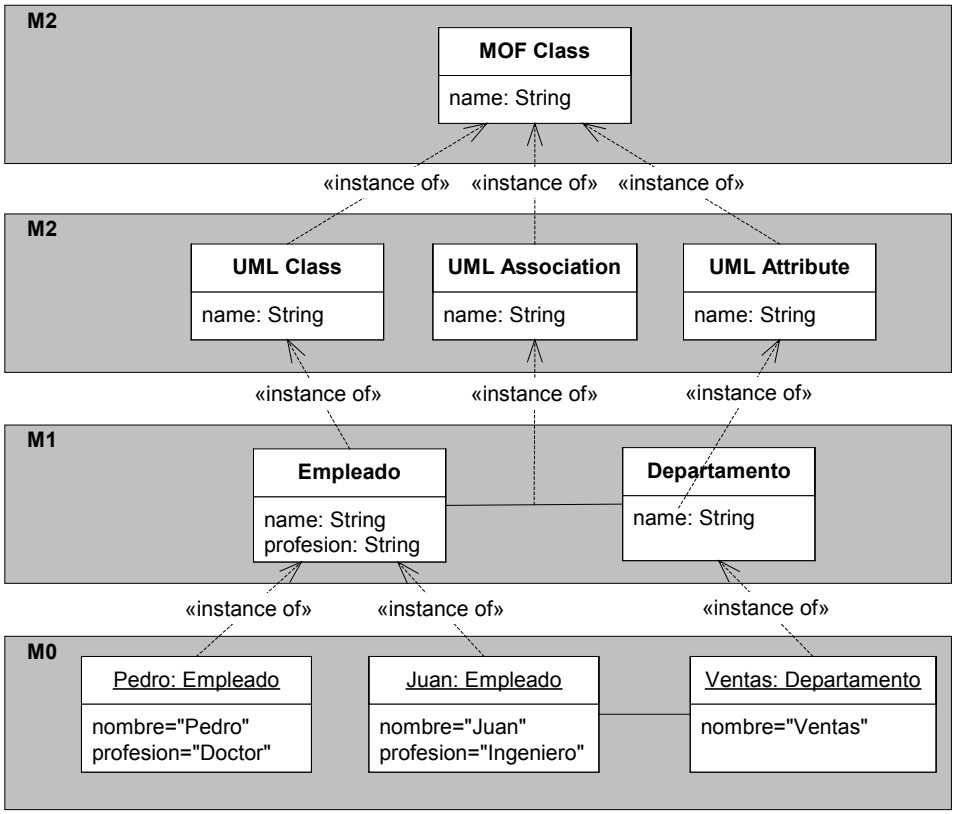


Figura 5-2. Un ejemplo de la jerarquía de metamodelo de cuatro capas

Finalmente, el nivel M3 corresponde al modelo de M2, es decir, el meta-metamodelo. Refiere a los elementos utilizados para definir los metamodelos de distintos lenguajes de modelado. De esta forma, el concepto de "Clase" definido en UML (nivel M2) puede verse como una instancia del elemento "Clase" correspondiente al nivel M3. En este nivel, el lenguaje MOF (Meta-Object Facility) (OMG, 2003c) ha sido definido para proveer los elementos del nivel M3. MOF puede considerarse como un lenguaje para describir lenguajes de modelado. Por ejemplo, UML es una instancia de MOF, debido a que el metamodelo de UML está definido con MOF.

Para llevar a cabo transformaciones de modelos, es necesario tener en cuenta los niveles de modelos, metamodelos y meta-metamodelos. Una transformación puede ser especificada de acuerdo a los elementos de los metamodelos. Por lo tanto, la técnica de metamodelación provee las bases tanto para describir modelos y definir lenguajes de modelado, como así también para definir transformaciones de modelos.

5.1.2. Características de un Método de Transformación de Modelos

Existe un conjunto de características generales a ser tenidas en cuenta en un método de transformación de modelos (Czarnecki y Helse, 2003), las cuales son mostradas en la Figura 5-3.

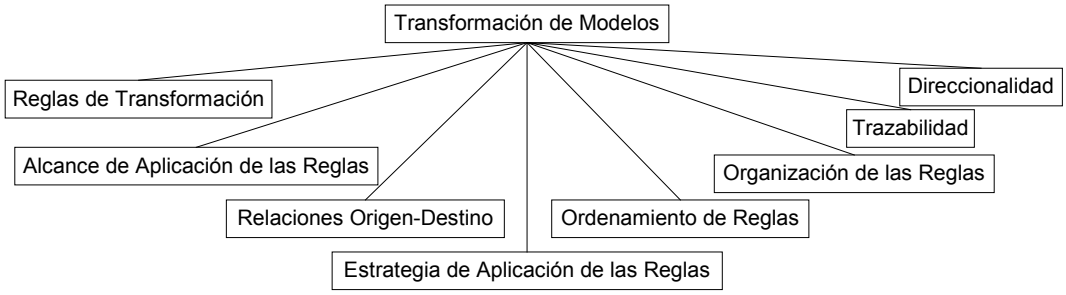


Figura 5-3. Características de un método de transformación de modelos

En primer lugar, toda transformación de modelos debería poseer *reglas de transformación*. Una regla debe consistir de dos partes: un lado izquierdo denominado LHS (Left-Hand Side) y un lado derecho denominado RHS (Right-Hand Side). El LHS accede al modelo de origen con el objetivo de encontrar un patrón de objetos dentro del mismo, mientras el RHS es utilizado para crear un patrón de objetos en el modelo de destino. Tanto el LHS como el RHS pueden ser representados usando una combinación de las siguientes características: variables, patrones y lógica. Las *Variables* mantienen los valores de los objetos de los modelos de origen y/o destino, como así también los valores de los atributos de los objetos o algún otro elemento. Los *Patrones* representan fragmentos de los modelos, los cuales pueden utilizar cero o más variables. Es posible especificar a los patrones utilizando una notación gráfica o textual. Independientemente de la notación, los patrones pueden ser representados usando la sintaxis abstracta o la sintaxis concreta de los lenguajes. La *Lógica* se refiere a la manipulación de los elementos de los modelos y las restricciones sobre los mismos. La lógica asociada a una regla podría ser ejecutable o no ejecutable. La lógica ejecutable puede tomar una forma declarativa o imperativa. Ejemplo de una lógica ejecutable, declarativa y textual es el uso de sentencias definidas en un lenguaje como OCL. La lógica imperativa tiene frecuentemente la forma de código en un lenguaje de programación, en donde se utiliza una API (Application Program Interface) para la manipulación directa de los modelos.

Otros aspectos que una regla puede poseer son: bidireccionalidad, implicando que puede ser ejecutada en ambas direcciones; y parametrización, implicando la utilización

de parámetros en una regla. Esto último está relacionado al *alcance de aplicación de las reglas*. A través de la definición de parámetros en una regla, es posible restringir las partes de un modelo que participan en la transformación.

Dentro del alcance de una regla, pueden existir varias ocurrencias de un patrón en el modelo de origen. Por lo tanto, es necesario indicar cómo la regla es aplicada para encontrar dichas ocurrencias en el modelo. Para ello, la *estrategia de aplicación de la regla* puede ser determinista o no determinista.

Con respecto a las *relaciones entre el origen y el destino*, un método puede crear un nuevo modelo separado del modelo de origen, o bien el modelo de origen y destino puede ser el mismo.

Con respecto al *ordenamiento de las reglas*, éste puede ser implícito o explícito. Un ordenamiento implícito implica que el usuario no tiene control sobre el algoritmo de ordenamiento, el cual está incorporado en la herramienta de transformación. Un ordenamiento explícito podría ser interno o externo. El primero implica que debería haber un mecanismo que permita a una regla invocar directamente otras reglas. En cambio, el ordenamiento externo implica una clara separación entre las reglas y la lógica del ordenamiento de las mismas. Utilizando un ordenamiento externo, el ordenamiento de reglas puede involucrar mecanismos de iteración y de recursividad, y el proceso de transformación podría ser organizado en varias fases o etapas.

La *organización de las reglas* se refiere a la composición y estructuración de las mismas. Los mecanismos posibles a utilizar aquí son: mecanismos de modularidad, que implica la agrupación de un conjunto de reglas en módulos; y mecanismos de reuso que permitan definir a una regla basada en una o más reglas.

Una transformación de modelos también podría registrar los vínculos (*trazabilidad*) entre los elementos del modelo de origen y aquellos del modelo de destino generados a partir de los primeros. A través de estos vínculos es posible analizar y depurar transformaciones, como así también sincronizar los modelos.

Finalmente, las transformaciones pueden ser unidireccionales o bidireccionales (*direccionalidad*). Las primeras pueden ser ejecutadas en una sola dirección, en donde el modelo de destino es generado en base al modelo de origen. Las segundas pueden ser ejecutadas en ambas direcciones, lo cual es útil para sincronización de modelos en

ambos sentidos. No obstante, estas últimas son difíciles de implementar. Las transformaciones bidireccionales pueden ser realizadas usando reglas bidireccionales o definiendo dos reglas unidireccionales separadas, una para cada dirección.

5.1.3. Enfoques de Transformación de Modelos

Actualmente no existe un lenguaje estándar para realizar transformaciones de modelos. Para soportar esta carencia dentro del contexto de MDA, el OMG está conduciendo un proceso de estandarización para tal tipo de lenguaje, el cual se conoce como QVT (Query/Views/Transformations) (OMG, 2003f). No obstante, el mismo está en sus etapas iniciales. Mientras tanto, para poder hacer realidad el desarrollo conducido por modelos, es necesaria la aplicación de algún método de transformación de modelos. Existen diferentes métodos propuestos, los cuales, de acuerdo a las técnicas utilizadas en los mismos, pueden ser clasificados (Czarnecki y Helse, 2003) en: *enfoques de manipulación directa, enfoques de transformaciones basadas en XML*, *enfoques de transformaciones de grafos* (también denominados enfoques operacionales), *enfoques relacionales* y *enfoques híbridos*. Estos últimos se corresponden con aquellos métodos que combinan dos o más de los restantes enfoques.

Los enfoques de manipulación directa se refieren a aquellos métodos en donde se proveen APIs para manipular los modelos de origen y de destino, como así también para definir las transformaciones. Una ventaja de este enfoque es que los desarrolladores utilizan lenguajes de programación bien conocidos, tales como Java o C, a través de los cuales es posible definir algoritmos de transformación complejos. No obstante, estos lenguajes carecen de abstracciones de alto nivel adecuadas para especificar transformaciones, y la definición de las mismas puede consumir demasiado tiempo y pueden ser difícil de entender y mantener.

Los enfoques de transformaciones basadas en XML utilizan XSLT (XSL Trasnformation) (W3C, 1999), el cual es un lenguaje estándar para la especificación de transformaciones de documentos XML. Debido a que los modelos basados en UML y MOF pueden ser almacenados como documentos XML usando el estándar XMI (XML Metadata Interchange) (OMG, 2003d), la implementación de transformaciones usando XSLT parece ser un enfoque atractivo. No obstante, este enfoque tiene varios inconvenientes. Principalmente, las transformaciones definidas con XSLT son generalmente muy extensas y difíciles de leer, y no proveen una noción abstracta y de

alto nivel de las transformaciones. Además es difícil separar las partes que tratan con el modelo de origen de aquellas partes que tratan con el modelo de salida. Algunos enfoques ocultan las definiciones de transformaciones en XSLT, utilizando un lenguaje de más alto nivel para generar las reglas XSLT. No obstante, estos enfoques también poseen varias desventajas (Czarnecki y Helse, 2003).

Otra categoría de enfoques son aquellos basados en la teoría de gramáticas de grafos y transformación de grafos. El concepto de transformación de grafos ha sido un candidato tradicional para especificar transformaciones en una manera operacional, debido a que los modelos visuales, como los diagramas de UML, pueden ser considerados como una forma especial de grafos. Por lo tanto, enfoques basados en transformaciones de grafos operan sobre grafos etiquetados, con atributos y tipos (Ehrig y otros, 1999), los cuales son un tipo de grafos especialmente diseñados para representar modelos UML. Ejemplos de estos enfoques son VIATRA (Varro y otros, 2002), GreAT (Agrawal y otros, 2003) Gmorph (Sendall, 2003) y AToM3 (De Lara y Vangheluwe, 2002). Estos enfoques son también conocidos como enfoques operacionales, ya que los mismos describen el proceso de transformación, es decir, especifican cómo los elementos de un lenguaje son transformados en elementos de otro lenguaje. La ventaja de estos enfoques es que tienen una base matemática sólida y existe una extensa teoría disponible para transformaciones de grafos (Ehrig y otros, 1999). Esto facilita el análisis de ciertas propiedades de las transformaciones de modelos, como terminación y confluencia.

La otra categoría de enfoques son aquellos conocidos como enfoques relacionales. En estos, una transformación de dos o más modelos es descrita estableciendo relaciones entre elementos del modelo de origen y elementos del modelo de destino, y especificando restricciones sobre dichas relaciones. A diferencia de los enfoques de transformación de grafos, los enfoques relacionales se basan en la declaración de un conjunto de relaciones entre los elementos de un lenguaje de origen y otro de destino, que especifican cuáles cambios suceden en una transformación, en lugar de especificar cómo estos cambios son realizados. Por lo tanto, estos enfoques ofrecen lenguajes declarativos. En algunos casos se provee una notación gráfica (QVTP, 2003) y en otros una notación textual (Akehurst y Kent, 2002). Estos métodos declarativos generalmente no son ejecutables. No obstante, algunos métodos proveen una semántica ejecutable para las definiciones, generalmente proveyendo un lenguaje basado en el paradigma de

programación lógica (Duddy y otros, 2003) o bien extendiendo OCL (Bezivin y otros, 2003).

Habiendo varios enfoques propuestos, aún no está claro cuál de ellos es más adecuado para ciertos tipos de aplicaciones, y además si será posible expresar todas las transformaciones de modelos posibles con un único enfoque (Küster y otros, 2004). A pesar de esta situación, los enfoques más utilizados son tanto los relacionales como los de transformación de grafos, debido a que proveen un nivel mucho más alto de abstracción para definir transformaciones.

Independientemente del enfoque utilizado, los métodos propuestos tienen limitaciones para ciertos tipos de transformaciones. Esto se debe a que los mismos han sido diseñados según las necesidades prácticas del dominio en el cual son aplicados, o bien, con el objetivo de dar respuesta a la solicitud de la OMG para definir el estándar QVT (Czarnecki y Helse, 2003). La consecuencia de esto es que en algunos escenarios de transformación de modelos, la aplicación de dichos métodos no es factible, aún cuando los mismos proveen mecanismos interesantes y poderosos para realizar las transformaciones.

Por ejemplo, VIATRA ha sido desarrollado para realizar transformaciones de modelos UML a lenguajes formales, con el objetivo de generar especificaciones para realizar la verificación formal de modelos UML. VIATRA ofrece un poderoso enfoque basado en transformaciones de grafos, en donde las reglas son definidas usando la sintaxis concreta de los lenguajes. AToM3 es otro método basado en transformaciones de grafos, en donde los patrones también son definidos teniendo en cuenta la sintaxis concreta de los lenguajes, al igual que en VIATRA. Aunque el uso de la sintaxis concreta tiene la ventaja de ser más familiar para los desarrolladores que trabajan con un lenguaje dado, ésta puede no ser adecuada en ciertas situaciones. En transformaciones en las cuales se manipulan modelos que representan estructuras de procesos, el uso de la sintaxis concreta podría no ser apropiado debido a la complejidad de la notación, lo cual hace difícil definir las reglas de transformación. En este caso, la definición de las reglas a través de la sintaxis abstracta de los lenguajes es más conveniente. Además, el uso de la sintaxis abstracta es adecuado cuando no existe una sintaxis concreta disponible (Czarnecki y Helse, 2003). Este es el caso de los modelos de procesos colaborativos específicos de la plataforma, los cuales están basados en un estándar B2B. Debido a que

los lenguajes provistos por los estándares B2B están basados en XML, es más conveniente generar el metamodelo de dicho lenguaje a partir del esquema XML del lenguaje, y especificar las transformaciones usando la sintaxis abstracta del lenguaje (elementos del metamodelo).

Con respecto a los enfoques relacionales, aunque varios métodos son interesantes (QVTP, 2003) (Akehurst y Kent, 2002), la mayoría de estos sólo proveen un lenguaje declarativo no ejecutable. Las transformaciones definidas en forma declarativa requieren ser traducidas a un lenguaje imperativo o bien a un lenguaje declarativo y ejecutable, para poder ejecutar transformaciones de modelos. Además, algunos enfoques relacionales utilizan un ordenamiento implícito de las reglas, cuando en algunos escenarios es más adecuado que los diseñadores de la transformación puedan determinar el orden de ejecución de las reglas en forma explícita.

Por otra parte, los métodos actuales también tienen limitaciones desde el punto de vista práctico para llevar a cabo transformaciones en diferentes dominios. Por ejemplo, las herramientas que soportan algunos de estos métodos no permiten la importación de modelos UML o modelos MOF en un formato XMI, para su posterior transformación. En su lugar, en algunos casos (como AToM3 y GreAT) se provee un ambiente de metamodelación y modelado, a través del cual es posible definir los lenguajes como así también los modelos. No obstante, tanto el lenguaje como el modelo deben ser definidos con la misma herramienta, lo cual tiene limitaciones en aquellos escenarios en donde los lenguajes son definidos como perfiles UML, con el objetivo de que los modelos puedan ser definidos con cualquier herramienta Case.

Otro problema práctico en la utilización de los métodos actuales es la generación de código. Algunos sólo soportan transformaciones de modelos a código. Otros sólo realizan transformaciones de modelos a modelos y no consideran la última etapa de generación de código. Más aún, existen diferentes tipos de código a ser generados. Aunque muchos de los métodos sólo tienen como objetivo generar código para aplicaciones basadas en la plataforma J2EE, existen otros escenarios en donde es necesario generar código XML para lenguajes basados en XML o código en algún lenguaje formal de verificación y validación.

Una solución, para tratar con estos y otros problemas relacionados a la carencia de generalidad de los actuales métodos de transformación de modelos, es expresar los

requerimientos particulares del dominio de transformación y definir o utilizar un método adecuado que satisfaga dichos requerimientos.

Por lo tanto, a continuación se describen los requerimientos particulares identificados para llevar a cabo transformaciones de modelos en el dominio de procesos colaborativos, con el objetivo de generar especificaciones de dichos procesos en un estándar B2B a partir de definiciones de los mismos con UP-ColBPIP.

5.2. Requerimientos y Objetivos

En primer lugar, es necesario soportar transformaciones en las cuales los modelos de entrada y salida estén basados en diferentes metamodelos. En el contexto de transformaciones de modelos de procesos colaborativos, los modelos de entrada serán aquellos modelos definidos con UP-ColBPIP. Los modelos de salida serán aquellos basados en un estándar B2B específico. Además, las transformaciones de modelos también podrían ser utilizadas para realizar el refinamiento de modelos UP-ColBPIP, como por ejemplo derivar la vista de interfaces de negocio a partir de la vista de protocolos de interacción. Para ello también es necesario soportar transformaciones en donde tanto los modelos de entrada como los de salida estén basados en el mismo metamodelo.

Otro aspecto a considerar es la forma en que los modelos, principalmente el modelo de entrada, serán almacenados para luego poder manipularlos. Una alternativa es la utilización de un repositorio común de objetos. Otra alternativa es almacenar los modelos en forma serializada en documentos XML utilizando el estándar XMI. Como se describió en el capítulo anterior, el lenguaje UP-ColBPIP ha sido implementado como un perfil UML, con el objetivo de que modelos de procesos colaborativos independientes de la tecnología puedan ser definidos con cualquier herramienta Case UML basada en UML2. Por lo tanto, la primera alternativa no es factible, debido a que cada herramienta Case UML define su propio repositorio de manera diferente. Para resolver esta dificultad, es necesario almacenar los modelos definidos con UP-ColBPIP en un formato XMI. Esto es factible debido a que la mayoría de las herramientas Case UML cuentan con funciones de importación y exportación de archivos XMI. Por lo tanto, la herramienta de transformación debería poder tomar un archivo XMI conteniendo un modelo definido con UP-ColBPIP, para luego realizar las

transformaciones y generar un modelo de salida en un formato XMI o directamente el código final en el formato XML definido por el estándar.

Esto último significa que la herramienta no sólo debería soportar transformaciones de modelo a modelo, sino también transformaciones de modelo a código XML. Para ello es necesaria la incorporación de reglas de producción específicas que permitan generar código XML a partir de un modelo específico de la plataforma.

Aunque podría generarse en forma directa el código XML a partir de un modelo definido con UP-ColBPIP, la utilización de modelos intermedios (los modelos específicos de la plataforma) permite disminuir la complejidad en el proceso de transformación, como así también permite definir transformaciones más modulares y más fáciles de mantener (Czarnecki y Helse, 2003).

Por otra parte, debido a que los estándares B2B aún están en evolución, no es adecuado realizar una implementación específica para cada transformación. En su lugar, se debería contar con un lenguaje que soporte la especificación de diferentes transformaciones de modelos, las cuales deberían poder ser interpretadas y ejecutadas en forma automática por una misma herramienta. En forma análoga a la gestión de procesos de negocio a través de workflows, en donde las especificaciones de diferentes procesos son ejecutadas por una misma máquina de workflow, diferentes especificaciones de transformaciones de modelos deberían poder ser ejecutadas por una misma máquina de transformación de modelos.

Otro punto a considerar es que los modelos de procesos de negocio pueden contener varios niveles de anidamiento, de acuerdo a los niveles de subprocesos que tengan. Particularmente, los protocolos de interacción en UP-ColBPIP no sólo tienen varios niveles de anidamiento de protocolos, sino que en un mismo protocolo, los segmentos de flujo de control también pueden tener varios niveles de anidamiento.

La Figura 5-4 muestra un ejemplo de un protocolo de interacción como una instancia del metamodelo de UP-ColBPIP, visualizándolo desde el punto de vista de su sintaxis abstracta en un diagrama de objetos. Este diagrama describe el modelo de un protocolo *A*. Éste contiene dos *lifelines* que representan los roles involucrados en el mismo. Además, contiene tres mensajes. Como elementos de primer nivel, el protocolo contiene: dos ocurrencias de eventos que representan el intercambio del mensaje *m1* entre los roles, un segmento de flujo de control, y una ocurrencia de interacción. Esta

última tiene una referencia al protocolo *B* y representa un anidamiento de protocolos. El segmento de control *CFS_A* tiene definido el operador *Xor* y contiene dos caminos de interacción. El primer camino contiene dos ocurrencias de eventos que representan el intercambio del mensaje *m2*. El segundo camino posee a su vez otro segmento de flujo de control, representando el anidamiento de segmentos. Este segmento contiene un único camino con dos ocurrencias de eventos, las cuales representan el intercambio del mensaje *m3*. La Figura 5-5 muestra la correspondiente sintaxis concreta del protocolo *A*.

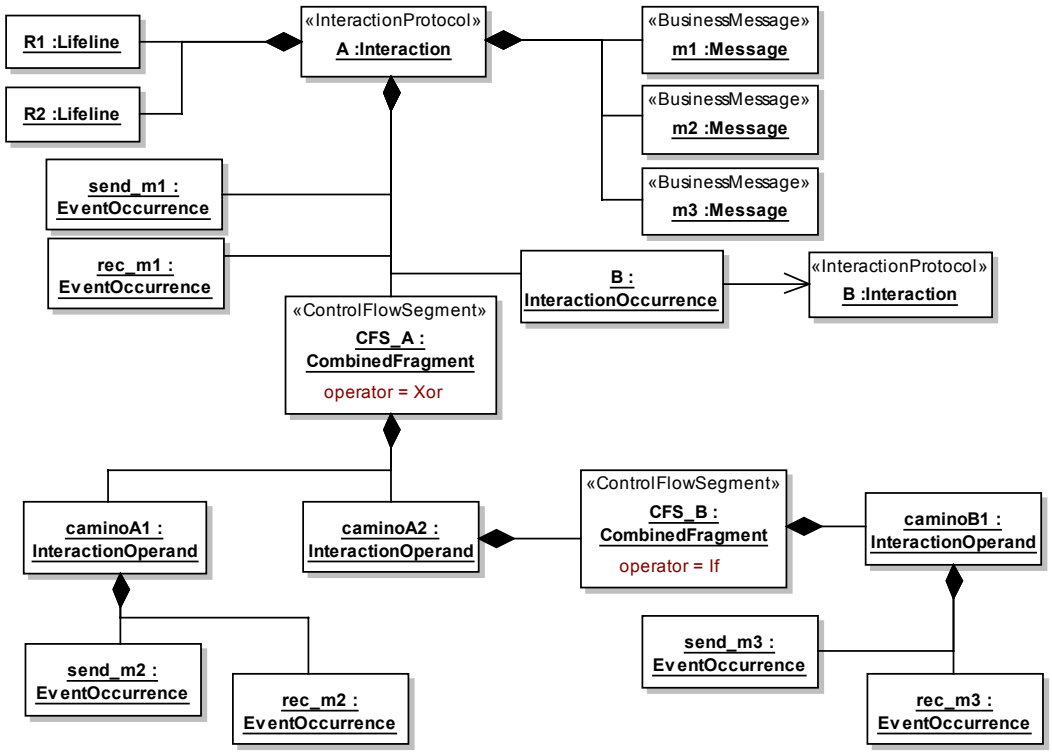


Figura 5-4. Instancia del metamodelo de UP-ColBPIP.

Observando la Figura 5-4, es posible visualizar los niveles de anidamientos que pueden existir en este tipo de modelos, los cuales no son posibles de determinar a priori cuando se define una transformación. Por lo tanto, recursividad y composición de reglas es un requerimiento a manejar en las transformaciones de este tipo de modelos, para tratar con tales niveles de anidamientos arbitrarios. Además, *"backtracking"* de reglas es un mecanismo poderoso y adecuado para manejar la recorrida de la estructura del flujo de control de un protocolo, principalmente cuando es necesario encontrar un camino reduciendo el espacio de búsqueda. Esto permite que la ejecución de las transformaciones sea más eficiente.

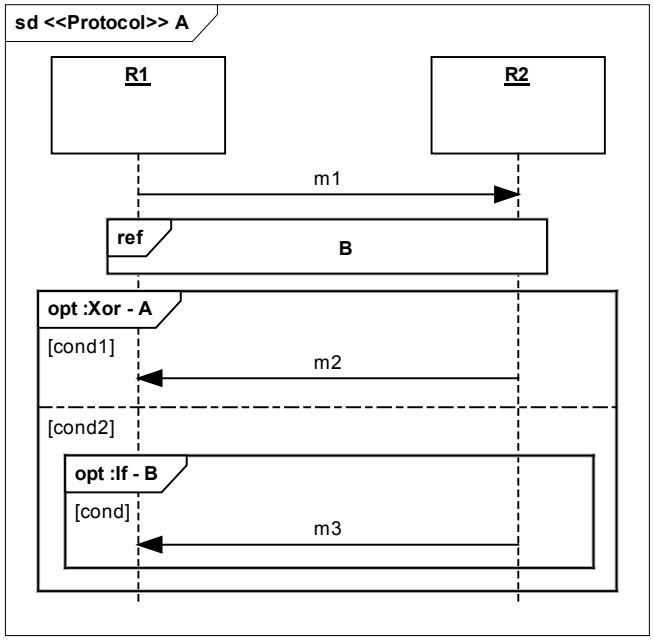


Figura 5-5. Sintaxis concreta del protocolo.

Por otro lado, separando la definición de las reglas de la estructura de ordenamiento de las mismas, también incrementa la reutilización de las reglas y permite al diseñador definir en forma explícita su ordenamiento. En modelos de protocolos de interacción, el ordenamiento explícito es requerido para indicar los caminos posibles a seguir en los diferentes puntos de una transformación de estos modelos. Esto también permite que la ejecución de la transformación sea más eficiente. Por ejemplo, la coreografía de un protocolo consiste de diferentes tipos de elementos, para los cuales no es posible determinar a priori el orden de los mismos. Por lo tanto, es necesario indicar el camino a seguir para transformar cada uno de los diferentes tipos de elementos.

Finalmente, para dar soporte a la composición y recursividad de reglas de transformaciones es necesario la utilización de parámetros de entrada y/o salida en las reglas, con el objetivo de posibilitar el pasaje de información entre las mismas. Además, el uso de parámetros ayuda a disminuir el espacio de búsqueda en los patrones de las reglas, ya que acota el ámbito de aplicación de las mismas.

Teniendo en cuenta lo expresado anteriormente, se propone un método y una herramienta de transformación de modelos, cuyo objetivo es:

- Hacer efectiva la definición, ejecución e implementación de transformaciones de modelos en el dominio de procesos colaborativos.

- Dar soporte a los requerimientos particulares identificados previamente para llevar a cabo transformaciones de modelos de procesos colaborativos.
- Considerar las principales características (mencionadas en la sección 5.1.2) que un método de transformación de modelos debería poseer.

En la siguiente sección se describe el método de transformación de modelos propuesto. La herramienta que soporta dicho método es descrita en la sección 5.4.

5.3. Método de Transformación de Modelos

El método de transformación de modelos propuesto consiste de los siguientes componentes:

- El **Lenguaje de Transformación de Modelos**. Este es el principal lenguaje, el cual ha sido implementado en la herramienta descrita en la sección 5.4, y a través del cual es posible definir y ejecutar transformaciones de modelos. Por un lado, este lenguaje permite declarar cada una de las reglas, con sus patrones, variables y parámetros. Por otro lado, el lenguaje provee los constructores necesarios para posibilitar la composición, recursividad y ordenamiento de las reglas. Además, el lenguaje es declarativo, ejecutable y visual. Aunque los patrones de las reglas también son especificados en forma declarativa con este lenguaje, la implementación de los mismos debe ser provista en forma separada. Para llevar a cabo dicha implementación y su integración a la herramienta que soporta el lenguaje de transformación de modelos, se provee una API en Java.
- El **Lenguaje de Modelado de Reglas**. Éste lenguaje es utilizado para especificar la estructura de las reglas, es decir, sus patrones. El propósito de este lenguaje es dar soporte a la definición, en una forma declarativa y visual, de la semántica operacional de los patrones que componen las reglas. Para ello, el mismo está definido como un perfil de UML y está basado en la teoría de gramáticas de grafos. Este lenguaje no pretende ser un lenguaje para la ejecución de reglas y patrones, sino sólo un lenguaje visual para la definición y entendimiento de los patrones de las mismas. Además, este lenguaje también tiene como propósito establecer la semántica operacional que debería ser seguida en la implementación de los patrones.

- Las **Reglas de Producción de Código XML**. Estas reglas son utilizadas para generar el código XML a partir de un modelo específico de la plataforma.

5.3.1. Lenguaje de Transformación de Modelos

El *Lenguaje de Transformación de Modelos* propuesto parte de la premisa que es más flexible y eficiente formular una transformación compleja en términos de un conjunto de reglas simples, las cuales luego son compuestas en un orden lógico para formar una transformación compleja, en lugar de definir unas pocas reglas complejas. La separación de la definición de las reglas, de la estructura de composición y ordenamiento de las mismas, posibilita una mayor reutilización de las reglas, como así también un mejor entendimiento y compresión del proceso de transformación. Esto también posibilita un proceso de transformación más robusto y manejable.

Además, definiendo reglas de transformación simples y sencillas permite establecer una relación conceptual y lógica entre los elementos que se están transformando y el significado de la regla. Esto se debe a que cada regla de transformación puede corresponderse con la transformación de un único o algunos conceptos del lenguaje de origen o destino, en lugar de expresar la transformación de muchos conceptos a la vez en donde las reglas son difíciles de entender y mantener.

Por lo tanto, en este lenguaje, una regla de transformación es definida como una regla simple, la cual posee parámetros de entrada y salida, variables privadas, patrones del lado izquierdo y patrones del lado derecho. La composición de las reglas es realizada dentro de módulos de reglas, los cuales contienen un árbol de ordenamiento de nodos que define la estructura de composición y ordenamiento de las reglas.

De esta manera se proveen dos niveles en la definición de transformaciones. Un nivel en el cual se describen todas las reglas simples, y otro nivel en donde se define la composición y ordenamiento de las mismas.

5.3.1.1. Metamodelo del Lenguaje de Transformación de Modelos

El lenguaje de transformación de modelos fue definido siguiendo los lineamientos propuestos por MDA y la OMG para definir lenguajes de modelado. En este caso, el lenguaje no ha sido definido como un perfil de UML, sino que su metamodelo ha sido definido usando MOF e implementado en una herramienta específica. A continuación se

describen los elementos conceptuales del metamodelo definido por el lenguaje, junto con las relaciones de los mismos. La Figura 5-6 muestra el metamodelo del lenguaje.

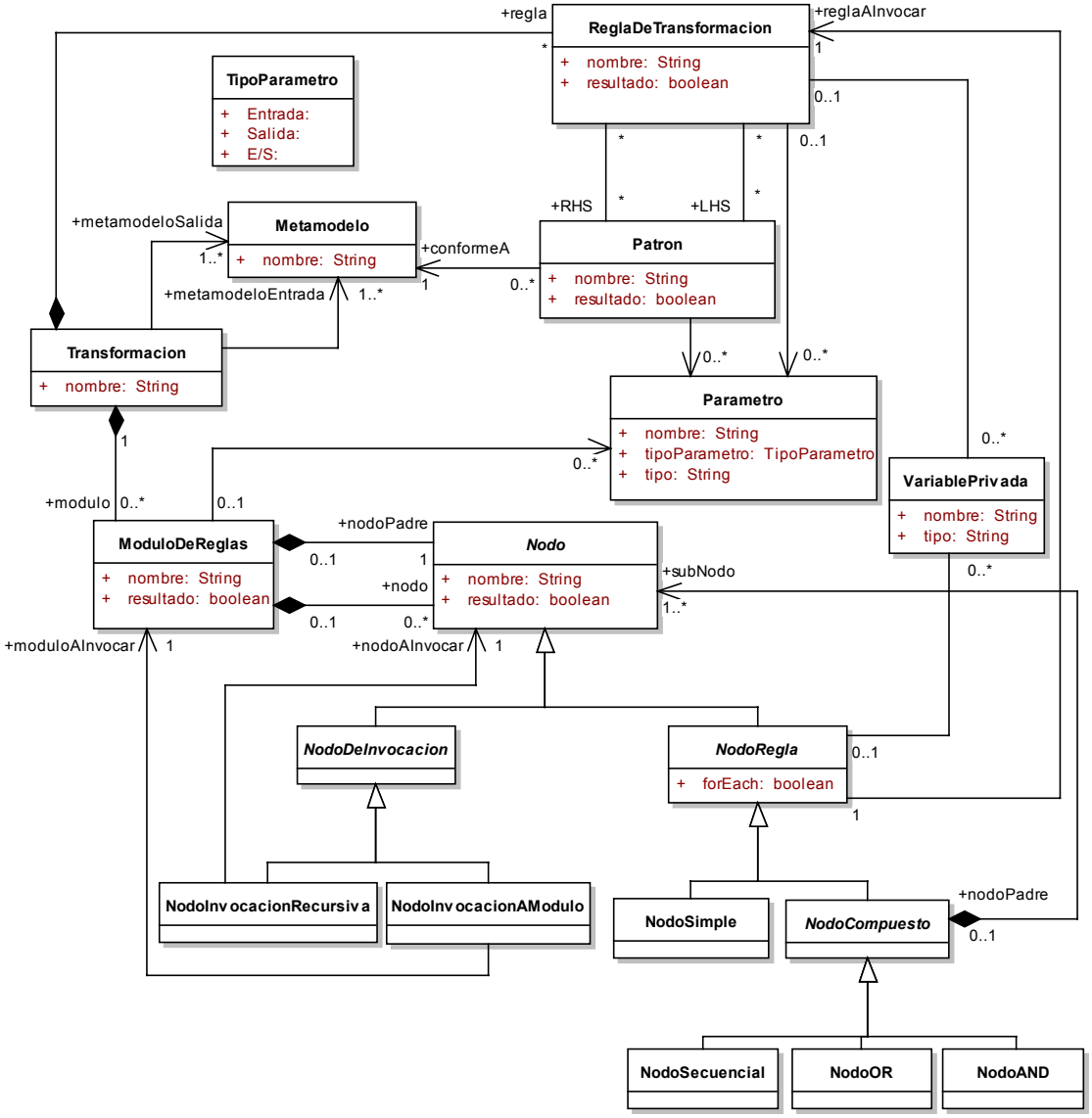


Figura 5-6. Metamodelo del lenguaje de transformación de modelos.

Una *Transformación* representa una definición o modelo de transformación, la cual determina cómo un modelo de origen o de entrada será transformado en un modelo de destino o de salida.

Para definir y ejecutar una transformación, es necesario especificar los *Metamodelos* de entrada y de salida, de tal forma de determinar los tipos de modelos de entrada y salida que pueden ser aceptados. Una transformación podría tener varios metamodelos de entrada y varios de salida, representando el escenario en el que existan

varios modelos de entrada definidos en lenguajes diferentes y/o varios modelos de salida definidos en lenguajes diferentes. Cuando los metamodelos de entrada y salida son diferentes, la transformación indica que los modelos de entrada y salida son diferentes y corresponden a diferentes lenguajes o dominios. Esto permite definir transformaciones verticales. Por ejemplo, la transformación de un modelo de procesos definidos con UP-ColBPIP a modelos de procesos basados en ebXML o BPEL. Cuando el metamodelo de entrada es el mismo que el metamodelo de salida, esto representa una transformación horizontal que implica el refinamiento o actualización de un modelo. Por ejemplo, la derivación de la vista de las interfaces de negocio, a partir de la vista de los protocolos de interacción, en un modelo UP-ColBPIP.

Una transformación está compuesta por un conjunto de reglas simples y por un conjunto de módulos de reglas.

Una *Regla de Transformación* es una especificación de una regla simple. Consiste de un conjunto de patrones: uno o varios patrones que corresponden al lado izquierdo de la regla (LHS), y uno o varios patrones que corresponden al lado derecho de la regla (RHS). La ejecución de la regla consiste en evaluar primero los patrones del lado izquierdo, y si los mismos retornan como resultado un valor verdadero, entonces se evalúan los patrones del lado derecho.

En el caso de transformaciones de un modelo basado en un lenguaje a otro modelo basado en otro lenguaje, los patrones del lado derecho de una regla siempre deben estar definidos de acuerdo a uno de los metamodelos de salida, es decir, deben operar sobre un modelo de salida. El patrón del lado derecho representa el fragmento que será generado en el modelo de salida.

Por el contrario, un patrón del lado izquierdo de una regla puede estar definido de acuerdo a un metamodelo de entrada como así también a un metamodelo de salida. En el primer caso, el patrón del lado izquierdo opera sobre un modelo de entrada con el objetivo de encontrar una ocurrencia del mismo en dicho modelo. En el otro caso, el patrón del lado izquierdo opera sobre el modelo de salida que se está generando para encontrar una ocurrencia del mismo en dicho modelo. Esto último es utilizado para representar dos situaciones: establecer condiciones de aplicación de la regla en base al estado del modelo de salida antes de generar el patrón del lado derecho; o para obtener información de los elementos del modelo de salida que se están generando con el

objetivo de generar el patrón del lado derecho en base a esta información.

Además, una regla puede tener sólo patrones del lado izquierdo o sólo patrones del lado derecho. No obstante, una regla siempre debe tener al menos un patrón.

Una regla de transformación puede poseer parámetros. Un *Parámetro* puede ser de entrada, de salida o de entrada/salida (E/S). Los parámetros de una regla son utilizados para pasar información a los patrones o para obtener información desde los patrones. Estos parámetros se corresponden con los parámetros solicitados por los patrones de la regla. Además, estos parámetros posibilitan el pasaje de información entre varias reglas, cuando las mismas son compuestas a través de nodos de reglas.

Dentro de una regla, también es posible manejar *Variables Privadas*. Estas son utilizadas para pasar información entre los patrones del lado izquierdo y los del lado derecho de la regla. Las mismas no generan entrada o salida a una regla. Una variable privada puede ser utilizada como argumento de salida en un patrón del lado izquierdo para obtener un valor, el cual luego será pasado a un patrón de lado derecho utilizando esta variable como argumento de entrada del mismo.

Tanto el tipo de los parámetros como el tipo de las variables privadas pueden corresponderse con uno de los tipos de datos básicos (string, entero, booleano o enumeración), o bien pueden corresponderse con una de las metaclases de los metamodelos referenciados en la transformación. De esta manera, a través de los parámetros y variables es posible pasar información (instancias de tipos de datos básicos u objetos de los modelos) entre reglas, entre una regla y sus patrones, y entre los patrones de una misma regla.

Un *Patrón* representa un fragmento o conjunto de objetos a encontrar o a generar en un modelo, en donde los tipos de los objetos se corresponden a alguna de las metaclases del metamodelo asociado al patrón. El metamodelo que un patrón tiene asociado debe ser uno de los metamodelos definidos en la transformación. Cuando el patrón forma parte del lado izquierdo de la regla, éste representa un fragmento a ser encontrado en un modelo. Cuando el patrón forma parte del lado derecho de la regla, éste representa el fragmento a ser generado en el mismo o en otro modelo, según si ambos patrones tienen asociados el mismo metamodelo o no.

Un patrón puede poseer parámetros de entrada, salida o entrada/salida. Los

parámetros de un patrón deben corresponderse con los parámetros de la regla o bien con las variables privadas de la regla en la cual está involucrado. Además, debido a que la implementación de los patrones es independiente de la regla, un patrón también puede ser reutilizado en varias reglas.

La ejecución de una regla de transformación implica un resultado, el cual puede ser *falso* o *verdadero*. La ejecución de un patrón también retorna un resultado *falso* o *verdadero*. El diagrama de actividades de la Figura 5-7 describe la semántica de evaluación de una regla de transformación. La evaluación de la regla comienza primero con la evaluación de los patrones del lado izquierdo y luego, los patrones del lado derecho. El orden en el que los patrones de cada lado son evaluados depende del orden en el que los mismos han sido definidos en cada lado. Una regla retorna falso cuando uno de sus patrones retorna falso. Esto ocurre cuando un patrón del lado izquierdo de la regla retornó falso porque no encontró el fragmento en un modelo, o bien cuando un patrón del lado derecho retornó falso como consecuencia de un fallo o excepción en su ejecución. En el caso contrario, cuando todos los patrones retornan un valor verdadero, la regla retorna verdadero. Esto último significa que la regla se ejecutó con éxito, es decir, que se encontró los fragmentos de los patrones del lado izquierdo en los modelos correspondientes, y además se generó con éxito los patrones del lado derecho en los modelos correspondientes.

En el caso que un patrón del lado derecho retorne falso, se generará una excepción, la cual significa que no se garantiza que se halla generado en forma consistente dicho patrón. Esto es utilizado para abortar el proceso de transformación.

Los conceptos de regla de transformación, patrón, parámetros y variables privadas son provistos para posibilitar la definición de reglas simples. No obstante, como se indicó anteriormente, estas reglas deberían estar compuestas en un orden lógico, con el objetivo de definir transformaciones complejas. Para ello, se provee el concepto de *Módulo de Reglas*. Una transformación puede poseer varios módulos de reglas.

Un *Módulo de Reglas* define una jerarquía de reglas a través de un árbol de reglas de transformación, el cual representa la composición y el ordenamiento de las reglas. El módulo de reglas inicial que desencadena la ejecución de la transformación debe ser seleccionado al momento de la ejecución de la transformación.

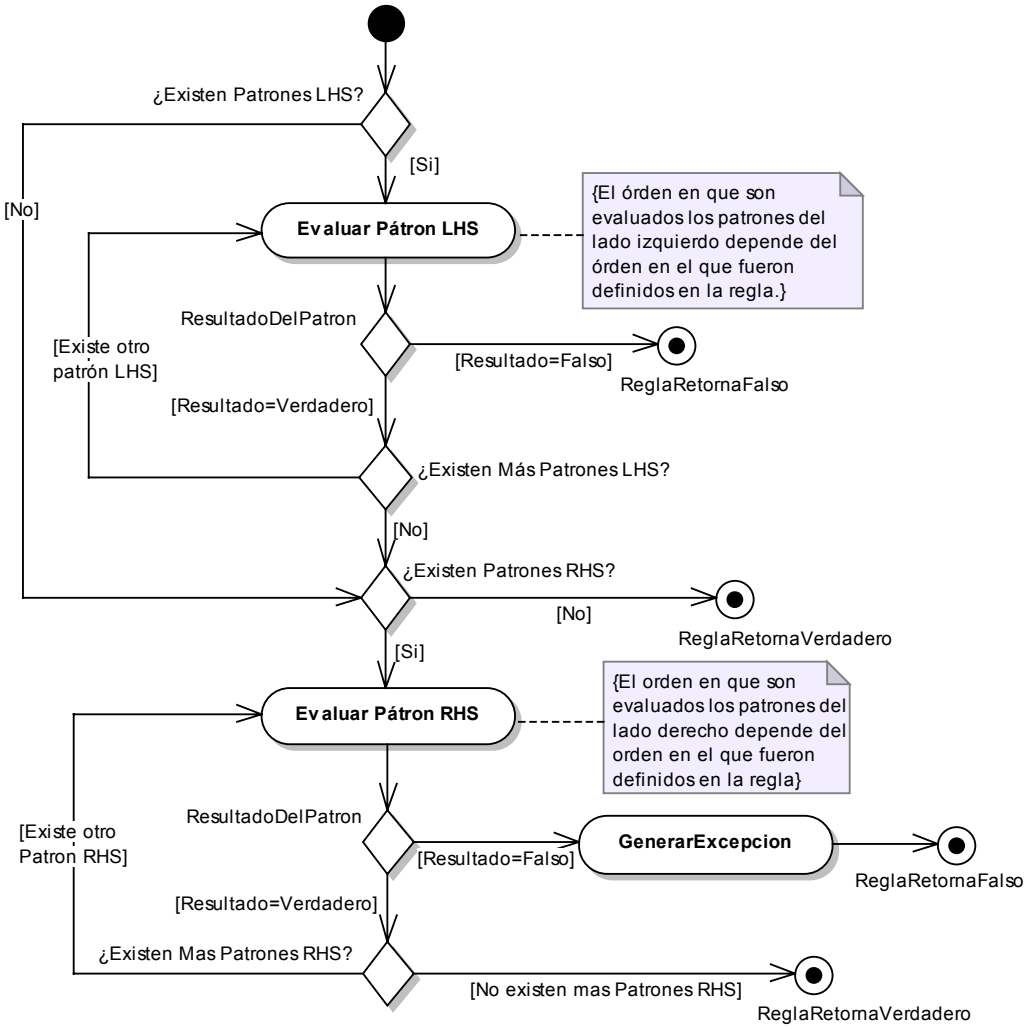


Figura 5-7. Semántica de evaluación de una regla de transformación

Un módulo de regla posee un único nodo raíz, del cual se desprende el árbol de reglas. Un módulo también retorna un valor verdadero o falso luego de su ejecución. El valor que puede retornar el módulo dependerá del valor que retorna la ejecución del nodo raíz. Las reglas son compuestas en módulos que tienen un significado lógico dado. La utilización de módulos de reglas permite que las reglas simples puedan ser reutilizadas en diferentes módulos de regla.

Además, al igual que una regla, un módulo de reglas puede poseer parámetros de entrada y/o salida, los cuales sirven para pasar información al módulo, como así también obtener información desde éste después de su ejecución.

Un *Nodo* de un módulo de regla representa la evaluación de una regla o la invocación a otros módulos o nodos (según el subtipo de nodo). Por lo tanto, todo nodo puede retornar un valor falso o verdadero como resultado de su ejecución.

Un *Nodo Regla* representa la invocación de una regla de transformación, la cual está asociada a dicho nodo. Esto implica que al momento de ejecutarse un nodo regla, éste realizará la evaluación de la regla asociada. Sólo una regla puede estar asociada a un nodo regla. El resultado del nodo dependerá en primer lugar del resultado de la evaluación de la regla. Para el caso de nodos compuestos, el resultado del nodo también dependerá de la semántica de los subtipos de nodos compuestos. A través de este tipo de nodo, las reglas pueden estar asociadas a diferentes nodos dentro de un árbol de un módulo de reglas, como así también en nodos de otros módulos de reglas.

Al igual que una regla de transformación puede contener variables para pasar información entre los patrones de la regla, un nodo regla también puede contener variables con el objetivo de pasar información entre las reglas asociadas a los nodos del árbol. Estas variables son utilizadas como argumentos en la llamada a la regla de transformación asociada a cada nodo, para pasar un valor a un parámetro de entrada de una regla o para obtener un valor desde un parámetro de salida de la regla. Además, las variables privadas de un nodo regla pueden ser utilizadas como argumentos en las reglas de transformación asociadas a cualquiera de sus subnodos, para el caso de nodos compuestos.

Todo nodo regla puede ser ejecutado una o varias veces mientras la evaluación de su regla asociada retorne verdadero. Esto es definido por el atributo *forEach*. Si este atributo posee un valor *falso*, entonces la regla asociada al nodo se evalúa una única vez. Si el atributo *forEach* posee un valor *verdadero*, entonces este nodo de regla podrá ser ejecutado *N* veces. El nodo de regla se ejecutará mientras su regla asociada retorne verdadero, es decir, mientras se encuentran ocurrencias de los patrones del lado izquierdo en los modelos correspondientes. Si el nodo de regla es compuesto, por cada vez que la evaluación de su regla asociada retorna verdadero, se evalúan sus subnodos. La forma en que serán evaluados los subnodos dependerá del subtipo de nodo compuesto.

Cuando un nodo regla tiene establecido el atributo *forEach* en verdadero, la regla asociada al mismo siempre debe tener al menos un patrón del lado izquierdo. Esto se debe a que los patrones del lado izquierdo expresan las condiciones que deben satisfacerse para generar el patrón del lado derecho. Si esta restricción no existiera, un nodo podría entrar en un bucle infinito, debido a que la regla siempre ejecutaría su

patrón del lado derecho.

Un *Nodo Simple* representa un nodo hoja, el cual no puede contener subnodos. El resultado de este nodo va a depender únicamente del resultado de la evaluación de su regla asociada.

Un *Nodo Compuesto* representa un nodo que puede contener subnodos. A través de los nodos compuestos es posible especificar la composición de las reglas simples y formar el árbol de reglas de transformación. Además, los nodos compuestos determinan el orden en que serán evaluados sus subnodos, es decir, los caminos posibles de ejecución dentro de un árbol. Los subnodos de un nodo compuesto siempre serán evaluados cuando la evaluación de la regla asociada al nodo compuesto retorne verdadero. En caso contrario los subnodos no serán evaluados. No obstante, el resultado final de un nodo compuesto dependerá del resultado de sus subnodos según el tipo de nodo compuesto.

Cuando se ejecuta una transformación, el recorrido del árbol es realizado siguiendo el algoritmo *depth-first (Primero en Profundidad)*. Este algoritmo siempre expande uno de los nodos a su nivel más profundo, y sólo cuando llega a un nodo hoja, regresa a los niveles menos profundos. Cada rama de un nodo compuesto es evaluada en su totalidad hasta llegar a los nodos hojas. Luego de ejecutar un nodo hoja, el algoritmo determina si existen hermanos y si no es así, regresa al nodo más cercano con subnodos no explorados (*"backtracking"*).

El valor que retorna la ejecución de un subnodo es utilizado por el nodo compuesto para determinar el camino a seguir, de acuerdo a la semántica del tipo de nodo compuesto. No obstante, es necesario determinar el valor que va a tomar un nodo compuesto cuando tiene establecido el atributo *forEach* en verdadero, es decir, cuando se ejecute varias veces. Esto se debe a que luego de ejecutarse varias veces su regla en forma exitosa, finalmente ésta retornará falso cuando no se encuentren más ocurrencias de sus patrones del lado izquierdo. Por lo tanto, el valor que va a retornar un nodo compuesto con el atributo *forEach* en verdadero será el valor que retorne en la primera evaluación de su regla asociada y de sus subnodos.

Las variables definidas en un nodo compuesto son heredadas en todos sus niveles de subnodos. De esta manera, estas variables pueden ser utilizadas como argumentos en

las llamadas a las reglas de los subnodos. Esto es útil para pasar información entre las reglas.

Los tipos de nodos compuestos son: *Nodo Secuencial*, *Nodo AND* y *Nodo OR*. Un *Nodo Secuencial* indica que la evaluación de sus subnodos debe ser realizada en forma secuencial, independientemente del resultado de sus subnodos. El orden de la secuencia se corresponde con el orden en que los subnodos fueron definidos. El resultado de un nodo secuencial dependerá solo del resultado de la evaluación de su regla asociada. La Figura 5-8 muestra la semántica de evaluación de este caso.

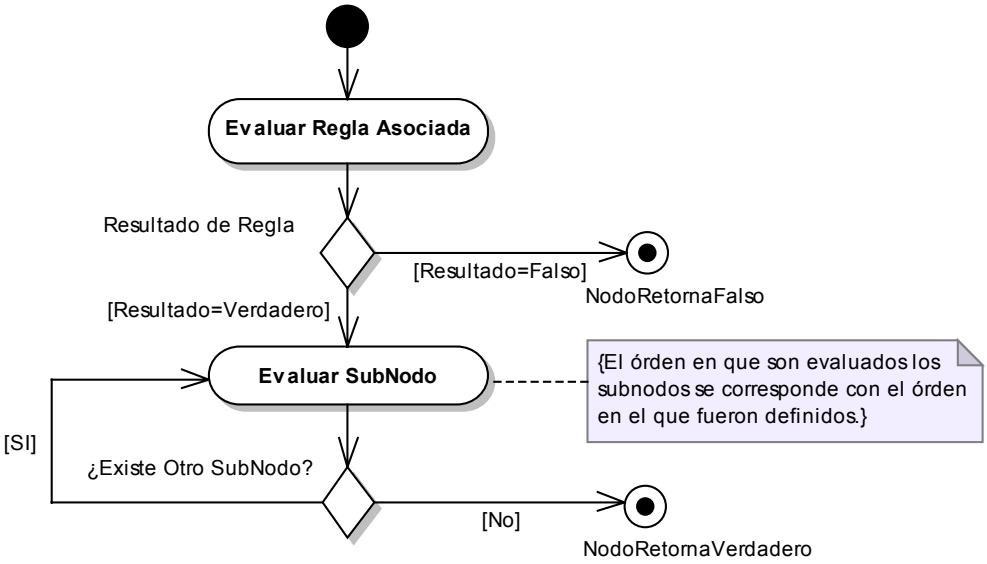


Figura 5-8. Semántica de evaluación de un nodo secuencial

Un *Nodo OR* representa una disyunción, en donde los subnodos serán evaluados hasta que uno de ellos retorne verdadero. Cuando esto ocurre, los restantes subnodos no serán evaluados. Si todos los subnodos retornan falso, entonces el nodo retornará falso. La Figura 5-9 muestra la semántica de evaluación de un *Nodo OR*.

Un *Nodo AND* especifica que todos sus subnodos serán evaluados mientras los mismos retornen verdadero. Si al menos uno de sus subnodos retorna falso, entonces retornará falso, y no se continuará con la ejecución de los restantes nodos. La Figura 5-10 muestra la semántica de evaluación de un *Nodo AND*.

El orden en que los subnodos de un *Nodo OR* y un *Nodo AND* son evaluados también se corresponde con el orden en el que los mismos han sido definidos.

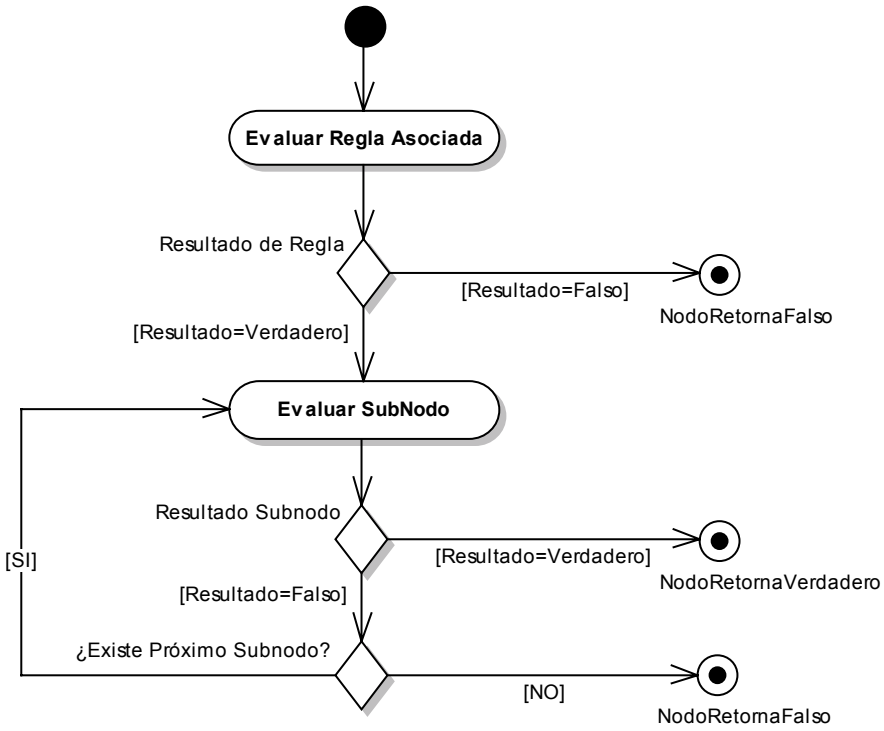


Figura 5-9. Semántica de evaluación de un *Nodo OR*

De esta manera, los diferentes tipos de nodos compuestos pueden ser utilizados para determinar cómo y cuándo sus subnodos deberán ser evaluados. Los resultados de los nodos compuestos siempre serán utilizados por sus nodos padres, para determinar los caminos a seguir en dicho nivel. Esto permite reducir los caminos a seguir en el árbol de reglas, ayudando a reducir el espacio de búsqueda de los patrones de las reglas y haciendo más eficiente una transformación de modelos.

Un *Nodo Invocación a Módulo* representa una llamada a otro módulo de regla, diferente al módulo donde el nodo está definido. Esto es utilizado para incorporar en un módulo de reglas el comportamiento definido en otro módulo. De esta manera es posible reutilizar también módulos de reglas. Los argumentos a utilizar en la llamada al módulo de reglas serán las variables definidas en los nodos ancestros del nodo de invocación a módulo, o bien los parámetros del módulo.

Un *Nodo de Invocación Recursiva* representa una llamada recursiva a un nodo definido dentro del mismo módulo de reglas. El nodo al que hace referencia el nodo de invocación recursiva debe ser un nodo ancestro de este último. De esta manera, es posible definir estructuras recursivas de nodos. No obstante, la terminación de la recursividad debería ser tenida en cuenta por el diseñador, para alcanzar un estado final

en la ejecución de la transformación. La terminación de la recursividad dependerá de la definición de los patrones del lado izquierdo de las reglas invocadas, como así también de la estructura del árbol de reglas. Al igual que un nodo de invocación a módulo, un nodo recursivo debe definir los argumentos que se corresponden con los parámetros del nodo a invocar. Los argumentos serán las variables definidas en los nodos ancestros del nodo de invocación a módulo.

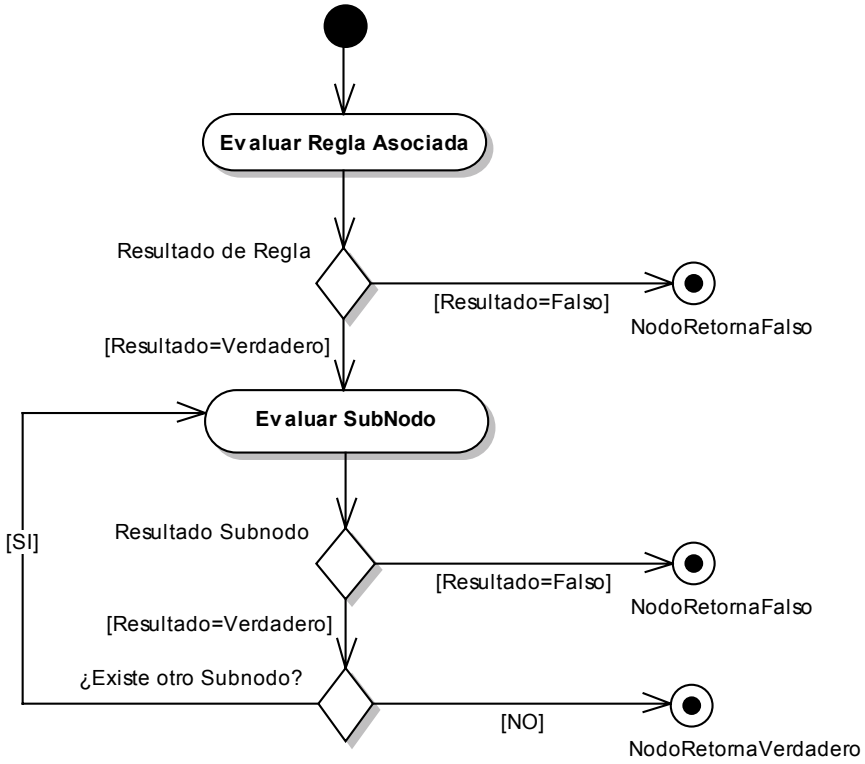


Figura 5-10. Semántica de evaluación de un *Nodo AND*

5.3.1.2. Soporte para la Implementación de los Patrones

El lenguaje de transformación de modelos soporta la declaración de los patrones de las reglas, pero no provee los constructores para especificar la semántica operacional de estos patrones. Por lo tanto, se deben proveer implementaciones de dichos patrones para que los mismos puedan ser integrados a las definiciones de las transformaciones realizadas con el lenguaje de transformación de modelos. Para soportar la implementación de los patrones de las reglas se provee una API Java, la cual es provista por la herramienta que soporta al lenguaje de transformación de modelos. Esta API consiste de un simple framework que define las clases que pueden ser utilizadas y extendidas para implementar los patrones. De esta manera, la lógica de los patrones debe ser definida usando el lenguaje de programación Java.

Un patrón puede ser implementado creando una clase Java que extienda la clase *PatronAbstracto*. Dicha clase provee los métodos para la definición y manipulación de los parámetros requeridos en los patrones. Para la manipulación de los objetos de los modelos dentro de los patrones se deberá utilizar otra API, específica para cada metamodelo en el que el patrón está basado, como se describe en la sección 5.4.

La lógica del patrón debe ser realizada por el desarrollador de la transformación, proveyendo una implementación del método *Ejecutar* de la clase *PatronAbstracto*. Dicha lógica podría ser implementada en forma ad-hoc o la misma podría seguir la semántica de algunos de los enfoques de transformación de modelos. En particular, en este método de transformación de modelos, se propone definir la lógica de los patrones siguiendo la semántica del lenguaje de modelado reglas, descrito a continuación.

5.3.2. El Lenguaje de Modelado de Reglas

Con la utilización del lenguaje anterior, es posible definir y ejecutar transformaciones de modelos. El lenguaje de transformación de modelos permite la definición de la composición y ordenamiento de las reglas, como así también la declaración de las reglas y sus patrones, en un alto nivel de abstracción. Esto posibilita comprender la semántica de ejecución de las reglas de transformación desde la perspectiva global del proceso de una transformación.

Por otra parte, también es importante entender el comportamiento de los patrones, para poder entender cada una de las reglas. No obstante, la definición e implementación de los patrones de las reglas utilizando un lenguaje de programación como Java provee un bajo nivel de abstracción. Con este lenguaje, el entendimiento de la semántica operacional de un patrón puede ser difícil. Por lo tanto, para definir y entender las reglas y sus patrones es necesario un lenguaje de más alto nivel. Para ello, el *Lenguaje de Modelado de Reglas* da soporte a la definición de los patrones de las reglas en una forma declarativa y visual. Dicho lenguaje no es ejecutable, es decir, actualmente no es interpretado por software. Este lenguaje tiene como propósito posibilitar la definición, análisis y entendimiento de los patrones de las reglas, sin entrar en los detalles de implementación de los mismos. El objetivo es utilizar las definiciones de los patrones de las reglas basadas en este lenguaje, como una guía para definir el algoritmo de los patrones cuando los mismos son implementados.

Debido a que en el lenguaje de transformación de modelos la declaración de los patrones está separada de la implementación, es posible utilizar diferentes enfoques de transformación de modelos para la definición de patrones. Por ejemplo, es posible utilizar un enfoque relacional, como el definido en QVTP (QVTP, 2003), el cual provee una notación gráfica para definir en forma declarativa las reglas. También, es posible utilizar un enfoque basado en gramáticas de grafos, como el propuesto en VIATRA (Varro y otros, 2002) y Gmorph (Sendall, 2003). Las gramáticas de grafos también proveen un enfoque declarativo para la especificación de los patrones de las reglas.

Aunque los enfoques relacionales y los enfoques basados en gramática de grafos son similares (Küster y otros, 2004), los últimos ofrecen una semántica operacional, la cual es más intuitiva para el entendimiento de los patrones de las reglas y por ende, para el entendimiento de las reglas. Las gramáticas de grafos han sido utilizadas en varios campos de aplicación y poseen una base formal sólida, como así también existen algoritmos que pueden ser utilizados para la aplicación de las mismas. A través del uso de herramientas que soportan las gramáticas de grafos, también es posible chequear ciertas propiedades de los patrones y las reglas de transformación.

Por otra parte, las gramáticas de grafos tienen otra ventaja particular para lenguajes de modelado gráficos como UML, debido a que, desde un punto de vista matemático, los modelos gráficos pueden ser considerados como grafos (Sendall, 2003). En el caso de UML, es posible definir el grafo que representa al patrón de una regla de transformación directamente como un fragmento de un diagrama de UML. No obstante, como se señala en varios trabajos (Varro y otros, 2002) (Sendall, 2003) es necesario extender los diagramas de UML para representar ciertas características que deberían ser tenidas en cuenta en los patrones de las reglas de transformación, cuando los mismos están basados en las gramáticas de grafos.

Por lo tanto, el lenguaje de modelado de reglas es provisto como un perfil UML y el mismo está basado en la teoría formal de gramáticas de grafos. Una gramática de grafos consiste de un conjunto de reglas, en donde cada una consiste de un grafo del lado izquierdo (LHS) y un grafo del lado derecho (RHS). La regla indica que si el grafo LHS es encontrado dentro del grafo de entrada (modelo de entrada), éste es reemplazado por el grafo RHS dentro del grafo de salida (modelo de salida). Para ser compatible con el lenguaje de transformación de modelos propuesto, una gramática

puede contener reglas que poseen uno o más grafos del lado izquierdo como así también uno o más grafos del lado derecho.

Para representar las gramáticas de grafos, el lenguaje de modelado de reglas propuesto está basado en el concepto de colaboraciones de objetos de UML2. Por lo tanto, los patrones de transformación de las reglas son visualizados en diagramas de estructura compuesta de UML2. Cada patrón es definido como una colaboración de objetos estereotipada <<Pattern>>. La regla es definida como una clase estereotipada <<TransformationRule>>, en donde los patrones asociados a las reglas son indicados con relaciones de dependencia estereotipadas <<LHS>> y <<RHS>>. Dichas relaciones indican para una regla cuáles son los patrones del lado izquierdo y los patrones del lado derecho, respectivamente. Como ejemplo, la Figura 5-11 muestra una regla Z, la cual posee como patrón LHS el patrón X y como patrón RHS el patrón Y. En forma general, la regla indica que el grafo representado por el patrón X es sustituido por el grafo del patrón Y.

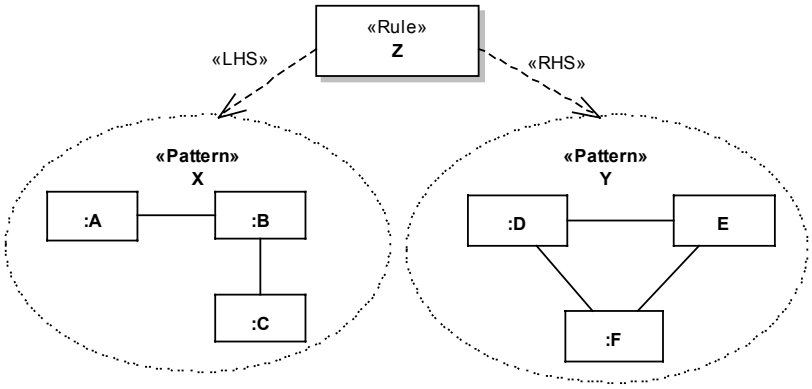


Figura 5-11. Ejemplo de definición de los patrones de una regla.

En el caso de patrones LHS, la colaboración de objetos representa el fragmento a encontrar en el modelo de entrada. En el caso de patrones RHS, la colaboración de objetos representa el fragmento que será generado en el modelo de salida.

Con el objetivo de definir patrones utilizando la sintaxis abstracta de los lenguajes de modelado, dentro de un patrón, los tipos de los objetos se corresponden con las metaclases del metamodelo correspondiente. Las asociaciones entre los objetos también se corresponden con las asociaciones establecidas en el metamodelo entre las metaclases de dichos objetos. En cambio, las multiplicidades de las asociaciones en un patrón no se corresponden con las multiplicidades formales definidas en el metamodelo.

Aquí sólo es posible utilizar multiplicidades fijas. De esta manera, una multiplicidad en un patrón de colaboración de objetos representa sólo el número de objetos a encontrar en la asociación. El valor por defecto de las multiplicidades es 1.

Como ejemplo de introducción de un simple patrón LHS, se describe el significado del patrón *X* definido en la regla de la Figura 5-11. Este patrón indica que en el modelo de entrada deben encontrarse tres objetos de tipo *A*, *B* y *C* respectivamente, en donde el objeto de tipo *A* debe tener una asociación con el objeto de tipo *B*, y éste último debe tener una asociación con el objeto de tipo *C*.

No obstante, tal como está definido el patrón, existen ciertas ambigüedades en la interpretación del mismo. Por ejemplo, si se considera un modelo de entrada que contiene los objetos *A1*, *B1* y *C1* cuyos tipos son *A*, *B* y *C* respectivamente, es posible que con este patrón se encuentren dos veces estos mismos objetos, cuando en realidad se desea encontrarlos una sola vez. Esto ocurre debido a que en una ejecución de la regla, el patrón puede comenzar encontrando un *A* y luego encuentra un *B* y un *C*, y en otra ejecución comienza encontrando un *C* y luego un *B* y un *A*.

Para eliminar esta ambigüedad dos características deben tenerse en cuenta. En primer lugar, es necesario indicar el primer objeto que debe ser encontrado y a partir del cual se realizará el recorrido del patrón LHS. Esto puede ser indicado adicionando la restricción *start* asociada a dicho objeto. Además, es necesario indicar que el algoritmo de un patrón LHS debe establecer que una vez que se encontró una ocurrencia del patrón LHS, los objetos encontrados deben ser marcados o almacenados como ya leídos. De esta manera, en la próxima ejecución de la regla, cuando se encuentren estos objetos ya leídos, el patrón los debe descartar y continuar con la búsqueda de otros objetos no leídos.

Como se especificó en el lenguaje de transformación de modelos, una regla posee parámetros para pasar y obtener información desde los patrones, como así también posee variables privadas para pasar información desde los patrones LHS a los patrones RHS. Además, tanto los parámetros como las variables pueden representar alguno de los tipos de datos básicos o bien alguna metaclase del metamodelo al que el patrón es conforme. Por lo tanto, un parámetro de entrada que representa un objeto a ser utilizado por el patrón es indicado con la restricción *In* asociada a dicho objeto. Este objeto debe tener definido un nombre, el cual representa el nombre dado al parámetro.

De manera similar, un parámetro de salida de un patrón es indicado con la restricción *Out* asociada al objeto que representa el parámetro. El nombre del objeto es el nombre del parámetro del patrón.

Las variables privadas son indicadas entre los signos "<" y ">", por ejemplo *<NombreProtocolo>*. Las variables privadas son utilizadas para pasar valores de los atributos de objetos entre los patrones LHS y RHS, como así también objetos. Existen diferentes operadores a utilizar para dar valor a las variables privadas a partir del valor de un atributo, asignar valores a los atributos a partir del valor de una variable, o determinar si el valor de un atributo es igual al de la variable. El operador "==>" es utilizado para guardar el valor de un atributo en la variable. El operador "<==" es utilizado para asignar al atributo, el valor contenido en la variable. El operador "==" es utilizado para determinar si el valor de un atributo es igual al valor mantenido en la variable. Además, los operadores "<==" y "=="pueden ser utilizados con constantes, en lugar de variables. Aquellos atributos de los objetos, los cuales no son utilizados dentro de los patrones son ignorados y no son visualizados.

La Figura 5-12 muestra un ejemplo de una regla, en donde los patrones se corresponden a diferentes metamodelos. La regla representa la transformación de una clase definida en UML a una clase en Java. En el patrón LHS, el objeto *P* de tipo *Package* es definido como el parámetro de entrada del patrón. Además, aunque en este caso es trivial, también es definido el objeto de comienzo por el cual se debe comenzar a buscar en el modelo de origen. Este patrón indica que se debe encontrar un objeto de tipo *class* contenido en el paquete pasado como parámetro. Cuando una ocurrencia del mismo se encuentra, se ejecuta el patrón RHS. Este patrón indica que se generará una clase Java correspondiente a la clase UML encontrada, la cual debe ser incluida dentro del paquete Java que es pasado como parámetro de entrada. La variable *NombreClase* es utilizada para asignar al nombre de la clase Java el nombre de la clase UML encontrada. Tanto la clase UML encontrada como la clase Java generada son parámetros de salida de sus respectivos patrones. Cada uno de los parámetros de los patrones también es definido como parámetro de la regla.

Para traducir un conjunto de clases UML, en un conjunto de clases Java, será necesario ejecutar esta regla varias veces, mientras se encuentre una ocurrencia del

patrón LHS. Cada vez que una clase UML es encontrada, será descartada en la próxima búsqueda.

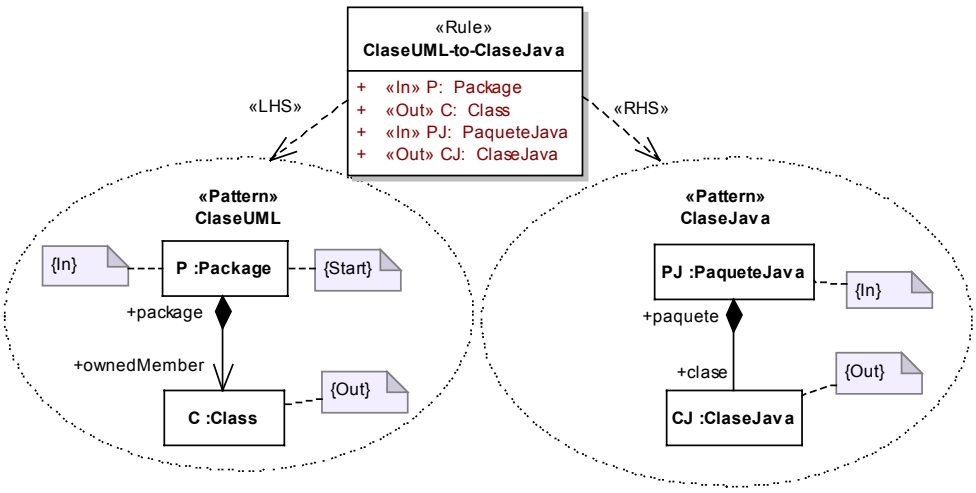


Figura 5-12. Ejemplo de patrones con diferentes metamodelos.

Para realizar transformaciones sobre un mismo modelo, es necesario especificar otros aspectos adicionales para definir los patrones:

- Los objetos del patrón LHS, que son mantenidos o preservados sin cambios en el patrón RHS generado, se indican simplemente definiendo en el patrón RHS aquellos objetos definidos en el patrón LHS.
- Los objetos del patrón LHS, que serán removidos y no existirán en el patrón RHS, se indican con la restricción remove asociada al objeto del patrón LHS que no será mantenido en el patrón RHS.
- Los nuevos objetos en el patrón RHS, que son creados y que no existían en el patrón LHS, se indican con la restricción *new* asociada al nuevo tipo de objeto creado en el patrón RHS.

Los anteriores puntos se corresponden con la semántica de los operadores *Bind*, *Delete* y *New* de la teoría de grafos (Agrawal y otros, 2003) (Varro y otros, 2002). Estos operadores son utilizados sólo para los objetos, pero no para las asociaciones entre objetos. Una asociación es mantenida si los pares de objetos en los extremos de la misma son mantenidos en el RHS. Una asociación es creada si esta no estaba en el patrón LHS. Una asociación es removida si uno de los nodos de sus extremos es eliminado.

Es posible indicar en un patrón LHS, las condiciones de aplicación negativas, tanto sobre los objetos como sobre las asociaciones, para indicar que un objeto o asociación no debe existir en el patrón a encontrar. Esto es indicado con una restricción *Neg* asociada al objeto o asociación que está restringiendo.

También es posible adicionar restricciones OCL (Object Constraint Language) (OMG, 2003e) sobre los patrones. Esto es necesario debido a que en algunos casos, no es posible o es muy engorroso, especificar restricciones con grafos. Para estos casos, un lenguaje lógico de predicado de primer orden, el cual permite definir expresiones lógicas sobre un modelo UML, es más adecuado debido a que tiene un mayor poder expresivo que una notación gráfica (Sendall, 2003). Por ejemplo, las restricciones negativas sobre un patrón LHS también podrían ser definidas directamente con OCL, en lugar de utilizar la restricción *Neg* sobre los objetos del patrón.

Un ejemplo de este tipo de regla es mostrado en la Figura 5-13. El patrón LHS posee como parámetro de entrada el objeto *A1* de tipo *A*. Este es también el objeto por el cual se debe comenzar la búsqueda en el patrón. Cuando un objeto de tipo A es encontrado, éste debe tener una asociación con un objeto *B1* de tipo *B*, quien a su vez debe tener un objeto de tipo *C*. La condición negativa sobre el objeto de tipo *D* indica que no debe existir un objeto de tipo *D* asociado al objeto de tipo *C*. Además, la restricción OCL asociada al patrón LHS indica que si el atributo *isActive* tiene un valor verdadero, el atributo *isNotRequired* en el objeto B1 debe también tener un valor verdadero. Finalmente, una vez encontrados estos objetos y chequeados las restricciones sobre los mismos, el objeto de tipo *B* debe ser removido.

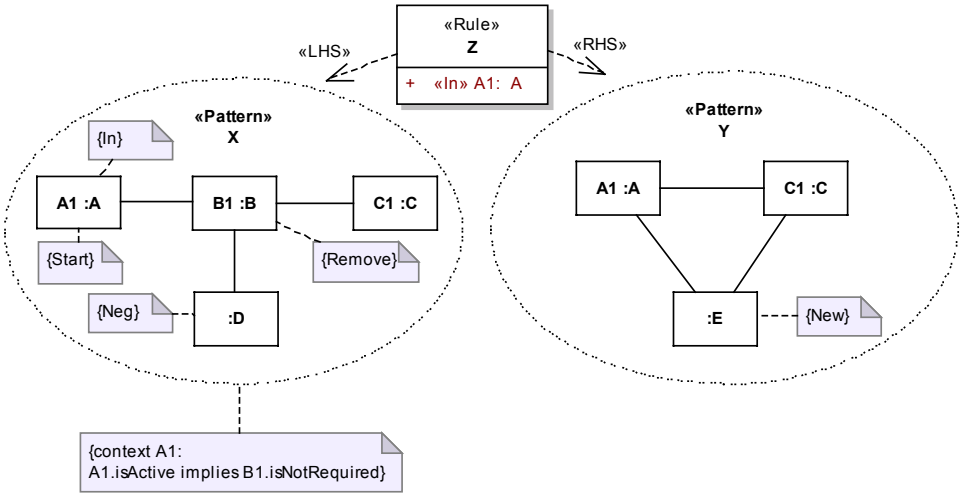


Figura 5-13. Ejemplo de patrones con el mismo metamodelo.

El patrón RHS indica que el resultado final de la regla debe ser el siguiente: Se debe crear una asociación entre los objetos de tipo *A* y de tipo *C* encontrados. Estos son los objetos que se deben mantener en el resultado final de la regla. Además, un nuevo objeto de tipo *E* debe ser creado, como así también deben crearse las asociaciones de este objeto con el objeto de tipo *A* y el objeto de tipo *C*.

5.3.2.1. Aspectos Formales del Lenguaje de Modelado de Reglas

Como se mencionó anteriormente, los fundamentos teóricos del lenguaje de modelado de reglas pueden ser expresados en términos de gramáticas de grafos. Una formalización detallada de gramáticas de grafos para transformación de modelos puede ser encontrada en (Varro y otros, 2002).

Tanto los metamodelos que expresan la sintaxis abstracta de los lenguajes de modelado (vistos en la forma de diagramas de clases), como los modelos (vistos en la forma de diagramas de objeto), pueden ser representados como grafos dirigidos, con tipos, etiquetados y con atributos. De esta manera, al nivel de metamodelo, cada clase es traducida en un nodo del grafo y todas las asociaciones son proyectadas como aristas del grafo, formando un *Grafo de Tipos GT*. El grafo de tipo representa la gramática válida para definir fragmentos de patrones, ya que los mismos pueden ser analizados como sentencias de dicha gramática. Al nivel de modelo, los objetos y asociaciones entre ellos son traducidos en nodos y aristas respectivamente, formando un *Grafo de Modelo M*, el cual está basado en la gramática definida por un grafo de tipo *GT*. En *M* los nodos y aristas son instancias de los nodos y aristas del *GT* correspondiente.

Una regla de transformación de grafos *R = (Lhs, Rhs)* consiste de dos grafos, el grafo *Lhs* que representa el lado izquierdo de la regla, y el grafo *Rhs* que representa el lado derecho de la regla. En nuestro caso es posible tener reglas con más de un *Lhs* y *Rhs*. Un grafo *Lhs* es definido de acuerdo a la gramática expresada por un grafo de tipo GT_E, el cual representa el metamodelo de entrada. Debido a que también se aceptan patrones LHS definidos en base al metamodelo de salida, un grafo *Lhs* también puede ser definido de acuerdo a un grafo de tipo GT_S, el cual representa el metamodelo de salida. Además, GT_E y GT_S pueden representar el mismo grafo de tipo cuando los modelos de entrada y salida corresponden al mismo metamodelo. Un grafo *Rhs* es siempre definido de acuerdo a la gramática expresada por un Grafo de Tipo GT_S que representa el metamodelo de salida.

Siguiendo la teoría de transformación de grafos, para la aplicación de una regla de transformación de grafos a un grafo M_E (el modelo de entrada) un algoritmo simplificado puede ser utilizado, el cual contiene los siguientes pasos:

1. Seleccionar los grafos *Lhs* definidos de acuerdo a un GT_E. Para cada uno de ellos realizar lo siguiente:
    a. Encontrar una ocurrencia del grafo *Lhs* en el grafo M_E.
        i. Iniciar la búsqueda por el nodo indicado como nodo de comienzo de la búsqueda.
        ii. Encontrar cada uno de los nodos indicados en el grafo, recorriendo las aristas de cada nodo.
        iii. Encontrar las aristas indicadas en el grafo.
    b. Chequear que se satisfagan las condiciones de aplicación negativas, definidas en el grafo *Lhs*, en el subgrafo M_E encontrado en el paso 1.a.
    c. Chequear que se satisfagan las restricciones OCL.
    d. Remover los nodos marcados como *remove* en el grafo *Lhs*, como así también aquellas aristas que tienen en uno de sus extremos los nodos removidos.
    e. Habiéndose realizado los anteriores pasos, las variables utilizadas en los grafos *Lhs* son instanciadas con los valores concretos de los nodos encontrados en el grafo M_E.
2. Seleccionar los grafos *Lhs* definidos de acuerdo a un GT_S. Para cada uno de ellos realizar los mismos pasos de la etapa *1*, pero operando sobre el grafo M_S que está siendo generado. Esto es útil para expresar precondiciones sobre el grafo M_S, antes de la generación definitiva del patrón *Rhs*. Este enfoque es similar al utilizado en (Sendall, 2003).
3. Crear el grafo *Rhs* definido de acuerdo a un GT_S sobre el grafo M_S. Crear un nodo en M_S para cada nodo que es parte del *Rhs* pero que no es parte de los grafos *Lhs*. Adicionar las aristas correspondientes a los nuevos nodos.

Este algoritmo puede ser utilizado como base para definir la semántica operacional de los patrones, cuando se provee una implementación de los mismos en el lenguaje de

programación Java. Para traducir este algoritmo a la implementación de los patrones es necesario tener en cuenta que la ejecución de los patrones LHS y RHS es realizada en forma separada, como se ha definido en el lenguaje de transformación de modelos. Esto es diferente a algunos enfoques de transformación de grafos, en donde el algoritmo de ejecución de la regla no está separado de la ejecución de los patrones. Por lo tanto, para implementar los patrones en base al algoritmo de aplicación de reglas de transformación de grafos, el desarrollador debe implementar los pasos *1* y *2* por dentro de la lógica de los patrones LHS, y el paso *3* en la lógica de los patrones RHS.

5.3.3. Reglas de Producción de Código XML

Los anteriores lenguajes tratan con las transformaciones de modelos a modelos. No obstante, como se resaltó en los requerimientos (sección 5.2), es necesario proveer mecanismos para transformar modelos a código XML. No obstante, las reglas de producción de código XML dependen de cómo los modelos de salida de la transformación son definidos. En el dominio de transformaciones de modelos de procesos colaborativos, los modelos de salida están basados en los lenguajes provistos por los estándares B2B. Dos soluciones son posibles para construir los modelos de salida: la construcción de un perfil UML o la generación de un metamodelo específico basado en MOF.

Los lenguajes provistos por los estándares B2B, tanto para especificar procesos de negocio como para especificar las interfaces de los socios, están basados en XML. Para estos lenguajes, el esquema XML que define a los mismos es lo que un metamodelo es para un lenguaje de modelado. Por lo tanto, un metamodelo de un lenguaje basado en XML se corresponde con un esquema XML para dicho lenguaje. Una instancia del metamodelo, es decir, un modelo, se corresponde con un documento XML, el cual es una instancia del esquema XML.

De esta manera, a partir de un esquema XML es posible derivar el metamodelo correspondiente. Luego, a partir de este metamodelo es posible construir modelos. Finalmente, a partir de estos modelos es posible generar el documento XML correspondiente (Figura 5-14).

A diferencia de la utilización de un perfil UML, el uso de metamodelos específicos tiene la ventaja de que la transformación de modelos a código XML es casi directa.

Además, las reglas de transformación para generar el código XML pueden ser reutilizadas en la generación de especificaciones para cada uno de los lenguajes de los estándares B2B.

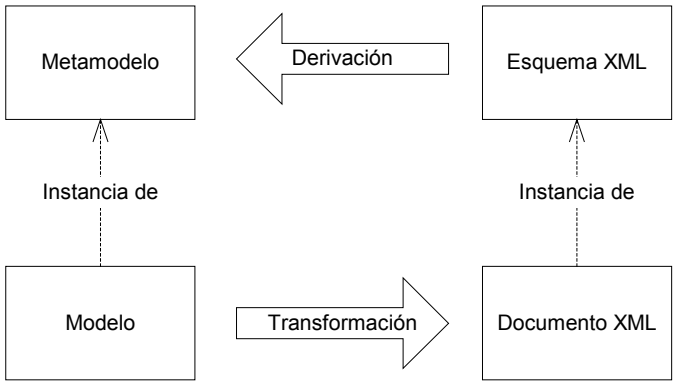


Figura 5-14. Derivación de metamodelos y transformación a XML

Teniendo en cuenta estos principios de correspondencia, las reglas de producción de modelos a código XML son las siguientes:

1. Los objetos de un modelo son traducidos en elementos XML. Es decir, por cada objeto del modelo, se genera un elemento XML que corresponde al tipo (metaclase) del objeto, en donde dicho elemento debe poseer un atributo *name* con el nombre del objeto. Por ejemplo, un objeto *ReleaseAgainstBlanketPurchaseOrder* de tipo *BinaryCollaboration* es traducido en el siguiente código XML:

```
<BinaryCollaboration name="ReleaseAgainstBlanketPurchaseOrder"
<BinaryCollaboration/>
```

2. Los atributos de los objetos son traducidos como atributos XML del correspondiente elemento XML, con los respectivos valores. Por ejemplo, si el objeto *ReleaseAgainstBlanketPurchaseOrder* posee un atributo *timeToPerform* con un valor "PT7D", entonces el código XML será el siguiente:

```
<BinaryCollaboration name="ReleaseAgainstBlanketPurchaseOrder"
                     timeToPerform="PT7D"
<BinaryCollaboration/>
```

3. Una asociación de agregación o composición entre dos objetos es expresada definiendo el objeto que es parte del objeto agregado, como un elemento hijo del elemento que representa el objeto agregado. El elemento hijo se corresponde con el tipo del objeto parte cuyo atributo *name* tiene el nombre del objeto parte. Por

ejemplo, si el objeto *ReleaseAgainstBlanketPurchaseOrder* es un objeto compuesto que tiene como parte al objeto *BlanketPurchaseOrder* de tipo *CollaborationActivity*, entonces el código XML generado será el siguiente:

```
<BinaryCollaboration name="ReleaseAgainstBlanketPurchaseOrder"
                          timeToPerform="PT7D"
   <CollaborationActivity name="BlanketPurchaseOrder"/>
<BinaryCollaboration/>
```

4. Una asociación unidireccional entre objetos es expresada definiendo un atributo en el elemento XML que corresponde al objeto origen de la asociación. El nombre del atributo debe ser el mismo que el nombre del tipo del objeto referenciado. El valor del atributo debe ser el nombre del objeto. El elemento XML que representa al objeto referenciado será luego generado con la primer regla y éste tendrá el atributo *name*, indicando el nombre del objeto referenciado. Por ejemplo, si el objeto *BlanketPurchaseOrder* de tipo *CollaborationActivity* tiene un atributo de asociación que referencia al objeto *BlanketPurchaseOrder* de tipo *BinaryCollaboration*, entonces el código XML generado será el siguiente:

```
<BinaryCollaboration name="ReleaseAgainstBlanketPurchaseOrder"
                          timeToPerform = "PT7D"
   <CollaborationActivity name="BlanketPurchaseOrder"
                     binaryCollaboration="BlanketPurchaseOrder"/>
<BinaryCollaboration/>
<BinaryCollaboration name = "BlanketPurchaseOrder"/>
```

La última sentencia del código anterior será generada por la aplicación de la regla 1, cuando se encuentre el objeto *BlanketPurchaseOrder* de tipo *BinaryCollaboration*.

5.4. Herramienta de Transformación de Modelos

Con el objetivo de dar soporte a la definición y ejecución de transformaciones de modelos en el dominio de procesos colaborativos a través del método definido anteriormente, se ha desarrollado un prototipo de la herramienta de transformación de modelos que soporta dicho método. La herramienta implementa el lenguaje de transformación de modelos, provee las APIs para la implementación de los patrones y también implementa las reglas de producción para la generación de código XML. La herramienta fue implementada en Java y la arquitectura de la misma es mostrada en la Figura 5-15. Dicha arquitectura está basada en tres capas.

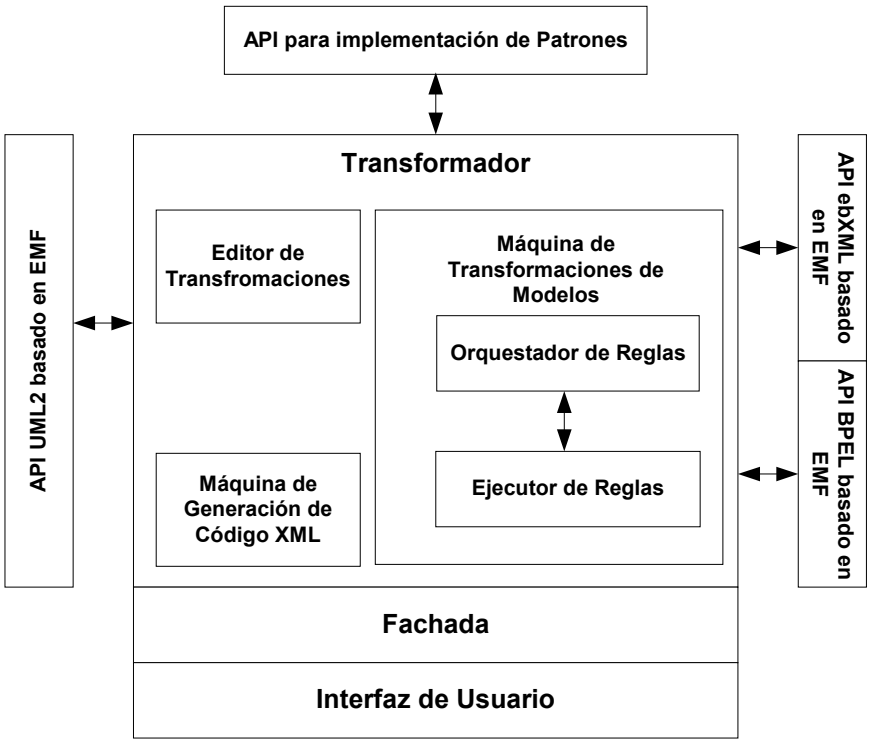


Figura 5-15. Arquitectura de la herramienta de transformación de modelos

El *Transformador* es la capa principal de la herramienta, la cual contiene los componentes requeridos para la definición, almacenamiento y ejecución de las transformaciones. Dentro de esta capa, el *Editor de Transformaciones* es el que soporta la definición de una transformación de modelos, de acuerdo al lenguaje de transformación de modelos definido anteriormente. Toda transformación de modelos es definida como un proyecto de transformación, el cual es almacenado en un archivo XML.

La *Máquina de Transformación de Modelos* interpreta las definiciones de las transformaciones de modelos y lleva a cabo la ejecución de las mismas. La máquina está compuesta de dos componentes: el orquestador de reglas y el ejecutor de reglas.

El *Orquestador de Reglas* interpreta la definición de los módulos de reglas para realizar la secuencia de ejecución de las reglas. Este componente es el encargado de la lógica de la transformación, siguiendo la semántica de ejecución de los nodos definidos en el árbol de reglas de cada módulo de reglas. Además, es el encargado de realizar el pasaje de información de los parámetros entre los módulos, como así también el manejo de las variables de los nodos de reglas, a ser utilizadas como argumentos en las llamadas

a las reglas asociadas a cada nodo. Cuando un nodo de regla es ejecutado, el orquestador de reglas invoca al ejecutor de reglas.

El *Ejecutor de Reglas* interpreta la definición de una regla simple y lleva a cabo la ejecución de sus patrones LHS y RHS. Éste es el encargado de establecer la correspondencia entre los argumentos definidos en la llamada a la regla en un nodo y los parámetros de las reglas. Además, realiza el pasaje de información entre una regla y sus patrones a través de los parámetros, como así también el pasaje de valor entre los patrones del lado izquierdo y los del lado derecho a través de las variables privadas.

A través del *Editor de Transformaciones* y la *Máquina de Transformación de Modelos* se especifican y ejecutan las transformaciones modelo-a-modelo. En la ejecución, la máquina de transformación de modelos toma un modelo de entrada y produce un modelo de salida. No obstante, es necesario generar a partir de un modelo de salida el código XML correspondiente, de acuerdo al estándar B2B que se está manejando. Esta tarea de la transformación es realizada por la *Máquina de Generación de Código XML*. Este componente tiene incorporado las reglas de producción definidas anteriormente para transformar un modelo de objetos a XML, con el objetivo de generar, a partir de estos modelos de salida almacenados en memoria, el documento XML correspondiente.

Para la generación de código XML, esta máquina recorre los objetos del modelo de salida utilizando el algoritmo *depht-first*. Cada vez que encuentra un objeto se ejecutan las reglas de producción en el orden en que fueron definidas. No obstante, para cada tipo de metamodelo, es necesario proveer una implementación específica para la generación del código XML. Aunque las reglas de producción definidas anteriormente son generales para cualquier tipo de modelo, se debe respetar el orden en el cual los elementos XML deben aparecer dentro del documento XML. Por lo tanto, es necesario proveer un algoritmo específico para recorrer los diferentes tipos de modelos de salida. Este algoritmo debe ser utilizado junto con las reglas de producción de código XML, para generar los documentos XML. Actualmente, la herramienta tiene implementado la generación de documentos en ebXML y en BPEL.

La *Fachada* es la capa que separa la capa principal de la capa de *interfaz de usuario*. En otras palabras, la *Fachada* permite la separación de la lógica del editor y las máquinas de transformación, del manejo de las interfaces de usuarios. De esta manera,

la interfaz del usuario puede ser provista como un cliente web basado en HTML o bien un cliente convencional. Actualmente, la herramienta provee este último tipo de interfaz de usuario.

Para la implementación de los patrones, la herramienta provee la API basada en Java, descrita anteriormente. Esta API permite integrar las definiciones de las transformaciones con la implementación de los patrones en el lenguaje de programación Java.

Por otra parte, como se describió en los requerimientos particulares, la herramienta de transformación de modelos debe tomar como modelos de entrada, modelos definidos con UP-ColBPIP y almacenados en un formato XMI. Actualmente, no existe una herramienta Case UML que permita la importación y exportación de archivos XMI que contengan modelos basados en UML2. Por un lado, algunas herramientas implementan sólo una parte del metamodelo de UML2. Por otro lado, otras herramientas implementan todo el metamodelo de UML2, pero generan los archivos XMI utilizando una versión anterior de UML. Este es el caso de la herramienta Case *Enterprise Architect* utilizada en el caso de estudio del capítulo anterior. Esto significa que el modelo generado en el archivo XMI no se corresponde con la sintaxis del metamodelo de UML2 y por lo tanto no es posible leer un modelo de UML2 sin tener información del formato específico utilizado por la herramienta para los archivos XMI.

Por lo tanto, para almacenar modelos UP-ColBPIP en un formato XMI para su posterior transformación, se ha utilizado Eclipse UML2 (Eclipse, 2004a), el cual es una implementación del metamodelo de UML2 basada en EMF (Eclipse Modeling Framework). EMF es un plug-in para el ambiente de desarrollo integrado Eclipse (Eclipse, 2004b) que soporta metamodelación y la generación de código para la construcción de herramientas de modelado. EMF provee un subconjunto de los conceptos básicos utilizados en MOF, y por lo tanto, EMF puede ser visto como una alternativa a MOF para la construcción de metamodelos.

El plug-in Eclipse UML2 contiene un editor a través del cual es posible definir modelos basados en UML2, y definir y utilizar perfiles de UML2. Soporta la definición de los modelos y los perfiles utilizando la sintaxis abstracta del metamodelo de UML2, pero no soporta la definición de modelos UML2 usando la sintaxis concreta de los mismos, es decir, usando la notación visual de UML2.

También, tanto para acceder y manipular los modelos de entrada, como para generar y manipular los modelos de salida, es necesario contar con las APIs específicas para estas tareas. Estas APIs son requeridas para la implementación de los patrones. Para el caso de los patrones definidos con respecto a los modelos de entrada, es posible acceder y manipular directamente el archivo XMI, a través de una API específica para el manejo de documentos XML, como DOM (Document Object Model). No obstante, también es posible crear la instancia del modelo de entrada en memoria a partir del archivo XMI, y manipular ésta instancia en los patrones. Esto puede ser realizado utilizando también Eclipse UML2, dado que el mismo provee una API para la manipulación de modelos de UML2 en forma programática. Además, esta API también puede ser utilizada para el caso de modelos de salida basados en UML2.

Para la manipulación, dentro de los patrones, de los modelos de salida que están basados en un metamodelo particular y no en UML2, una API específica también debería ser provista. Una opción es generar dicha API modelando el metamodelo del modelo de salida en un diagrama de clases utilizando una herramienta Case UML, que permita luego la generación de código Java a partir de dichas clases. Otra opción es generar dicha API en forma automática utilizando el plug-in EMF para Eclipse. A través del editor de EMF es posible construir metamodelos y luego generar la API correspondiente para la manipulación de modelos basados en dichos metamodelos. El código generado por EMF es código Java. En lugar de editar el metamodelo, la API en EMF también puede ser generada a partir de un diagrama de clases describiendo el metamodelo o bien a partir de un esquema XML. Esto último es de suma utilidad, ya que como se dijo anteriormente, los estándares B2B proveen los esquemas XML de sus respectivos lenguajes. Actualmente, se han generado las APIs basadas en EMF para la manipulación de modelos conformes a ebXML y a BPEL. Estas APIs son utilizadas en la herramienta, principalmente para la implementación de los patrones, en las transformaciones de modelos UP-ColBPIP a modelos basados en dichos estándares B2B.

5.4.1. Definiendo una Transformación con la Herramienta

A continuación se describen las interfaces del editor de transformaciones de la herramienta a través de un simple ejemplo de transformación de modelos. Dicho ejemplo consiste en una transformación de modelos utilizada para el refinamiento de

modelos UP-ColBPIP, particularmente, para generar la vista de las interfaces de negocio a partir de la vista de protocolos de interacción.

La herramienta consiste de una interfaz sencilla (Figura 5-16), en la cual la ventana principal contiene tres pestañas:

- La pestaña del Proyecto de Transformación, la cual contiene la información general del proyecto, como los metamodelos de entrada y salida permitidos.
- La pestaña de Reglas de Transformación, en donde se define y visualizan las reglas con sus patrones.
- La pestaña Módulos de Reglas, en la cual se definen y visualizan los módulos de reglas con sus respectivos árboles de reglas.

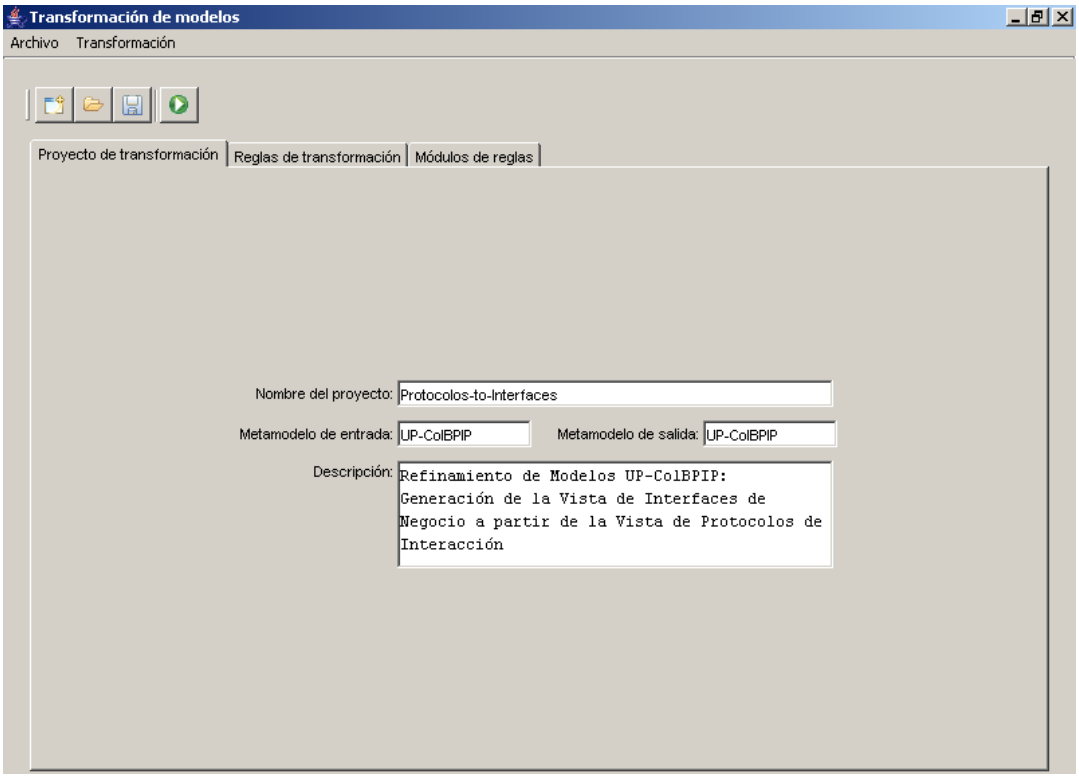


Figura 5-16. Ventana principal de la herramienta de transformación de modelos

La Figura 5-16 muestra la información principal del proyecto de transformación definido. Aquí se indica el nombre del proyecto, y los metamodelos de entradas y salidas, y una descripción del mismo. En este ejemplo, UP-ColBPIP es definido como el metamodelo de entrada y también como el metamodelo de salida.

Una vez definido el proyecto es posible comenzar a declarar las reglas con sus patrones. Actualmente, las clases que representan a los patrones deben estar previamente definidas de acuerdo a la API provista para su implementación, aunque en esta etapa no es necesario proveer la implementación de la lógica de los patrones.

La Figura 5-17 muestra la ventana que permite definir las reglas de transformación. En este caso, se define la regla *Roles-to-BusinessInterfaces*. La regla posee varios parámetros. Por ejemplo, el parámetro *InterfaceR1ToR2* es de tipo *Interface*, y el mismo es de salida y será utilizado como argumento en los patrones de la regla. Además, la regla posee varias variables con sus tipos correspondientes.

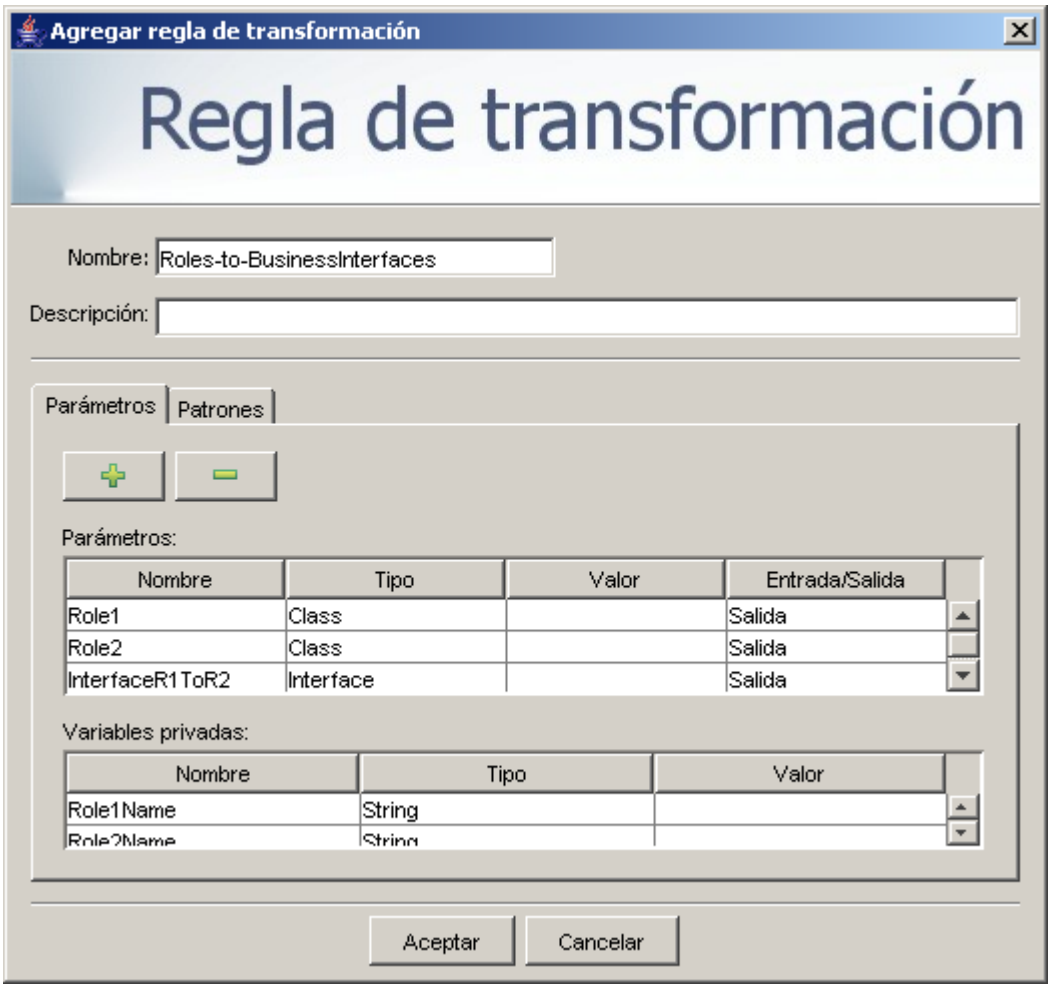


Figura 5-17. Creación de regla de transformación (parámetros y variables)

La incorporación de un patrón a la regla es mostrada en la Figura 5-18. Aquí es posible seleccionar el patrón desde la lista de patrones implementados y determinar a qué lado de la regla corresponde, es decir, si es un patrón del lado izquierdo o del lado derecho. Cuando se agrega un patrón a la regla deben especificarse los argumentos del

patrón, los cuales son los parámetros o las variables privadas de la regla. Cuando se define un parámetro requerido por el patrón, en el campo nombre es posible obtener la lista de parámetros y variables definidas en la regla que pueden ser utilizados como argumentos.

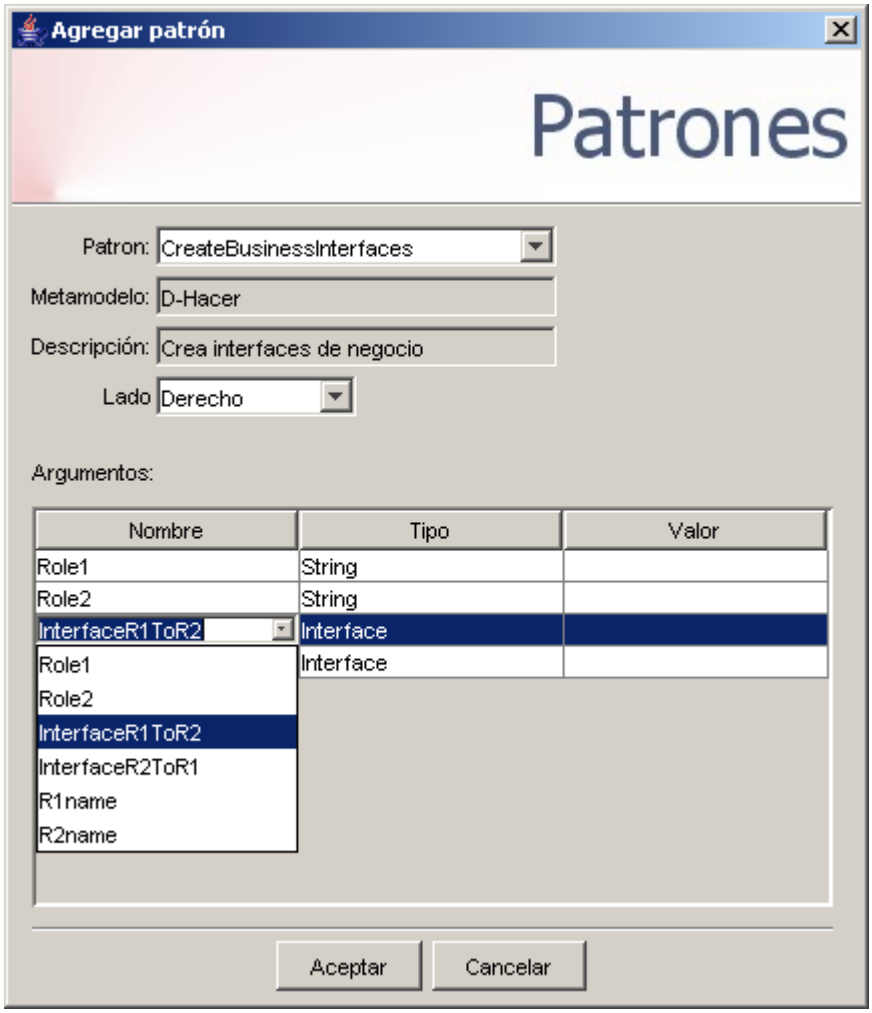


Figura 5-18. Incorporación de un patrón a una regla

La Figura 5-19 muestra la otra pestaña del cuadro de diálogo *creación de regla*, en donde se especifican los patrones de la regla.

La Figura 5-20 muestra la pestaña de la ventana principal, en la cual se visualizan las reglas que han sido definidas.

A medida que se definen las reglas, es posible definir cada módulo de reglas y los nodos de su árbol de reglas. La Figura 5-21 muestra la interfaz utilizada para visualizar y crear estos elementos. En la parte izquierda se provee la barra de herramientas que permite: agregar módulos, agregar nodos regla, agregar nodos de invocación recursiva,

y agregar nodos de invocación a módulo. Para agregar un nodo, debe seleccionarse previamente el nodo debajo del cual se desea agregar el nuevo nodo. Un módulo siempre se agrega debajo del nodo raíz que representa el modelo de transformación, cuyo nombre es el nombre del proyecto de transformación.

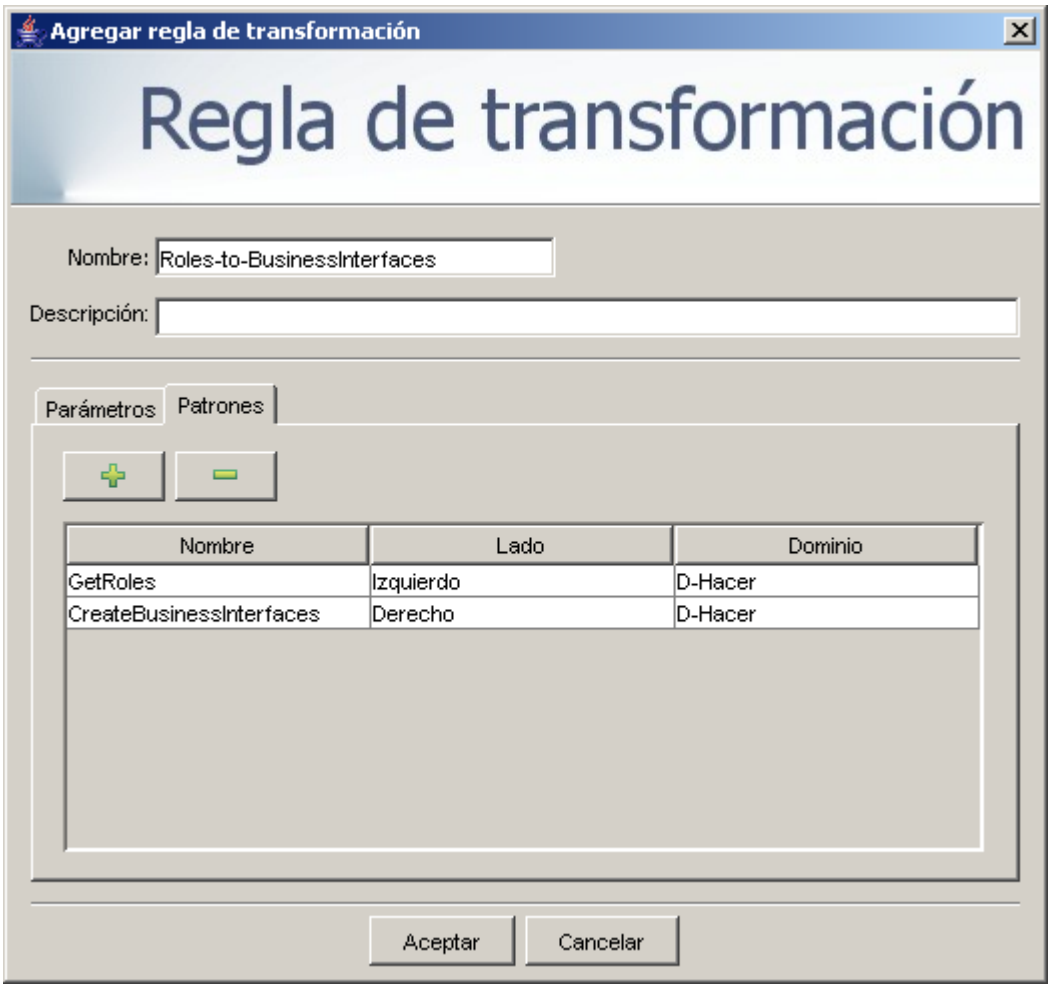


Figura 5-19. Creación de regla de transformación (patrones)

La Figura 5-22 muestra la definición del nodo *GetProtocol*. Cuando se está adicionando un nodo regla, ya sea un nodo compuesto o simple, se debe seleccionar entre las que están definidas, la regla a ser invocada. Las reglas disponibles son visualizadas en la caja de *Reglas de Transformación*. Además, es necesario indicar las variables que serán utilizadas como argumentos en la invocación a la regla asociada, de acuerdo a los parámetros requeridos en la regla. Esto se define en la caja de *Variables*.

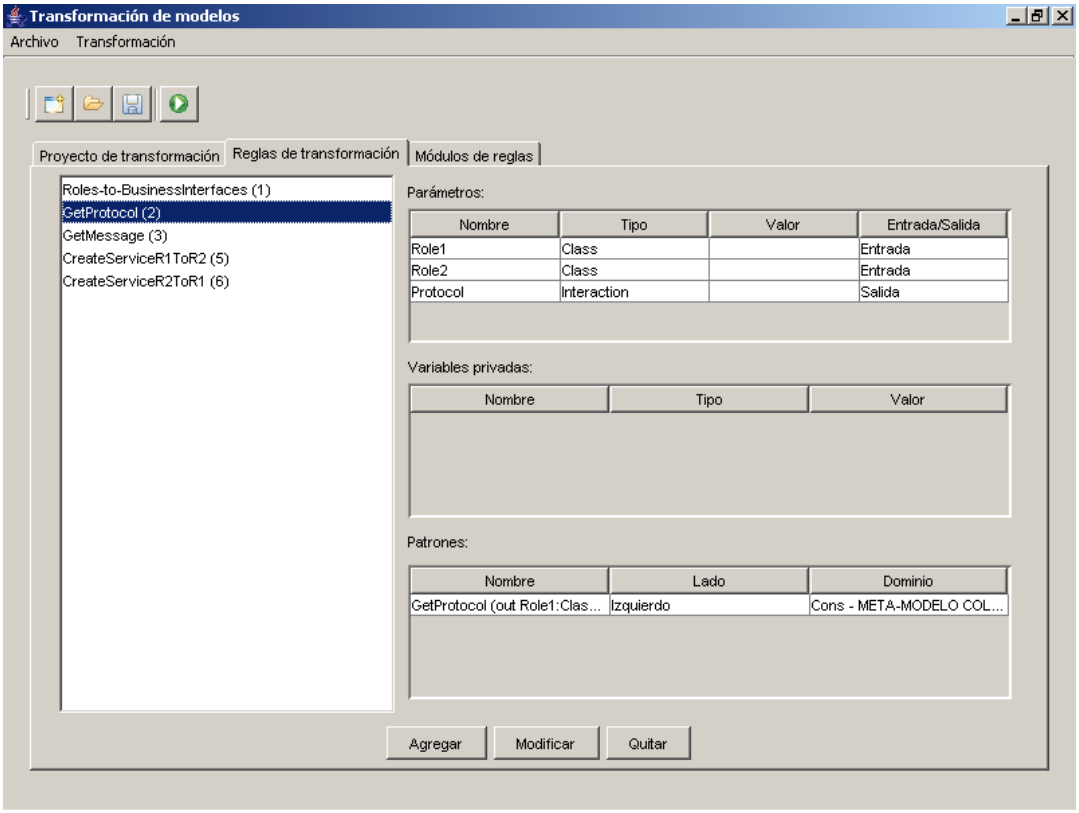


Figura 5-20. Reglas de transformación

Debido a que es posible utilizar variables definidas en los nodos ancestros o utilizar los parámetros definidos en el módulo de reglas, el campo *Nombre* contiene una lista desplegable con estos elementos. Por ejemplo, las variables *Role1* y *Role2* son heredadas del nodo padre de este nodo que se está definiendo. Si se requiere definir una nueva variable se escribe el nombre de la variable, el cual debe ser diferente a las ya definidas. Por ejemplo, la regla asociada a este nodo posee un parámetro de salida de tipo *Interaction*. La variable *IP* es creada para ser utilizada como argumento en dicho parámetro. El campo *Asignación* con el símbolo "<==" indica que la variable *IP* tomará el valor que retorne el parámetro de salida de la regla.

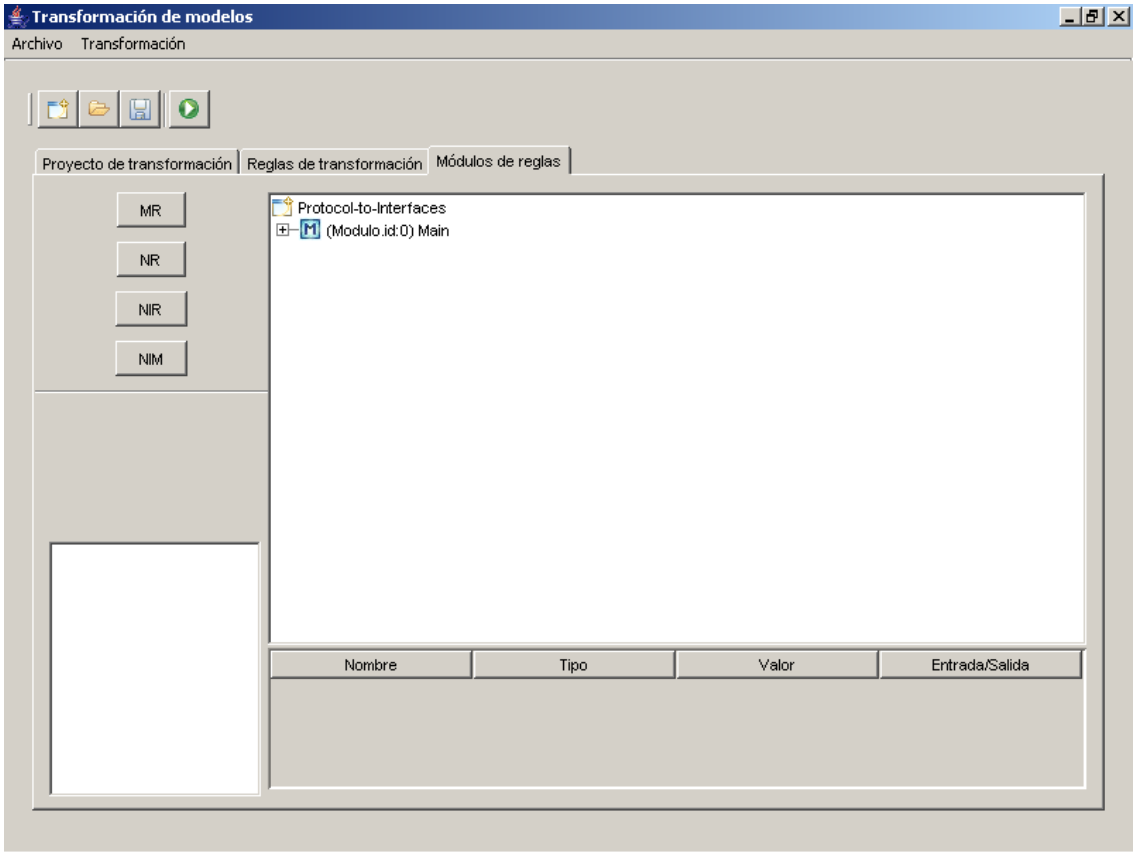


Figura 5-21. Módulos de reglas

La Figura 5-23 muestra la definición completa del modelo de transformación *Protocol-to-Interfaces*, el cual contiene un único módulo de regla. En este caso el árbol constituido en dicho módulo es muy sencillo. El módulo contiene el nodo raíz *Roles-to-BusinessInterfaces*, el cual es de tipo secuencial. Este nodo invoca la regla que genera las interfaces de negocio de los roles. La regla invocada por el subnodo *GetProtocol* permite obtener los protocolos de interacción definidos en el modelo, en los cuales participan los roles obtenidos anteriormente. Este nodo tiene definido el atributo *ForEach*, el cual significa que la regla es evaluada mientras sea exitosa, es decir mientras se obtenga un protocolo. Cuando un protocolo es encontrado se deben analizar los mensajes de negocio del mismo. La regla invocada en el subnodo *GetMessage* obtiene un mensaje. La regla de este nodo también será evaluada mientras se encuentre un mensaje en el protocolo. Por cada mensaje obtenido se deben derivar los servicios de negocio correspondientes en las interfaces de negocio de los roles. De acuerdo a la dirección del mensaje, uno de los subnodos del nodo *GetMessasge* será evaluado. Por lo tanto, este nodo es de tipo *OR*. Si el servicio ya fue generado, la evaluación de estos subnodos retornará falso y no se genera nuevamente el servicio. Esto se debe a que los

servicios pueden dar soporte al envío o recepción de mensajes en diferentes protocolos. La notación utilizada para cada uno de los nodos y los módulos de reglas es mostrada en la tabla 5-1.

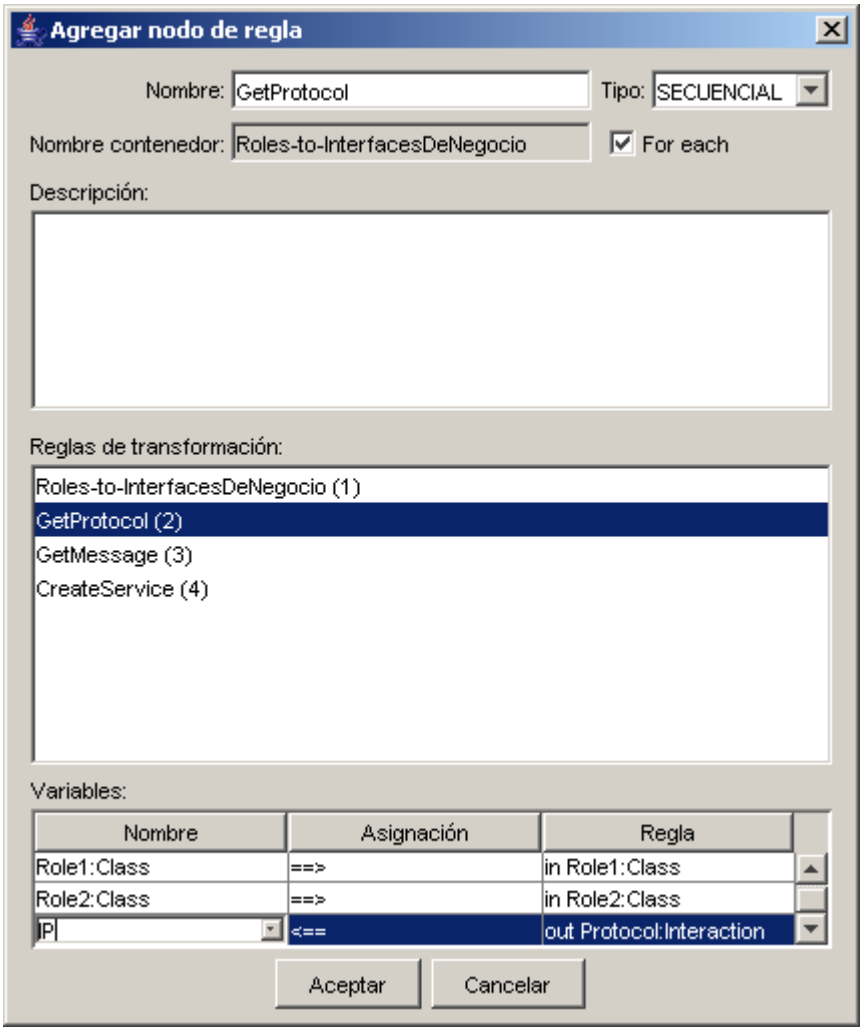


Figura 5-22. Creación de un nodo regla

Concepto del Lenguaje de Transformación de Modelos	Notación
Módulo de Regla	
Nodo Simple	
Nodo Secuencial	
Nodo OR	
Nodo AND	
Nodo de Invocación a Módulo	
Nodo de Invocación Recursiva	

Tabla 5-1. Notación para los nodos y módulos de reglas.

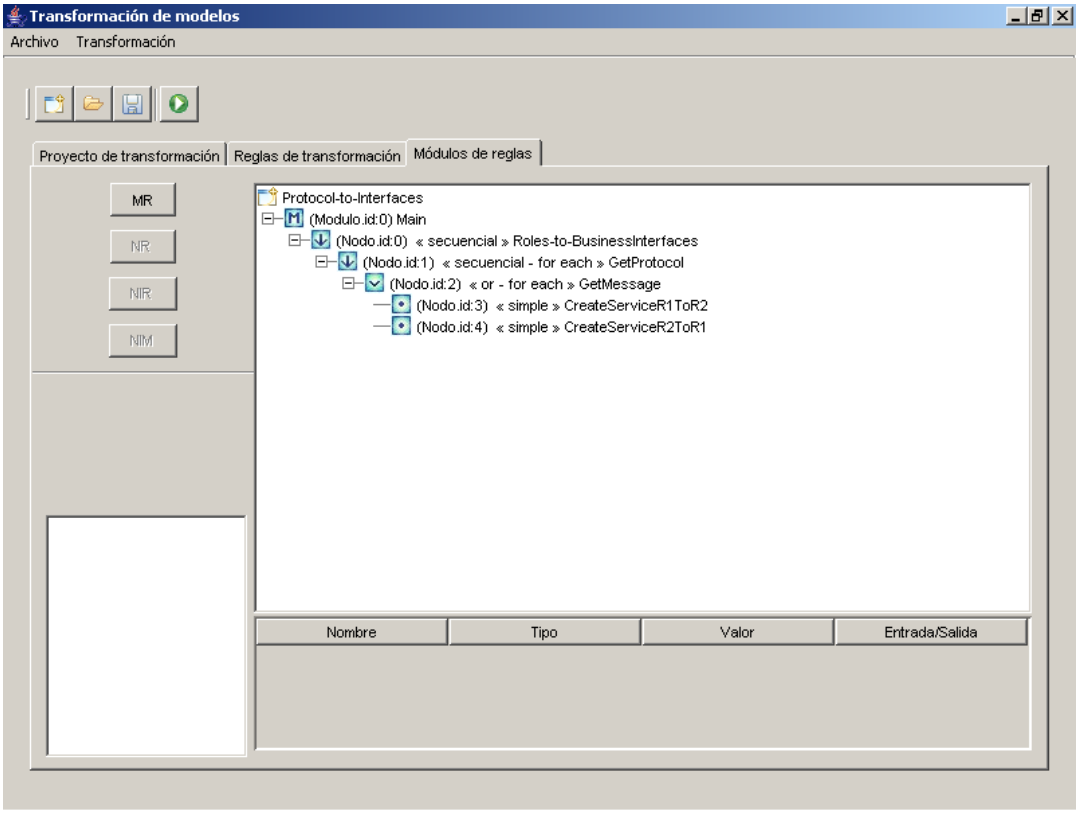


Figura 5-23. Definición del modelo de transformación *Protocol-to-Interfaces*

5.5. Etapas y Pasos para la Transformación de Modelos

El proceso a seguir para llevar a cabo una transformación de modelos con el método y la herramienta propuestos, puede dividirse en dos grandes etapas: especificación de la transformación y ejecución de la transformación.

La etapa de especificación involucra, en primer lugar, identificar las reglas simples y definir los patrones de las mismas utilizando el lenguaje de modelado de reglas. De acuerdo a la experiencia obtenida en la definición de transformaciones complejas, como las definidas en los capítulos 6 y 7, es más conveniente comenzar la definición de la transformación definiendo las reglas simples en forma separada. Es decir, para realizar transformaciones de MIPs a MEPs, es más adecuado comenzar la transformación pensando en la correspondencia entre un concepto de un lenguaje y uno o varios del lenguaje de destino.

Luego, se debe especificar la transformación en forma completa a través del editor de transformaciones de la herramienta, el cual soporta el lenguaje de transformación de modelos. Al mismo tiempo, o una vez definida la transformación en forma completa, se

debe proveer la implementación de cada uno de los patrones definidos en la herramienta.

Una vez definida la transformación e implementados los patrones, es posible pasar a la etapa de ejecución de la transformación. Esta etapa es soportada por las máquinas de transformación de modelos y de generación de código XML. Estos componentes de la herramienta permiten realizar la ejecución de la transformación en forma completamente automatizada sin la intervención del usuario. No obstante, para lanzar la ejecución, el usuario debe indicar el módulo de reglas inicial que desencadenará la ejecución de las reglas. Es posible seleccionar cualquiera de los módulos definidos en un proyecto de transformación. Además, el usuario debe especificar el nombre y ubicación del archivo XMI que contiene el modelo de entrada, y el nombre y ubicación del archivo XML que será generado.

De esta manera, la etapa de ejecución comienza realizando la transformación del modelo de origen en el modelo de destino, utilizando la máquina de transformación de modelos. Para ello este componente toma como entrada el proyecto que contiene la definición de la transformación a aplicar y el archivo XMI con el modelo de entrada (por ejemplo, un modelo UP-ColBPIP). La máquina interpreta la definición de la transformación y genera el modelo de destino (por ejemplo, un modelo ebXML). También genera como salida un log informando el orden en que los nodos y módulos fueron ejecutados, como así también el número de veces que las reglas fueron ejecutadas. Una vez generado el modelo de destino, el mismo es pasado a la máquina de generación de código XML, la cual genera el código XML correspondiente al modelo de salida (por ejemplo, las especificaciones ebXML en archivos XML).

5.6. Conclusiones

Debido a la carencia de generalidad de los métodos y herramientas de transformación de modelos actuales y sus limitaciones para hacer realidad transformaciones de modelos en el dominio de procesos de negocio colaborativos, se han identificado los requerimientos particulares para tal tipo de transformaciones. Principalmente, el objetivo es soportar las características requeridas para la transformación de modelos de procesos definidos con UP-ColBPIP a modelos de procesos basados en un estándar B2B como ebXML y BPEL.

Para manejar la complejidad de dichos requerimientos, se ha propuesto un método para dar soporte a la definición y ejecución de transformaciones en el dominio de procesos colaborativos. Dicho método consiste de varios componentes, algunos de los cuales están soportados por la herramienta de transformación de modelos propuesta.

El método y la herramienta permiten definir y ejecutar transformaciones de modelos independientes de la plataforma a modelos dependientes de la plataforma, y también ejecutar transformaciones de estos últimos a código XML. De esta manera, se ofrece un soporte al proceso completo requerido para realizar el desarrollo conducido por modelos en el dominio de procesos colaborativos. Además, el método y la herramienta también podrían ser utilizados para realizar refinamientos de modelos independientes de la plataforma, en la etapa de modelado conceptual de procesos.

La aplicabilidad tanto del método como de la herramienta ha sido probada en las transformaciones de modelos de procesos de UP-ColBPIP a especificaciones de procesos basadas en ebXML BPSS, como se verá en el siguiente capítulo.

El método propuesto consiste de dos lenguajes, el lenguaje de transformación de modelos y el lenguaje de modelado de reglas. El primero es el lenguaje principal, y a través del mismo es posible definir una transformación completa. Este lenguaje, soportado completamente por la herramienta. Ofrece una forma declarativa y visual de definir las transformaciones, y a la vez es ejecutable. La herramienta provee los componentes necesarios para interpretar y ejecutar las transformaciones definidas en dicho lenguaje. A través del editor gráfico es posible declarar las reglas con sus patrones, y componer a las mismas dentro de módulos de reglas utilizando una notación de árbol. La notación en forma de árbol de la composición de las reglas como así también de la estructura de control de las mismas, facilita la comprensión de la transformación completa.

La principal ventaja de este lenguaje es la separación de la definición de las reglas, de la composición y estructura de control de las mismas. Esto posibilita la reutilización de las reglas en varias partes de la transformación, sin requerir la definición de regla complejas. Por otra parte, la capacidad para componer las reglas simples en una jerarquía de nodos dentro de los módulos de reglas incrementa la legibilidad, modularidad y el mantenimiento de transformaciones complejas. Además, ofrece un mecanismo de ordenamiento explícito.

La implementación de los patrones utilizando el lenguaje de programación Java a través de la API provista, tiene como ventaja que los programadores poseen más experiencia en este tipo de lenguajes. Además, es posible implementar diferentes mecanismos de búsqueda en los patrones, de acuerdo a la complejidad de los modelos a manipular. No obstante, como contrapartida esto tiene la desventaja de que sólo las personas experimentadas en tal lenguaje pueden definir transformaciones, debido a que no se proveen mecanismos de mayor abstracción. Además, la implementación de los patrones por parte del desarrollador no garantiza que se provean implementaciones libres de errores. Por otra parte, el entendimiento de la lógica de los patrones es más difícil, como así también el desarrollo y mantenimiento de los patrones.

Para salvar algunos de estos inconvenientes, se ha propuesto el lenguaje de modelado de reglas, el cual permite definir las reglas con sus patrones utilizando una notación gráfica conforme a UML. El lenguaje está basado en la teoría de gramáticas y transformación de grafos, lo cual permite que la semántica operacional de la definición de una regla pueda ser entendida en forma intuitiva y clara. De esta manera, es posible implementar los patrones siguiendo la semántica operacional que ofrece este lenguaje.

Los lenguajes y técnicas propuestas por el método de transformación de modelos están altamente acoplados a los principios establecidos en la iniciativa MDA para el desarrollo conducido por modelos. El lenguaje de transformación de modelos ha sido construido a través de un metamodelo basado en MOF. El lenguaje de modelado de reglas permite definir los patrones de las mismas utilizando diagramas de UML. Para definir las reglas de transformación se requiere la aplicación de la técnica de metamodelación, ya que los patrones son definidos teniendo en cuenta la sintaxis abstracta de los lenguajes de modelado.

Por otra parte, la herramienta desarrollada es completamente independiente de las herramientas Case UML, como así también de los ambientes de metamodelación. Permite transformar modelos UP-ColBIP almacenados en un formato XMI. Además, se han incorporado a la herramienta las APIs necesarias para la manipulación de los modelos de entrada y salida, las cuales fueron generadas con otro ambiente de metamodelación. Tanto las APIs como los modelos de entrada almacenados en formato XMI se presupone que están bien formados y son consistentes con los metamodelos correspondientes. Actualmente no se proveen mecanismos para el chequeo de

consistencia de los modelos de entrada y salida, como así también de los patrones, con respecto a sus correspondientes metamodelos.

Finalmente, aunque el método y la herramienta fueron diseñados para llevar a cabo transformaciones de modelos en el dominio de procesos colaborativos, los mismos son bastante generales y en principio podrían ser utilizados para realizar transformaciones en otros dominios. No obstante, se debería verificar previamente si los requerimientos de transformación de modelos de estos dominios son soportados en el método.

Capítulo

6

TRANSFORMACIÓN DE MODELOS UP-COLBPIP A EBXML

En este capítulo, en primer lugar se describe el estándar ebXML. En particular se describen los lenguajes BPSS y CPPA provistos por este estándar. Luego como parte del método de desarrollo conducido por modelos de procesos colaborativos, se describen: la transformación de modelos UP-ColBPIP a BPSS y la transformación de modelos UP-ColBPIP a CPPA. Parte de la salida final de dichas transformaciones, correspondiente al código XML que generan, es descrita utilizando como modelo de entrada el modelo UP-ColBPIP definido en el capítulo 4. Finalmente se presentan las conclusiones.

6.1. El Estándar ebXML

Una de las tecnologías disponibles para dar soporte a la ejecución de los procesos colaborativos en una colaboración B2B es ebXML. El estándar ebXML (Electronic Business XML) (OASIS, 1999) consiste de un conjunto de lenguajes (también conocidos como especificaciones) que permiten llevar a cabo interacciones B2B entre los componentes autónomos, heterogéneos y distribuidos de los socios. Estos lenguajes soportan la interoperabilidad en las diferentes capas en las cuales estos componentes interactúan: comunicación, documentos de negocio y procesos de negocio. Para dar soporte a la capa de comunicación, ebXML provee un lenguaje para el intercambio de mensajes (Message Service Specification). Para dar soporte a la capa de contenido, el lenguaje especificación *Core Component* provee los componentes que permiten definir la estructura de los documentos de negocio, a través de los cuales los socios intercambian información. Para dar soporte a la capa de procesos de negocio, ebXML provee el lenguaje Business Process Specification Schema (BPSS) para definir procesos de negocio. Además, en esta capa también provee el lenguaje Collaboration-Protocol Profile and Agreement (CPPA), la cual permite definir los aspectos operacionales de las interfaces de los socios. Además, ebXML propone una infraestructura basada en un *repositorio (Registry Services Specification)*, en el cual los socios pueden registrar en un

lugar compartido las definiciones de procesos, documentos, servicios y acuerdos entre los socios.

Debido a que los lenguajes BPSS y CPPA son los que dan soporte a la capa de procesos de negocio, estos son los lenguajes más relevantes para el propósito de generar especificaciones en ebXML que permitan la ejecución de los procesos de negocio colaborativos. A continuación se describen los principales conceptos de los mismos.

6.1.1. El Lenguaje ebXML Business Process Specification Schema

Para que dos o mas empresas puedan realizar la ejecución de los procesos colaborativos, las definiciones de estos procesos deben ser compartidas y entendidas de igual manera por los componentes B2B de los socios. El lenguaje ebXML Business Process Specification Schema (BPSS) (UN/CEFACT y OASIS, 2003) da soporte a este requerimiento. Este lenguaje permite definir especificaciones ejecutables de procesos de negocio colaborativos, las cuales están basadas en XML y pueden ser procesadas por software.

La estructura del lenguaje es definida en un esquema XML. No obstante, también se provee una versión UML de dicho esquema a través de un diagrama de clase. Este diagrama es utilizado en nuestro método de desarrollo conducido por modelos como el metamodelo del lenguaje BPSS, con el cual es posible construir modelos de procesos basados en BPSS. El metamodelo de BPSS es mostrado en la Figura 6-1. Este hace referencia a la versión 1.10 del lenguaje. A continuación se describen cada uno de los principales conceptos provistos en este lenguaje.

BPSS utiliza el concepto de *Business Collaboration (Colaboración de Negocio)* para representar procesos de negocio colaborativos. Una *Business Collaboration* consiste de un conjunto de roles que interactúan a través de *Business Transactions (Transacciones de Negocio)* dentro de las cuales intercambian *BusinessDocuments (Documentos de Negocio)*. Existen dos tipos de colaboraciones de negocio: *Multiparty Collaboration*, la cual es entre más de dos roles, y *Binary Collaboration (Colaboración Binaria)*, la cual es entre dos roles. Colaboraciones de tipo *Multiparty Collaboration* siempre son sintetizadas en dos o más colaboraciones binarias. No obstante, este concepto será dejado de lado en las nuevas versión del estándar y por lo tanto sólo podrán ser definidas colaboraciones binarias.

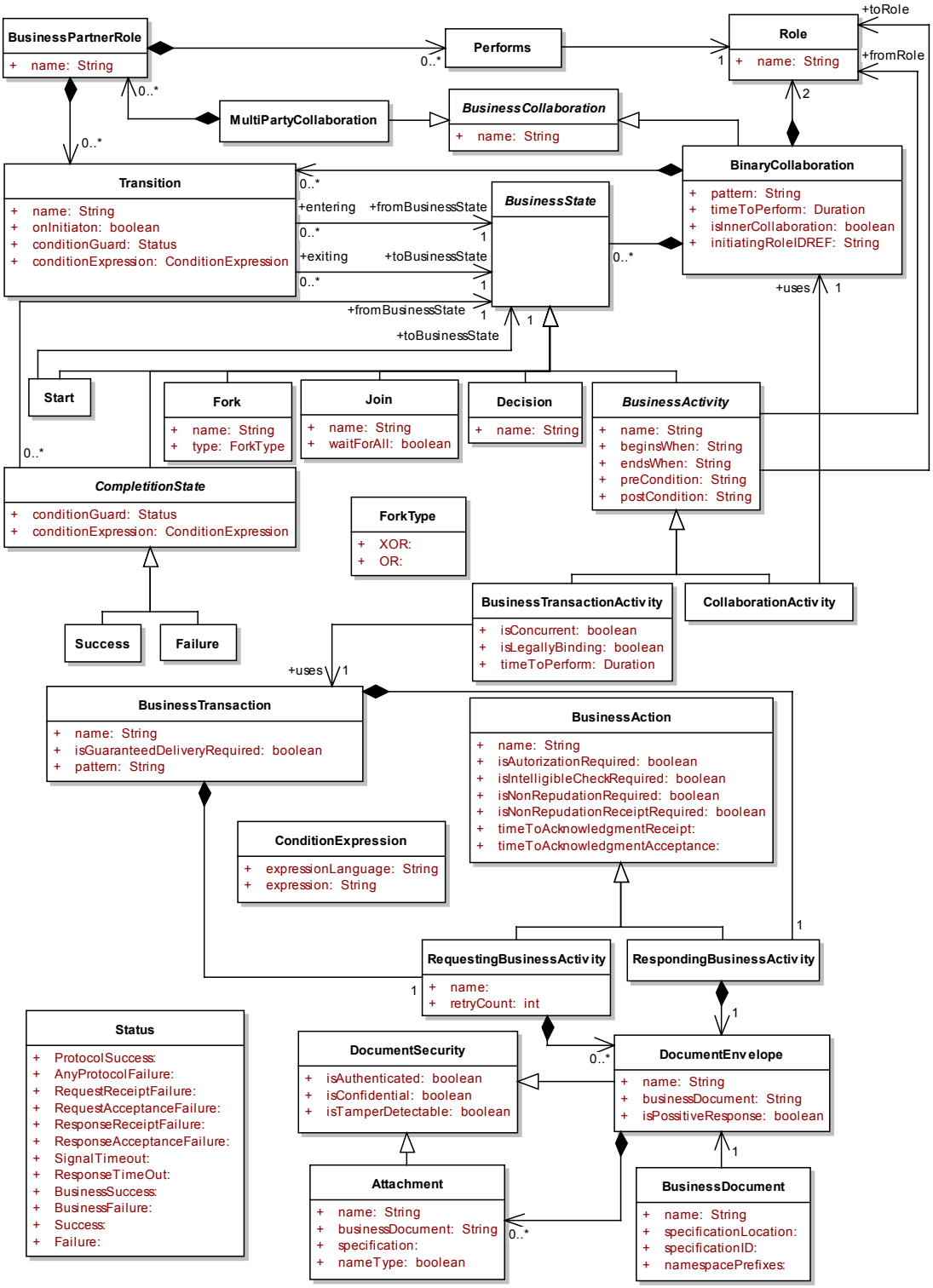


Figura 6-1. Metamodelo de BPSS

Una *Binary Collaboration* es definida como un conjunto de actividades de negocio entre dos roles. Una *Business Activity (Actividad de Negocio)* puede ser de dos tipos: una *Business Transaction Activity (Actividad de Transacción de Negocio)*, la cual

representa la ejecución de una transacción de negocio; o una *Collaboration Activity (Actividad de Colaboración)*, la cual representa la ejecución de otra colaboración binaria. Esta última permite la definición de colaboraciones binarias anidadas o subcolaboraciones. Una colaboración binaria podría ser referenciada en varias colaboraciones binarias a través de una actividad de colaboración definida en estas últimas. De la misma manera, una transacción de negocio podría ser ejecutada por diferentes actividades de transacción de negocio, definidas en diferentes colaboraciones binarias o en la misma colaboración binaria. De esta manera, transacciones de negocio y colaboraciones binarias pueden ser reutilizadas en diferentes colaboraciones binarias.

Una *Business Transaction (Transacción de Negocio)* representa la unidad atómica de trabajo en BPSS y es el principal bloque de construcción de un proceso colaborativo. Esta consiste de una *Requesting Activity (Actividad de Solicitud)* y una *Responding Activity (Actividad de Respuesta)*, y uno o dos flujos de *Business Documents (Documentos de Negocio)* entre estas actividades. Existe siempre un documento de negocio de solicitud, y opcionalmente cero, uno o más documentos de negocio de respuesta. Es decir, una actividad de solicitud representa el envío de un documento y una actividad de respuesta representa el envío de cero o más documentos de negocio. Ambas actividades poseen atributos comunes heredados de la clase *BusinessAction*, los cuales tienen que ver con el manejo de las denominadas señales de negocio. Estos atributos especifican condiciones, como el tiempo para el acuse de recibo de un documento, la no-repudiación de documentos, la autorización requerida y el chequeo requerido de documentos. La Figura 6-2 muestra los flujos de documentos de negocio y las señales de negocio posibles en una transacción de negocio. Estos flujos dependen del valor de los anteriores atributos.

Una transacción de negocio no especifica cuáles de los roles de una colaboración binaria están involucrados en la transacción. Estos roles son especificados en la actividad de transacción de negocio que ejecuta la transacción de negocio, como lo indican los atributos de asociación *fromRole* y *toRole* de la misma. De esta manera, es posible que una transacción de negocio sea referenciada por dos o más actividades de transacción de negocio, en donde los roles estén invertidos.

Una colaboración binaria también describe el orden de las actividades de colaboración a través de una coreografía. En forma similar a una máquina de estado, la

coreografía es especificada en términos de *Business States (Estados de Negocio)* y *Transitions (Transiciones)* entre estos estados. Una transición puede contener condiciones (*conditionGuards*) que restringen cuándo un estado puede continuar luego de otro. El tipo de estado más importante es la actividad de negocio. Los restantes estados corresponden a los conceptos de flujo de control, tales como: estado de comienzo (*Start*), estado de terminación (*CompletionState*), estado *Fork*, estado *Join,* y estado de decisión (*Decision*). Existen dos tipos de estado de terminación: *Success* y *Failure*. Estos indican una terminación exitosa o con falla, respectivamente. La terminación con falla sólo es una indicación de que la lógica de la colaboración finalizó en un cierto estado, la cual puede ser utilizada para ser reportada o para iniciar otra colaboración.

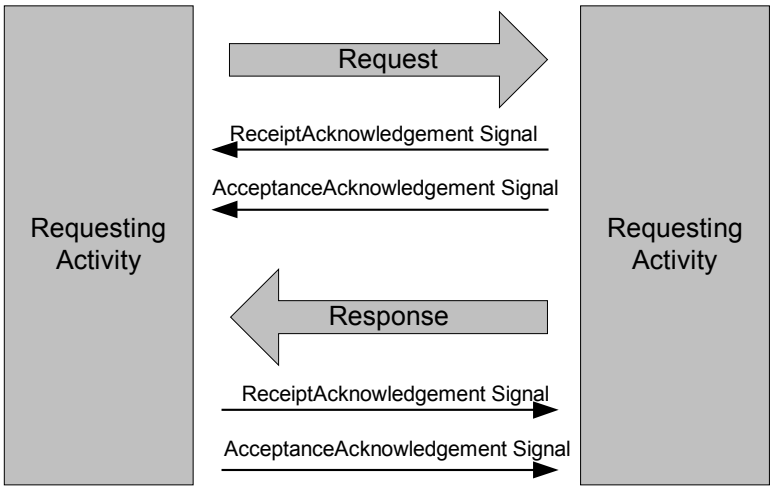


Figura 6-2. Flujos de documentos y señales en una *Transacción de Negocio*

Los conceptos de *Start*, *Success* y *Failure*, a pesar de que son tipos de estados de negocio, los mismos representan una agregación de un estado y una transición. Es decir que sus atributos de asociación representan una transición desde (estados *Success* y *Failure*) o hacia (estado *Start*) otros estados de negocio. En el caso de los estados de terminación, los mismos pueden tener condiciones que indican cuándo una transición debe ocurrir hacia dichos estados.

6.1.2. El Lenguaje Collaboration-Protocol Profile and Agreement

Para soportar la ejecución de los procesos colaborativos, también se requiere definir la compatibilidad entre las interfaces de los Sistema de Administración de Procesos Colaborativos (SAPC) de los socios, en términos de los mensajes que cada socio debe

enviar y recibir. Además, las partes deben acordar diferentes parámetros a ser tenidos en cuenta durante la ejecución de tales procesos, como ser el protocolo de comunicación (ej: HTTP o SOAP) a utilizar y otras propiedades de seguridad. El lenguaje Collaboration-Protocol Profile and Agreement (CPPA) (OASIS, 2002) da soporte a estos requerimientos.

Este lenguaje permite definir para cada socio un Collaboration Protocol Profile (CPP), y además permite definir un Collaboration Protocol Agreement (CPA) común a ambos. La manera en que las partes intercambian información, ejecutando colaboraciones de negocio definidas en una o más especificaciones de proceso en BPSS, es definido en los CPP de cada socio. El acuerdo entre las partes es expresado en un CPA. El acuerdo se refiere a detalles técnicos como los protocolos de comunicación a utilizar en los CPP de cada socio.

Un CPP describe las capacidades de negocio y técnicas de un socio, con respecto al rol que éste desempeña en las colaboraciones de negocio definidas en una especificación de procesos BPSS. Las capacidades de negocio se refieren a las colaboraciones de negocio que son soportadas por el socio. Las capacidades técnicas se refieren a los parámetros de configuración con respecto a los protocolos de comunicación que el sistema del socio soporta. Cuatro elementos en un CPP definen estas capacidades: *ProcessSpecification*, *DeliveryChanel*, *DocExchange* y *Transport*. El primero es relevante para el soporte en la capa de procesos de negocio. Los restantes corresponden a la capa de comunicación y de transporte. Por lo tanto, estos últimos no son descritos en esta tesis, ya que los mismos no pueden ser derivados a partir de modelos independientes de la tecnología. La Figura 6-3 muestra el metamodelo del lenguaje CPPA para definir los perfiles de protocolo de colaboración (CPPs). El metamodelo sólo contiene los elementos de CPPA que pueden ser generados a partir de modelos independientes de la tecnología y que dan soporte a la capa de procesos de negocio.

Un CPP consiste de una *PartyInfo*, en donde se indica el nombre del socio y el cual está compuesto de un elemento *CollaborationRole*. Este elemento contiene la información correspondiente a la interfaz de un socio de negocio. El elemento *Role* define el rol que el socio va a desempeñar en una especificación de procesos. El rol debe corresponderse con uno de los roles definidos dentro de una especificación de

procesos BPSS. Esta especificación es referenciada en un elemento *ProcessSpecification*. Un rol contiene una definición de los servicios (transacciones de negocio) que el socio soporta. Esto es representado por un elemento *ServiceBinding*. Este elemento especifica los mensajes que el socio puede recibir (elemento *CanReceive*) y los mensajes que el socio puede enviar (elemento *CanSend*), de acuerdo a las actividades de solicitud y respuesta de las transacciones de negocio, como así también las señales de negocio de las mismas.

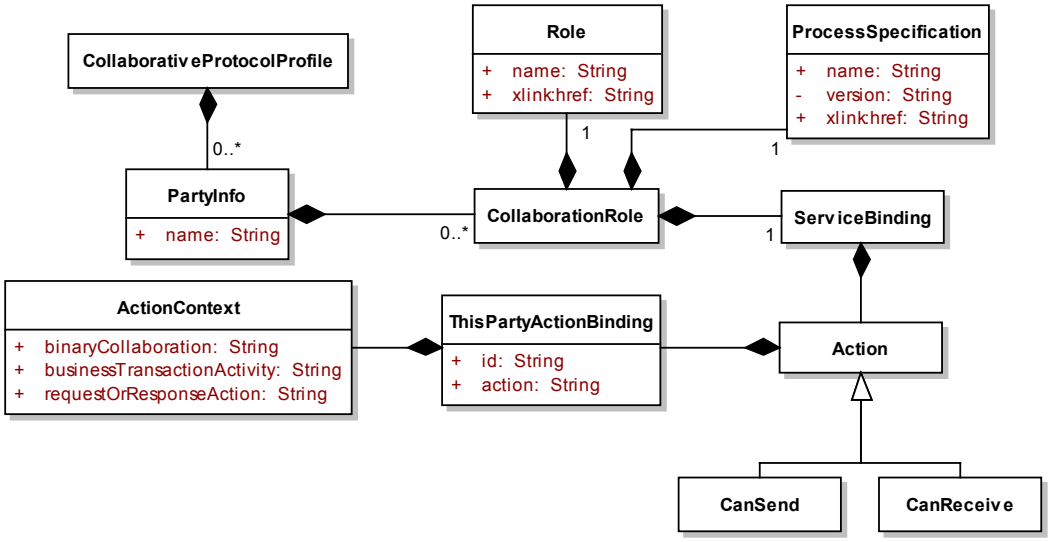


Figura 6-3. Metamodelo de CPPA

Un *ServiceBinding* puede contener uno o más elementos *CanSend* y *CanReceive*. El primero representa una acción a través de la cual un socio es capaz de enviar un mensaje. El último representa una acción a través de la cual un socio es capaz de recibir un mensaje. Ambos elementos están compuestos de un elemento *ThisPartyActionBinding*. El mismo contiene un elemento *ActionContext*¸ el cual posee tres atributos. El atributo *binaryCollaboration* contiene el nombre de la colaboración binaria, definida en la especificación de procesos BPSS, que es soportada. El atributo *businessTransactionActivity* contiene el nombre de la transacción de negocio que es soportada. El atributo *requestOrRespondeAction* contiene el nombre de la actividad de solicitud o de respuesta, definida en la transacción, que es soportada. En el caso de referirse al contexto de una acción que recibe un mensaje (*CanReceive*), el nombre de este atributo debe corresponderse con el nombre de la actividad de respuesta de la transacción de negocio referenciada en el atributo *businessTransactionActivity*. En el caso contrario una acción para enviar un mensaje, el atributo *requestOrRespondeAction*

debe corresponderse con el nombre de la actividad de solicitud de la transacción de negocio referenciada en el atributo *businessTransactionActivity*

6.1.3. Ventajas de ebXML como Solución Tecnológica para dar Soporte a Procesos Colaborativos

Existen varias propiedades de estos lenguajes provistos por ebXML, que sitúan a esta tecnología como un candidato adecuado para desarrollar soluciones tecnológicas que soporten la ejecución de procesos colaborativos.

Por un lado, BPSS permite especificar procesos colaborativos abstractos, es decir, éste no trata con la definición de los detalles privados de los socios. Los detalles de la lógica interna de cada rol, acerca de cómo un documento recibido es procesado o cómo un documento enviado es generado, no forman parte de una colaboración binaria. Por otra parte, colaboraciones binarias y transacciones de negocio proveen una vista global y común de las interacciones entre los socios. Una especificación de procesos en BPSS muestra el comportamiento de la colaboración desde un punto de vista común a todos los socios y por lo tanto, permite especificar interacciones peer-to-peer entre las partes.

Con este enfoque específico sobre colaboraciones globales, la interfaz de cada participante puede ser configurada con CPPA desde las mismas definiciones de colaboraciones acordadas entre las partes. Esto permite que las interfaces de los socios sean compatibles e interoperables de acuerdo a los procesos definidos.

De esta manera, en una solución tecnológica basada en ebXML, el SAPC de cada socio deberá ser desarrollado para interpretar definiciones en BPSS y CPP. Con esta tecnología, cada SAPC puede ser definido en forma completamente independiente.

Muchas de estas características de ebXML han sido identificadas como esenciales para el modelado conceptual de procesos colaborativos y son soportadas en dicho nivel por el lenguaje UP-ColBPIP propuesto. Por lo tanto, soluciones tecnológicas basadas en ebXML son adecuadas para ser derivadas a partir de lo definido en el nivel de negocio a través de modelos de procesos colaborativos definidos con UP-ColBPIP.

6.2. Transformación de UP-ColBPIP a ebXML

Siguiendo los lineamientos del método propuesto basado en el desarrollo conducido por modelos para procesos colaborativos (Sección 3.3.2), la generación de

especificaciones basadas en ebXML a partir de modelos conceptuales basados en UP-ColBPIP puede ser vista como se muestra en la Figura 6-4.

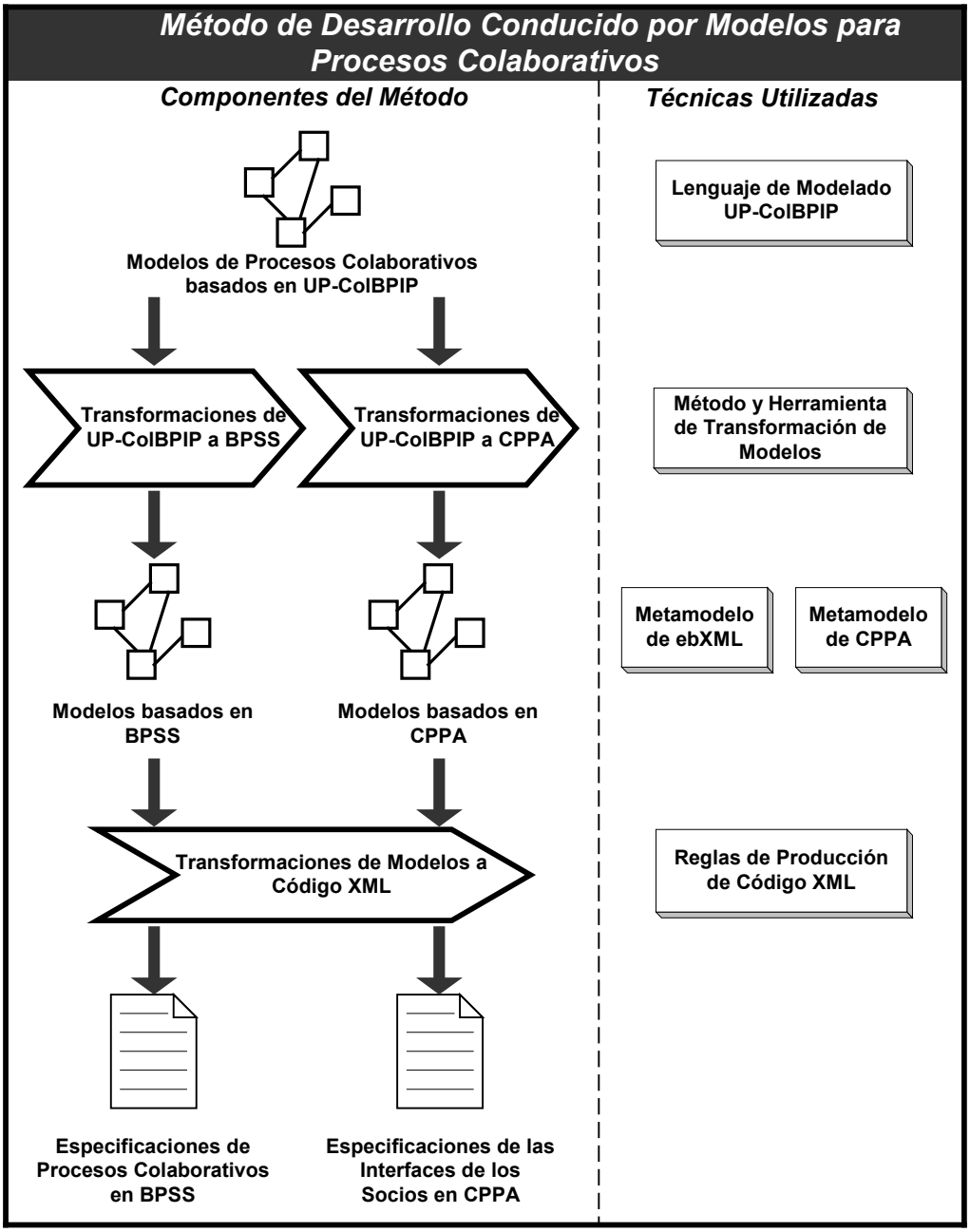


Figura 6-4. Técnicas y componentes del método de desarrollo conducido por modelos para la generación de soluciones tecnológicas basadas en ebXML

Los modelos independientes de la plataforma son aquellos modelos de procesos colaborativos definidos con UP-ColBPIP. Los modelos dependientes de la plataforma son aquellos basados en ebXML. En particular, dos tipos de modelos dependientes de la plataforma deben ser tenidos en cuenta para dar soporte a la capa de procesos de

negocio: aquellos modelos de procesos colaborativos basados en BPSS, y aquellos modelos basados en CPPA, los cuales definen los aspectos operacionales de las interfaces de los socios. Ambos tipos de modelos pueden ser construidos a través de los metamodelos descritos en la sección anterior. A partir de un modelo UP-ColBPIP se generará un modelo BPSS. En cambio, debido a que un modelo basado en CPPA define la interfaz de un único socio, se deben generar varios modelos CPP por cada uno de los socios involucrados en la colaboración B2B. A partir de estos modelos basados en ebXML, se generan las especificaciones de los mismos en documentos XML.

Las transformaciones de un modelo UP-ColBPIP a BPSS y a CPPA son definidas y ejecutadas en forma separada. Esto se debe a que BPSS y CPPA son dos lenguajes diferentes, y por lo tanto sus metamodelos también son diferentes. Aunque las transformaciones a estos lenguajes podrían ser integradas en una misma transformación, la salida final debe consistir de varios documentos XML, uno correspondiente a la especificación de los procesos en BPSS y los restantes correspondientes a las especificaciones CPP, una para cada socio. La separación de estas transformaciones es más adecuada ya que posibilita un proceso de desarrollo más flexible y manejable.

Debido a que UP-ColBPIP da soporte no sólo la etapa de diseño conceptual de procesos colaborativos, sino también a la etapa de análisis, no todos los conceptos utilizados en UP-ColBPIP serán transformados en conceptos de los lenguajes de ebXML. Más aún, la mayoría de los conceptos utilizados en UP-ColBPIP para definir la vista de la colaboración B2B y la vista de los procesos colaborativos no deberían ser transformados. Esto se debe a que los mismos sólo son provistos para llevar a cabo la etapa de análisis y por lo tanto seguramente carecerán de una correspondencia en el nivel tecnológico. De esta manera, dentro de un modelo UP-ColBPIP, las principales vistas a partir de las cuales es posible generar una solución tecnológica basada en ebXML son: la vista de los protocolos de interacción y las vistas de las interfaces de negocio. Los conceptos de BPSS y CPPA podrán ser generados a partir de los conceptos provistos por UP-ColBPIP para definir estas vistas de un modelo.

La aplicación de este método de desarrollo conducido por modelos posibilita:

- La generación de una solución tecnológica basada en ebXML para ejecutar procesos colaborativos, a partir de modelos conceptuales basados en UP-ColBPIP.

- La garantía de que la solución tecnológica basada en ebXML es consistente y se corresponde con lo definido en el nivel de negocio.
- La reducción de la distancia semántica entre los conceptos provistos por UP-ColBPIP en el nivel de negocio y los conceptos provistos por ebXML en el nivel tecnológico.

A continuación se describen cómo estas transformaciones han sido definidas e implementadas. Las mismas han sido llevadas a cabo a través del método y la herramienta de transformación de modelos definidos en el capítulo anterior. En primer lugar se describe la transformación de modelos UP-ColBPIP a especificaciones en BPSS. Posteriormente se describe la transformación de modelos UP-ColBPIP a especificaciones en CPPA.

6.2.1. Transformación de UP-ColBPIP a BPSS

Los principales conceptos de UP-ColBPIP a partir de los cuales es posible generar especificaciones de procesos colaborativos en BPSS, son aquellos utilizados en la vista de los protocolos de interacción. Por lo tanto, la mayoría de los elementos de BPSS podrán ser derivados a partir de dicha vista de los modelos UP-ColBPIP.

La transformación de UP-ColBPIP a BPSS ha sido definida utilizando el método y la herramienta de transformación de modelos descritos anteriormente (Capítulo 5). Por un lado, se han definido las reglas de transformación utilizando el lenguaje de modelado de reglas, las cuales describen cómo uno o más conceptos de UP-ColBPIP son transformados en uno o más conceptos de BPSS. De esta manera se definen las correspondencias entre los conceptos de UP-ColBPIP y los de BPSS. Estas reglas representan la gramática de grafos definida para transformar modelos basados en UP-ColBPIP a modelos basados en BPSS.

Por otro lado, para definir y ejecutar las transformaciones se ha definido un modelo de transformación utilizando la herramienta de transformación de modelos, y se ha provisto una implementación en Java de cada uno de los patrones de las reglas.

El modelo de transformación UP-ColBPIP-to-BPSS consiste de cuatro módulos (Figura 6-5). El módulo *Main* es el principal. A través de dicho módulo se desencadena el proceso de transformación completo. Básicamente, este módulo realiza la transformación de la colaboración B2B, los documentos de negocio, y obtiene los

protocolos de interacción a transformar. La transformación de estos últimos es realizada en el módulo *InteractionProtocols-to-BinaryCollaboration*. Éste realiza la transformación de los roles, agrega los estados de terminación en la colaboración binaria, e invoca al módulo *Fragments-to-BusinessStates*. Este módulo lleva a cabo la transformación de los diferentes tipos de elementos que componen un protocolo de interacción, como así también los que componen a los caminos de los segmentos de flujo de control. El módulo *CreateTransition* es invocado dentro del anterior, y es el encargado de generar las transiciones entre los estados generados en las colaboraciones binarias. El módulo *Fragments-to-BusinessStatesWithoutTransitions* tiene la misma estructura del módulo *Fragments-to-BusinessStates*, excepto que luego de transformarse un elemento, no se invoca al módulo *CreateTransition* para crear la transición entre los estados correspondientes de las colaboraciones binarias. Este módulo sólo transforma un único elemento, el cual es pasado como parámetro de entrada del mismo. En cambio, el módulo *Fragments-to-BusinessStates* realiza la transformación de los elementos de un protocolo o camino de interacción, mientras encuentre un elemento en los mismos.

Figura 6-5. Módulos del modelo de transformación *UP-ColBPIP-to-BPSS*

En las siguientes subsecciones se describen, según el concepto de UP-ColBPIP a transformar, las reglas que componen estos módulos y el árbol de reglas de los mismos. La semántica operacional de las reglas es descrita para las principales transformaciones.

Por otra parte, para ejemplificar la salida final de estas transformaciones, se describen algunas partes del código XML que corresponden a la especificación BPSS generada a partir del modelo UP-ColBPIP que ha sido definido en el caso de estudio del capítulo 4 (Sección 4.4). Este modelo contiene los procesos colaborativos basados en protocolos de interacción que permiten llevar a cabo un reaprovisionamiento colaborativo entre dos empresas basado en un modelo VMI. La Figura 6-6 muestra las entradas y salidas utilizadas en el proceso de transformación realizado para la generación de la especificación en BPSS a partir de dicho modelo UP-ColBPIP.

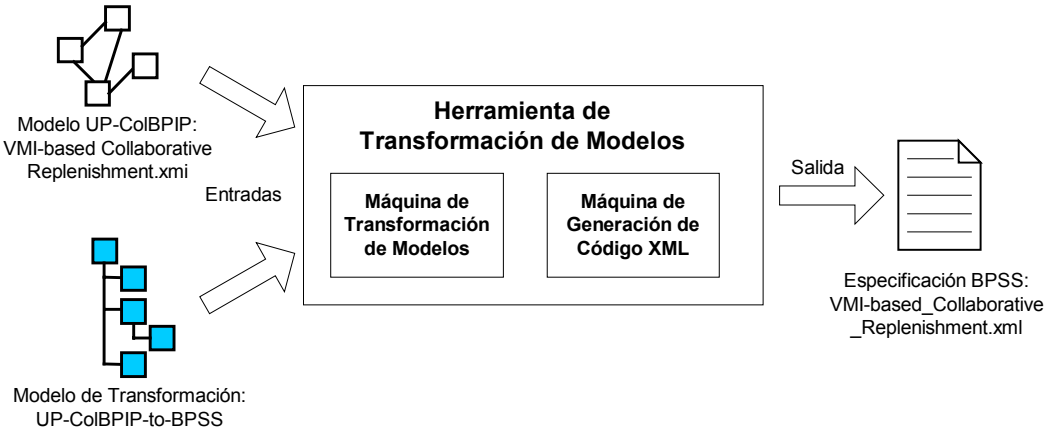


Figura 6-6. Entradas y salidas del ejemplo de transformación de UP-ColBPIP a BPSS

6.2.1.1. Transformación de la Colaboración B2B

El primer elemento a generar en un modelo BPSS es la especificación de procesos (*ProcessSpecification*), dentro del cual se adicionan los diferentes elementos que definen una especificación de procesos en BPSS. Este concepto es derivado a partir de la colaboración B2B definida en el modelo UP-ColBPIP, como lo indica la regla *B2BCollaboration-to-ProcessSpecification* (Figura 6-7).

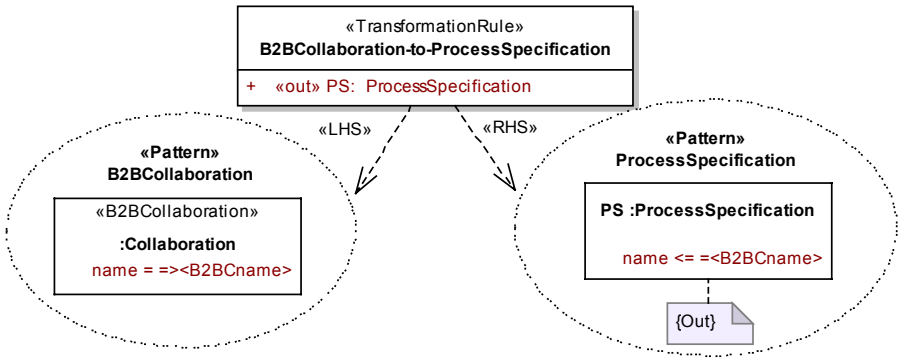


Figura 6-7. Regla de transformación *B2BCollaboration-to-ProcessSpecification*

La regla define que para un objeto de tipo *B2BCollaboration* encontrado en un modelo UP-ColBPIP, un objeto de tipo *ProcessSpecification* es generado en un modelo de BPSS. El nombre del objeto *ProcessSpecification* es el mismo que el de la colaboración B2B encontrada. El objeto *ProcessSpecification* generado es un parámetro de salida de la regla. El mismo será utilizado como parámetro de entrada en otras reglas, para adicionar a dicho objeto los documentos de negocio, las transacciones de negocio y las colaboraciones binarias que sean generadas.

Esta es la primer regla a ser ejecutada en la transformación, como se indica en el módulo *Main* (Figura 6-8). Luego de la ejecución de la misma se deben generar los documentos de negocio y las colaboraciones binarias. Esto es realizado en los subnodos del nodo *B2BCollaboration-to-ProcessSpecification*.

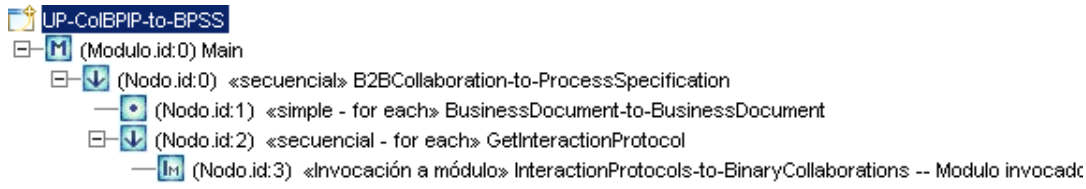


Figura 6-8. Árbol de reglas del módulo *Main*

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

En este caso, el código generado con esta regla en la transformación del modelo UP-ColBPIP de ejemplo, es el siguiente:

```
<ProcessSpecification name = "VMI-based Collaborative Replenishment" />
```

6.2.1.2. Transformación de Documentos de Negocio

En BPSS, los documentos de negocio son definidos en forma separada de la definición de las colaboraciones de negocio, debido a que estos pueden ser utilizados en varias de ellas. Además, BPSS no soporta la especificación de documentos de negocio, sino que sólo apunta a las definiciones de los documentos de negocio, las cuales están almacenadas en otro documento XML. En UP-ColBPIP, los documentos de negocio también son definidos en forma separada de los procesos colaborativos, debido a que ellos pueden ser utilizados en varios procesos. Además, UP-ColBPIP tampoco se enfoca en la definición de la estructura de los documentos, sino que solamente se identifican los documentos a ser utilizados. Por lo tanto, la regla *BusinessDocument-to-BusinessDocument* define que un documento de negocio de UP-ColBPIP, es transformado en un documento de negocio en BPSS (Figura 6-9).

En UP-ColBPIP, un documento de negocio tiene asociado un tipo de documento de negocio, el cual tiene un atributo que indica el documento esquema XML que define la estructura del componente. El valor de este atributo es asignado en BPSS al atributo *specificationLocation*, en el cual se indica el esquema XML del documento. El patrón RHS de la regla también indica que el documento de negocio generado debe ser adicionado al objeto *ProcessSpecification* pasado como parámetro.

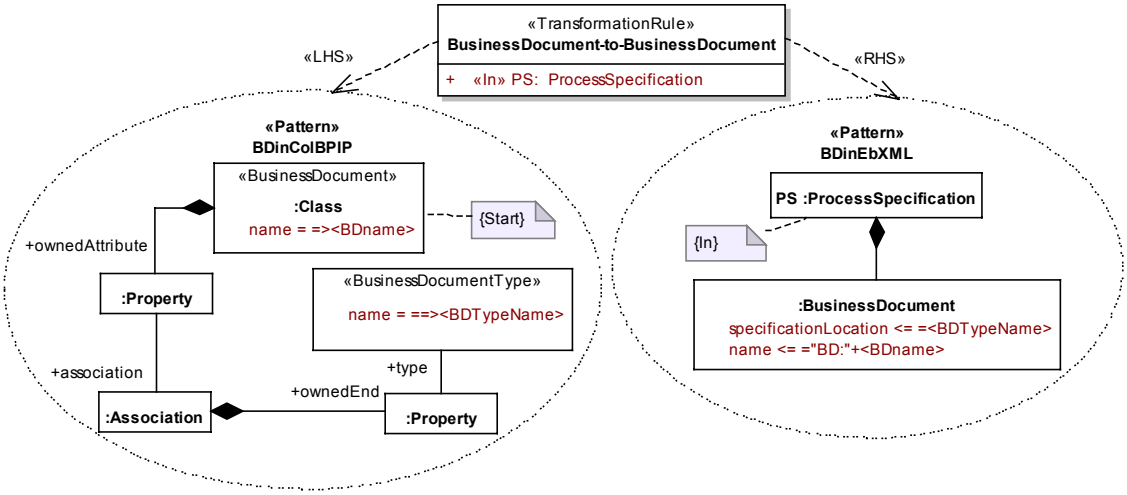


Figura 6-9. Regla de transformación *BusinessDocument-to-BusinessDocument*

En el árbol de reglas (Figura 6-8), esta regla será ejecutada varias veces mientras existan documentos de negocio, como lo indica el estereotipo <<ForEach>> en el nodo *BusinessDocument-to-BusinessDocument*.

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, a continuación se muestra el código que corresponde a algunos de documentos de negocio generados en la especificación BPSS, a partir de los documentos de negocio definidos en el modelo UP-ColBPIP de entrada.

```
<BusinessDocument name = "BlanketPurchaseOrder" specificationLocation =
"http://www.cidx.com/CIDX_CeS_v4.0_Message_OrderCreate.xsd"/>
<BusinessDocument name = "BPOResponse" specificationLocation =
"http://www.cidx.com/CIDX_CeS_v4.0_Message_OrderResponse.xsd"/>
<BusinessDocument name = "BPOChange"
  specificationLocation = "http://www.cidx.com/CIDX_CeS_v4.0_Message_OrderChange.xsd"/>
<BusinessDocument name = "BPOChangeResponse"
  specificationLocation = "http://www.cidx.com/CIDX CeS v4.0 Message OrderResponse.xsd"/>
```

6.2.1.3. Transformación de Protocolos de Interacción

En UP-ColBPIP, un proceso colaborativo es descrito a través de un protocolo de interacción. En BPSS, un proceso colaborativo es descrito en una colaboración binaria. Por lo tanto, un protocolo de interacción es transformado en una colaboración binaria.

En el módulo *Main* (Figura 6-8), primero se obtiene el protocolo de interacción a transformar, a través de la regla *GetInteractionProtocol*. Esta regla obtiene un protocolo, si aún no ha sido generada la colaboración binaria correspondiente. El nodo que contiene esta regla tiene un subnodo que representa la invocación al módulo *InteractionProtocols-to-BinaryCollaborations*. Cada vez que la regla

GetInteractionProtocol retorne verdadero como consecuencia de que ha encontrado un protocolo aún no transformado, se invocará este módulo.

El módulo *InteractionProtocols-to-BinaryCollaborations* (Figura 6-10) tiene como parámetro de entrada un objeto de tipo protocolo de interacción, es decir, el protocolo a ser transformado. El parámetro de salida es la colaboración binaria generada.

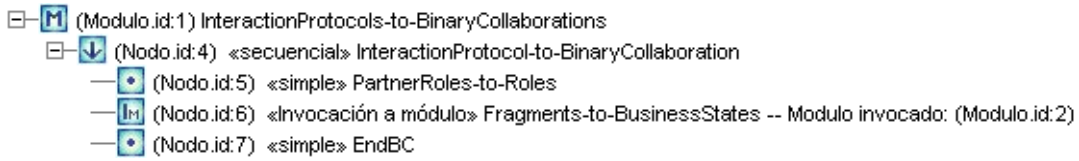


Figura 6-10. Árbol de reglas del módulo *InteractionProtocols-to-BinaryCollaborations*

La regla principal en este módulo es la regla *InteractionProtocol-to-BinaryCollaboration* (Figura 6-11). Esta regla indica que para un protocolo de interacción pasado como parámetro, se crea su correspondiente colaboración binaria, la cual es adicionada a la especificación de procesos también pasada como parámetro. La colaboración binaria es un parámetro de salida de la regla.

Los siguientes pasos en el árbol de reglas del módulo *InteractionProtocols-to-BinaryCollaborations* son: transformar los roles (Sección 6.2.1.4), luego los fragmentos o elementos que componen la coreografía del protocolo (sección 6.2.1.5) y crear los estados finales de la colaboración binaria (regla *EndBC*) como se describe continuación.

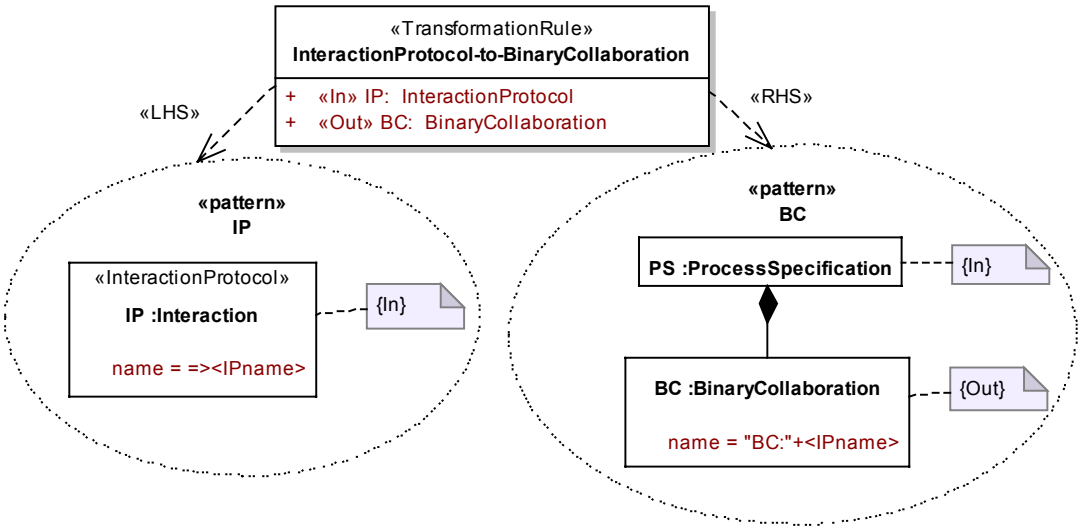


Figura 6-11. Regla de transformación *InteractionProtocol-to-BinaryCollaboration*

Un protocolo de interacción sin eventos *Success* o *Failure* tiene una terminación implícita. No obstante, en BPSS una colaboración binaria siempre debe tener una terminación explícita, ya sea a través de un estado *Success* o un estado *Failure*. Por lo tanto, luego de la transformación de los elementos de un protocolo, es necesario crear estos estados en la colaboración binaria generada a partir del protocolo. La regla *EndBD* es utilizada para tal caso (Figura 6-12). La regla sólo tiene un patrón RHS, el cual indica que se crean dos estados finales, un *Success* y un *Failure*, en la colaboración binaria pasada como parámetro de entrada. Además, para estos estados finales se agregan las asociaciones hacia el último estado de negocio generado (también pasado como parámetro). Estas asociaciones representan la transición desde este estado de negocio a los estados finales.

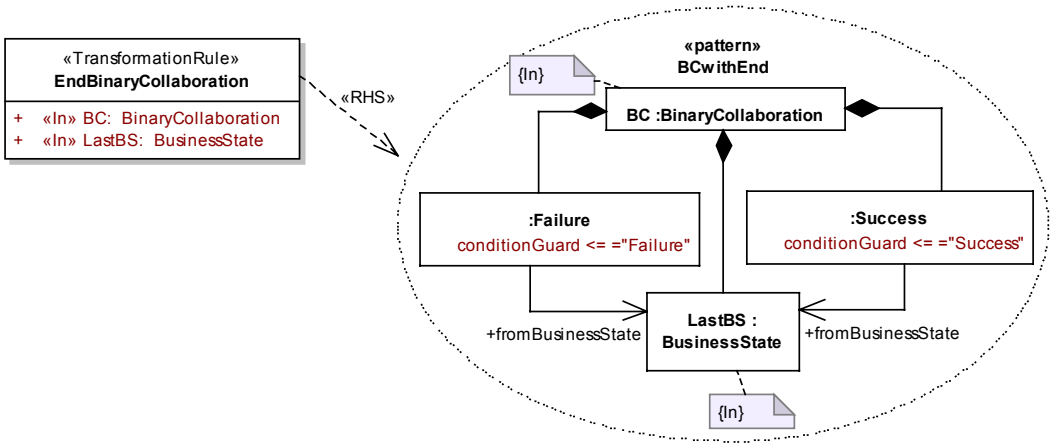


Figura 6-12. Regla de transformación *EndBC*

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, a continuación se muestra el código generado correspondiente al primer protocolo de interacción del modelo UP-ColBPIP de ejemplo, el cual es el protocolo *Forecast-based VMI*. Sólo el código resaltado es generado por las reglas descritas anteriormente. El restante código es generado en la invocación al módulo *Fragments-to-BusinessStates* cuando se transforman los fragmentos del protocolo.

```
<BinaryCollaboration name = "BC:Forecast-based VMI">
   <Role name = "Customer" nameID = "01A"/>
   <Role name = "Supplier" nameID = "01B"/>
   <CollaborationActivity name = "CA:Blanket Purchase Order"
         binaryCollaboration = "BC:Blanket Purchase Order"
         fromRole = "Customer" toRole = "Supplier"/>
   <Start toBusinessState = "CA:Blanket Purchase Order"/>
   <CollaborationActivity name = "CA:Planning and Replenishment Cycles(For)"
         binaryCollaboration = "BC:Planning and Replenishment Cycles(For)"
         fromRole = "Customer" toRole = "Supplier"/>
   <Transition fromBusinessState = "CA:Blanket Purchase Order"
         toBusinessState = "CA:Planning and Replenishment Cycles(For)"/>
   <Success fromBusinessState = "CA:Planning and Replenishment Cycles(For)"
         ConditionGuard = "Success"/>
   <Failure fromBusinessState = "CA:Planning and Replenishment Cycles(For)"
         ConditionGuard = "Failure"/>
</BinaryCollaboration>
```

6.2.1.4. Transformación de Roles

Las colaboraciones binarias en BPSS poseen dos roles. Por lo tanto, en principio sólo es posible transformar protocolos de interacción que posean dos roles. Para transformar protocolos con más de dos roles es necesario primero sintetizar a los mismos en protocolos de a dos roles, previo a la transformación a BPSS. De otra manera esta transformación resulta muy complicada. En este modelo de transformación se asume que los protocolos sólo poseen dos roles.

La regla *PartnerRole-to-Role* (Figura 6-13) indica que por cada rol de un protocolo de interacción, un rol es generado en la correspondiente colaboración binaria. Los roles de un protocolo son obtenidos a partir de las *lifelines* del protocolo, las cuales hacen referencia a los roles de los socios.

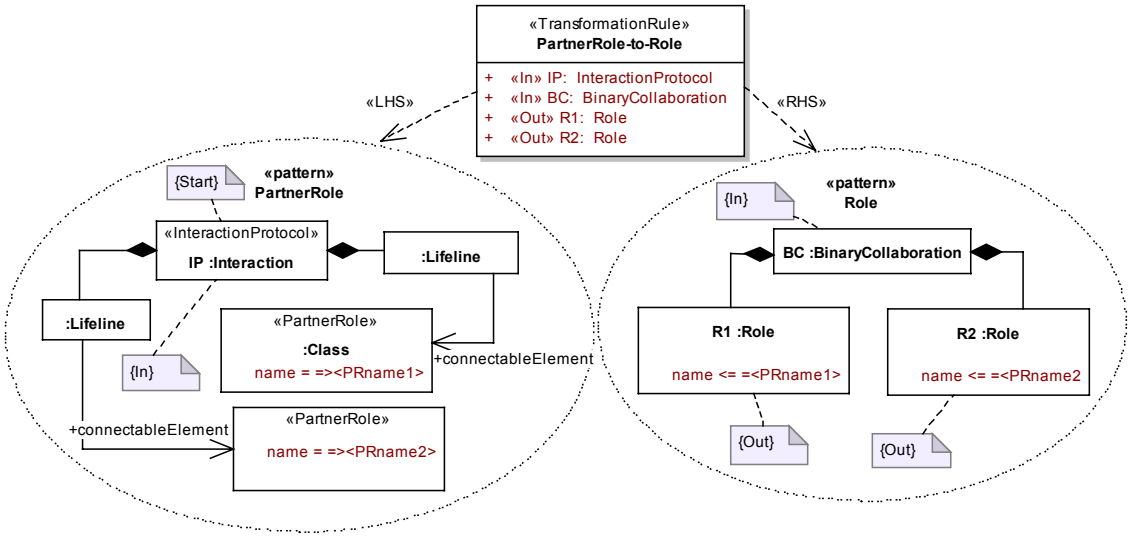


Figura 6-13. Regla de Transformación *PartnerRole-to-Role*

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, a continuación se muestra resaltado el código correspondiente a los roles de la colaboración binaria que representa al protocolo *Forecast-based VMI*.

```
<BinaryCollaboration name = "BC:Forecast-based VMI">
   <Role name = "Customer" nameID = "01A"/>
   <Role name = "Supplier" nameID = "01B"/>
   <CollaborationActivity name = "CA:Blanket Purchase Order"
         binaryCollaboration = "BC:Blanket Purchase Order"
         fromRole = "Customer" toRole = "Supplier"/>
   <Start toBusinessState = "CA:Blanket Purchase Order"/>
   <CollaborationActivity name = "CA:Planning and Replenishment Cycles(For)"
         binaryCollaboration = "BC:Planning and Replenishment Cycles(For)"
         fromRole = "Customer" toRole = "Supplier"/>
   <Transition fromBusinessState = "CA:Blanket Purchase Order"
         toBusinessState = "CA:Planning and Replenishment Cycles(For)"/>
   <Success fromBusinessState = "CA:Planning and Replenishment Cycles(For)"
         ConditionGuard = "Success"/>
   <Failure fromBusinessState = "CA:Planning and Replenishment Cycles(For)"
         ConditionGuard = "Failure"/>
</BinaryCollaboration>
```

6.2.1.5. Transformación de Fragmentos

La transformación de los elementos o fragmentos que componen la coreografía de un protocolo es realizada en el módulo *Fragments-to-BusinessStates* (Figura 6-14). Un fragmento es transformado, de acuerdo a su tipo, en uno de los tipos de estados de negocio de BPSS. El módulo tiene como parámetros de entrada un objeto de tipo *InteractionFragment* y un objeto de tipo *BinaryCollaboration*. El primero es utilizado para aceptar como entrada un protocolo de interacción, cuando el módulo es invocado desde el módulo *InteractionProtocols-to-BinaryCollaborations*. Además, también es utilizado para aceptar como entrada caminos de interacción que componen un segmento de flujo de control en un protocolo, cuando este módulo es invocado en forma recursiva en la transformación de segmentos de control. Además, el módulo tiene como parámetro de salida un objeto de tipo *BusinessState*, a través del cual se retorna el último estado de negocio generado.

La primer regla a ejecutar en el módulo es la regla *Fragment-to-BusinessState* (Figura 6-15), la cual es utilizada para obtener y determinar la existencia de un fragmento en el protocolo o camino de interacción pasado como parámetro de entrada en el módulo. El patrón LHS indica que para el objeto *IF* de entrada se debe encontrar un fragmento. El objeto *NewIF* representa el fragmento encontrado el cual es un parámetro de salida de la regla, dado que el fragmento será transformado por otras reglas de acuerdo a su tipo específico. El objeto *NewIF* también es de tipo

InteractionFragment, lo cual permite encontrar fragmentos independientemente de su subtipo específico. Además, la regla no posee un patrón RHS y por lo tanto no genera objetos en el modelo de salida.

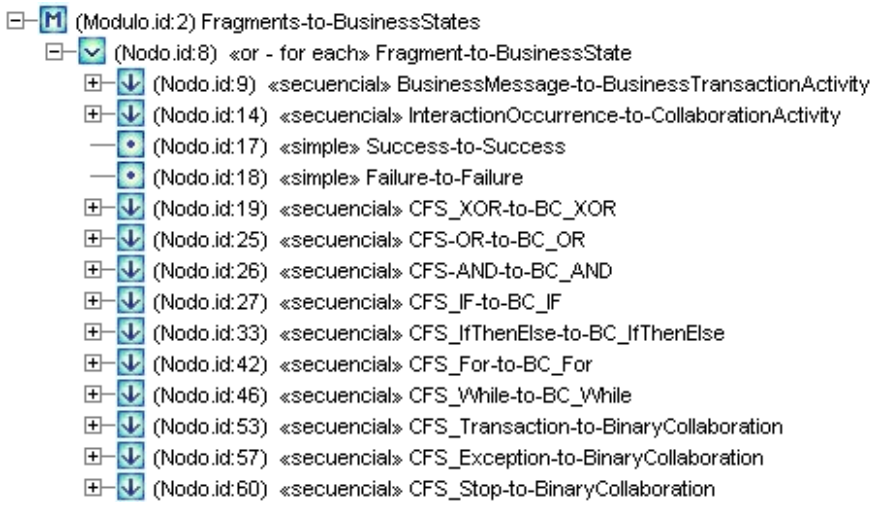


Figura 6-14. Árbol de reglas del módulo *Fragments-to-BusinessStates*

En el módulo, esta regla será evaluada mientras exista un fragmento en el protocolo o camino de interacción de entrada. Cada vez que se ejecute la regla y encuentre un fragmento, las reglas de los subnodos serán evaluadas para transformar dicho fragmento. Sólo una de ellas será evaluada con éxito, de acuerdo al tipo de fragmento encontrado. Por lo tanto, el nodo *Fragment-to-BusinessState* es de tipo *OR*. Las siguientes reglas definen la transformación de cada uno de los tipos de fragmentos de un protocolo de interacción.

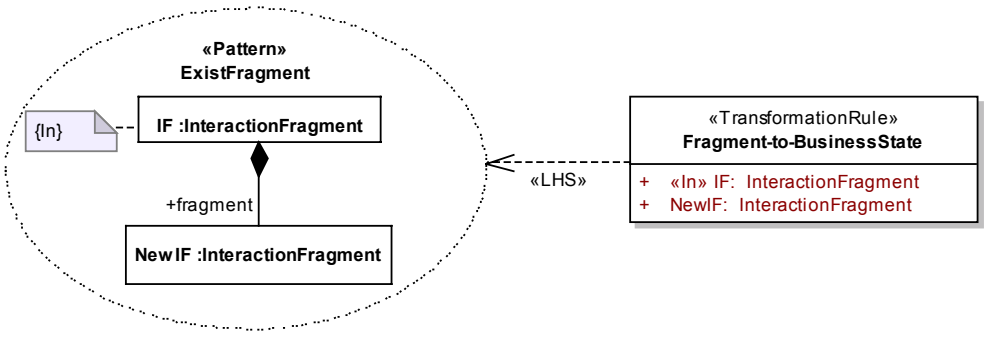


Figura 6-15. Regla de Transformación *Fragment-to-BusinessState*

6.2.1.6. Transformación de Mensajes

Dentro de la coreografía de un protocolo de interacción, una interacción entre dos partes es realizada a través de un mensaje de negocio, basado en un acto de

comunicación, el cual constituye el concepto atómico de un proceso en UP-ColBPIP. Dentro de la coreografía de una colaboración binaria, una interacción entre dos partes es realizada a través de una actividad de transacción de negocio, la cual a su vez representa la ejecución de una transacción de negocio. Esta última constituye el concepto atómico de un proceso en BPSS. Por lo tanto, un mensaje de negocio es transformado en una actividad de transacción de negocio con su correspondiente transacción de negocio que representa el acto de comunicación y otras propiedades del mensaje. La transformación de un mensaje de negocio es realizada a través del conjunto de reglas mostrada en la Figura 6-16.

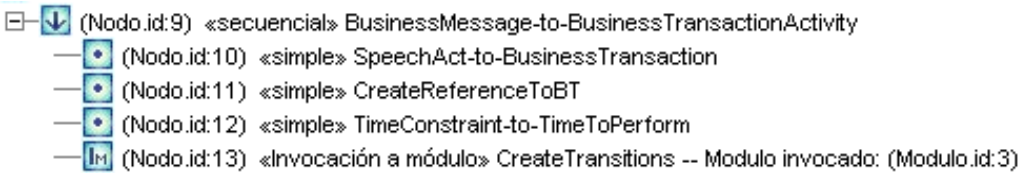


Figura 6-16. Subárbol de reglas para la transformación de mensajes de negocio

La regla *BusinessMessage-to-BusinessTransactionActivity* (Figura 6-17) indica que un mensaje de negocio es transformado en una actividad de transacción de negocio, ya que ambos conceptos representan una interacción entre dos roles. El patrón LHS de la regla indica lo siguiente: el objeto *NewIF* pasado como parámetro debe ser de tipo *EventOccurrence* y su atributo de asociación *sendMessage* debe hacer referencia a un mensaje de negocio; y además debe existir otro fragmento del protocolo también de tipo *EventOccurrence* y su atributo de asociación *receiveMessage* debe hacer referencia al mismo mensaje de negocio. El mensaje de negocio es obtenido a través de las ocurrencias de eventos. Esto se debe a que el orden de los mensajes en la coreografía de un protocolo de interacción es deducido a partir de pares de ocurrencias de eventos, una representando el envío del mensaje sobre una *lifeline* y otra representando la recepción del mismo sobre otra *lifeline*. Por lo tanto, el rol remitente y el rol receptor del mensaje son obtenidos desde estas *lifelines*.

En el patrón RHS, la restricción OCL es utilizada para establecer la correspondencia entre los roles involucrados en el mensaje de negocio, obtenidos en el patrón LHS, y los roles de la colaboración binaria que cumplirán la función de receptor y remitente, los cuales son parámetros de entrada en la regla. El nombre de la actividad de transacción de negocio es la concatenación del nombre del acto de comunicación del mensaje, seguido del nombre del documento de negocio.

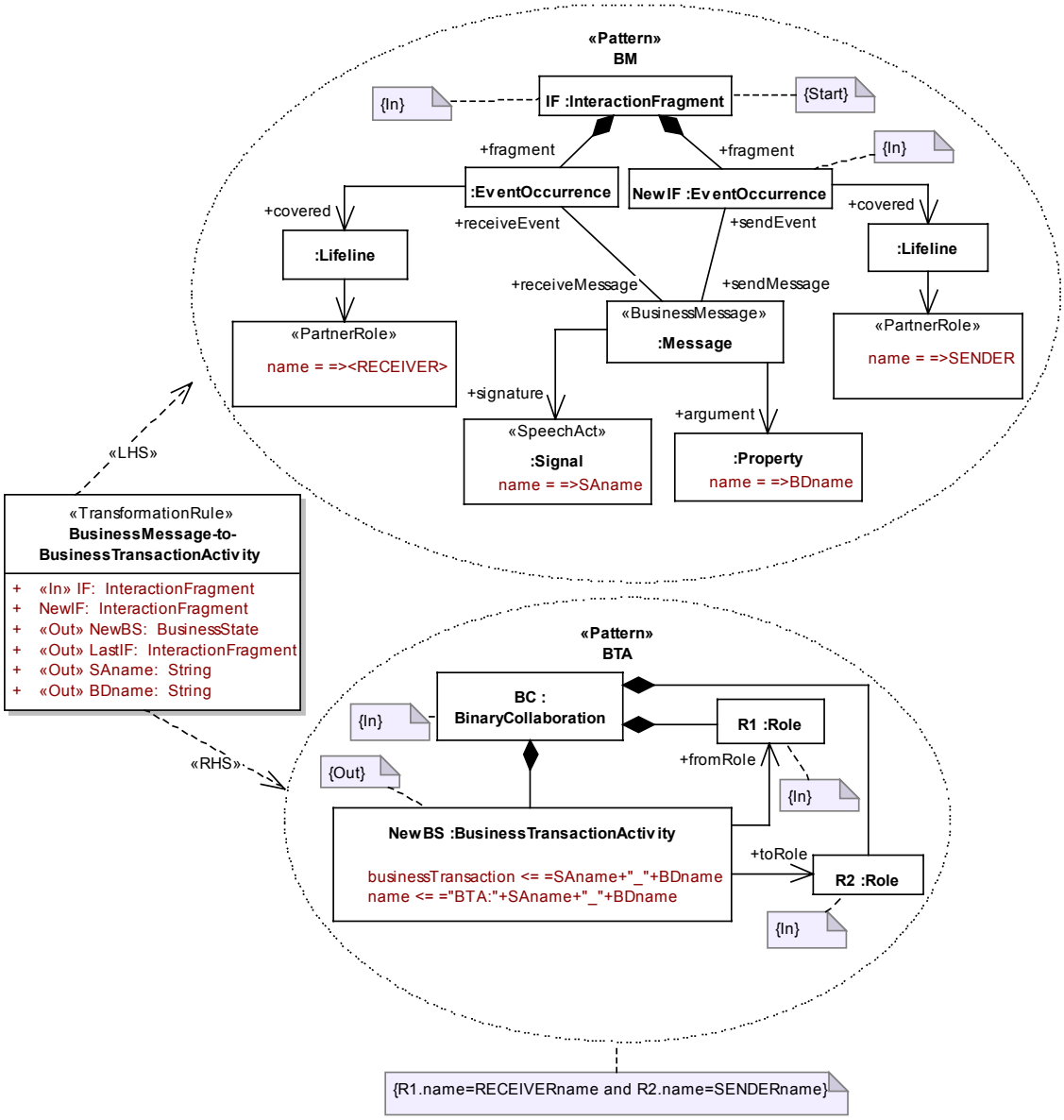


Figura 6-17. Regla de transformación *BusinessMessage-to-BusinessTransactionActivity*

El concepto de actividad de transacción de negocio permite representar la dirección del mensaje, es decir, el rol remitente y el rol receptor. No obstante, otras propiedades de un mensaje de negocio son representadas en la transacción de negocio que es ejecutada en la actividad de transacción de negocio.

La regla *SpeechAct-to-BusinessTransaction* permite generar las siguientes propiedades de un mensaje: la característica asincrónica, el acto de comunicación, el documento de negocio y los acuses de recibo. Por un lado, para representar la característica asincrónica de un mensaje, sólo la actividad de solicitud es generada en una transacción de negocio. La actividad de respuesta no es generada. Por otro lado, la

actividad de solicitud representa la intención del acto de comunicación. En consecuencia, el nombre de la misma es el nombre del acto de comunicación. Esta actividad además es definida con un objeto *DocumentEnvelope* que referencia al documento de negocio a ser enviado, el cual se corresponde con el documento enviado en el mensaje.

La regla *SpeechAct-to-BusinessTransaction* es mostrada en la Figura 6-18. Esta regla posee dos patrones LHS, uno que opera sobre el modelo de entrada y otro que opera sobre el modelo de salida. El primero es utilizado para obtener los atributos *timeToReceiptAcknowledgment* y *timeToReadAcknowledgment* del mensaje de negocio. Estos atributos se corresponden con los atributos *timeToAcknowledgmentReceipt* y *timeToAcknowledgmentAcceptance* de la transacción de negocio generada, como se indica en el patrón RHS. El segundo patrón LHS es utilizado para determinar si la transacción de negocio aún no ha sido generada. Esto es indicado con la restricción *Neg* sobre el objeto de tipo *BusinessTransaction*. Además, el patrón es utilizado para obtener el documento de negocio ya generado que se corresponde con el documento de negocio del mensaje. Si la ejecución de este patrón retorna verdadero, es decir si no existe tal transacción de negocio, se ejecuta el patrón RHS. En este patrón, se crea la transacción de negocio, se la adiciona a la especificación de procesos, se crea la actividad de solicitud con el nombre del acto de comunicación, se crea el objeto de tipo *DocumentEnvelope* y se adiciona una asociación del mismo al objeto *BD*, el cual es el mismo objeto encontrado en el patrón LHS.

Luego de la ejecución de la regla *SpeechAct-to-BusinessTransaction*, se ejecuta la regla *CreateReferenceToBT*. Dicha regla permite crear la referencia a la transacción de negocio, dentro de la actividad de transacción de negocio que representa al mensaje.

Luego de la ejecución de las reglas anteriores, la siguiente regla a ser evaluada es la regla *TimeConstraint-to-TimeToPerform*. Esta determina si existe una restricción de tiempo asociada al mensaje de negocio. Si esto ocurre se asigna valor al atributo *timeToPerform* de la actividad de transacción de negocio que representa al mensaje.

Finalmente, se invoca al módulo *CreateTransitions* (Sección 6.2.1.17), a través del cual se crea la transición entre la actividad de transacción de negocio generada y el estado de negocio previo que a la misma.

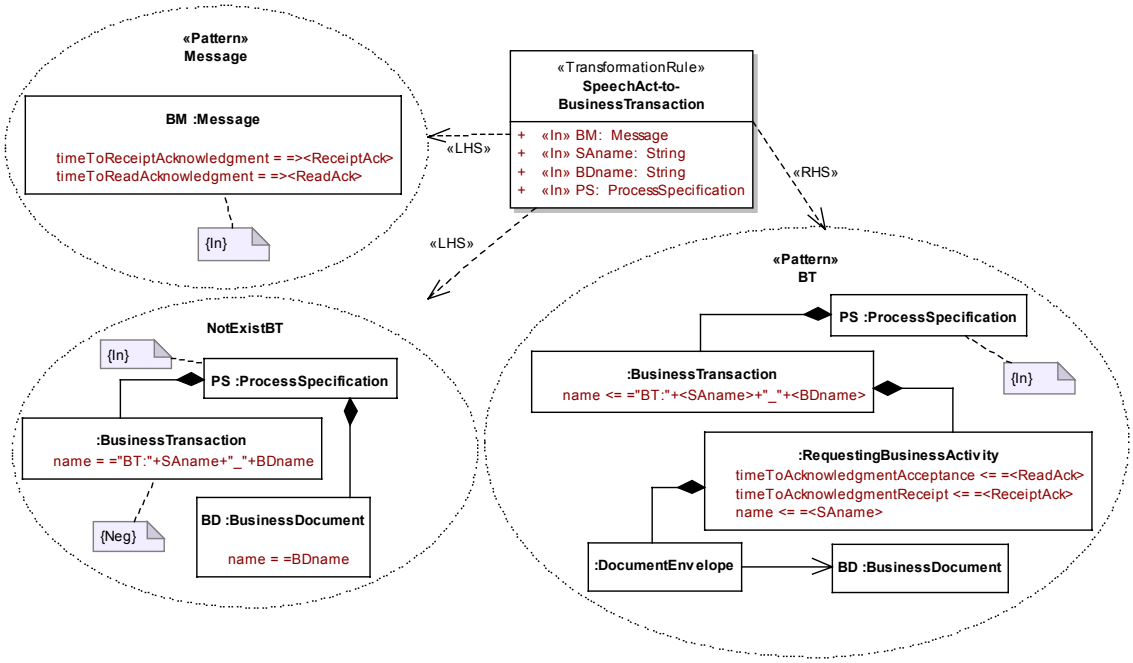


Figura 6-18. Regla de transformación *SpeechAct-to-BusinessTransaction*

No todos los atributos de una actividad de transacción de negocio y de una transacción de negocio pueden ser derivados a partir de un modelo UP-ColBPIP. Esto se debe a que los mismos constituyen conceptos de más bajo nivel de abstracción. Por ejemplo, el atributo *isGuaranteedDeliveredRequired* se refiere a si el servicio de mensajes utilizado para ejecutar la transacción debe garantizar la entrega de mensajes. Además, las propiedades de seguridad de los documentos de negocio enviados en cada transacción, las cuales son definidas a través de los conceptos *DocumentSecurity* y *Attachment* tampoco pueden ser derivadas desde UP-ColBPIP. Las mismas deben ser configuradas luego, cuando se definen los detalles de bajo nivel necesarios para la ejecución de los procesos.

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, a continuación se muestra el código resultante en la transformación del mensaje *propose(BlanketPurchaseOrder)*, del protocolo de interacción *Blanket Purchase Order*. El código resaltado pertenece al generado por las reglas definidas anteriormente.

```
<ProcessSpecification name = "VMI-based Collaborative Replenishment" … />
  <!--Business Transactions -->
  <BusinessTransaction name = "propose_BlanketPurchaseOrder">
     <RequestingBusinessActivity name = "propose"
           timeToAcknowledgeReceipt = "PT1H" timeToAcknowledgeAcceptance = "PT1H">
           isInteligentCheckRequired = "true"
         <DocumentEnvelope businessDocument = "BlanketPurchaseOrder"/>
     </RequestingBusinessActivity>
  </BusinessTransaction>
  ....
  <BinaryCollaboration name = "BC:Blanket Purchase Order"
           timeToPerform = "P6D">
     <Role name = "Customer" nameID = "01A"/>
     <Role name = "Supplier" nameID = "01B"/>
     <BusinessTransactionActivity name = "BTA:propose_BlanketPurchaseOrder"
           fromRole = "Customer" toRole = "Supplier"
           businessTransaction = "propose_BlanketPurchaseOrder"/>
     ....
  </BinaryCollaboration>
```

6.2.1.7. Transformación de Referencias a Protocolos Anidados

Una referencia a un protocolo es transformada en una actividad de colaboración, la cual representa una referencia a otra colaboración binaria. Para ello, el protocolo que es referenciado (el protocolo anidado) también debe ser transformado en la colaboración binaria correspondiente (la colaboración binaria anidada) a ser referenciada por la actividad de colaboración. El conjunto de reglas utilizado para llevar a cabo esto es mostrado en la Figura 6-19.

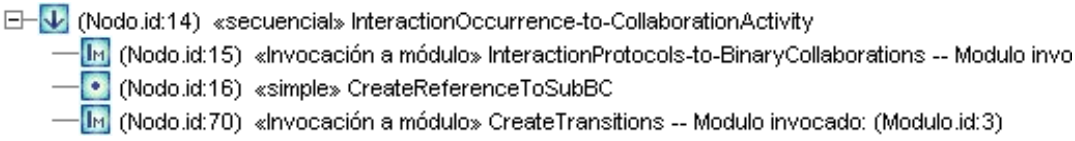


Figura 6-19. Subárbol de reglas para la transformación de Protocolos Anidados

La regla *InteractionOccurrence-to-CollaborationActivity* (Figura 6-20) indica que una referencia a un protocolo es convertida en una actividad de colaboración, es decir, una referencia a una colaboración binaria. El patrón LHS de la regla indica que el objeto *IF* pasado como parámetro debe ser de tipo *InteractionOccurrence* y el mismo debe tener una referencia a un protocolo de interacción, el cual es un parámetro de salida de la regla. El mismo será transformado en las siguientes reglas. El patrón RHS indica que una actividad de colaboración debe ser generada con el mismo nombre de la referencia a protocolo encontrada.

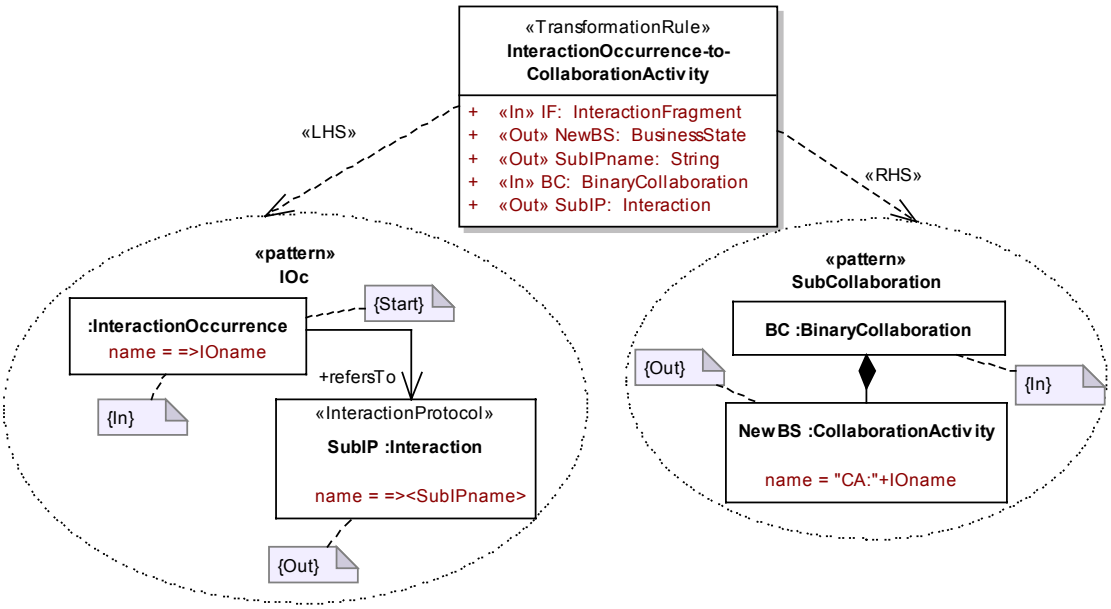


Figura 6-20. Regla de transformación *InteractionOccurrence-to-CollaborationActivity*

El siguiente paso es la transformación del protocolo de interacción anidado en la correspondiente colaboración binaria anidada, si esta aún no ha sido generada. Para ello se invoca al módulo *InteractionProtocols-to-BinaryCollaborations*. Esto implica que se ejecutan nuevamente todo los pasos requeridos para transformar cada uno de los elementos que componen el protocolo anidado, en el caso que la colaboración binaria correspondiente aún no fue generada. En el caso contrario, la evaluación de la primera regla de este módulo retornará falso, lo cual significa que el protocolo anidado ya fue transformado.

Una vez finalizada la ejecución del módulo anterior, el siguiente paso es la creación de la referencia, dentro de la actividad de colaboración generada, a la colaboración binaria anidada. Esto se realiza a través de la regla *CreateReferenceToSubBC*.

Finalmente, se invoca al módulo *CreateTransitions* (Sección 6.2.1.17), a través del cual se crea la transición entre la actividad de colaboración generada y el estado de negocio previo que halla sido generado.

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, se muestra el código resultante de la transformación de la referencia al protocolo *Blanket Purchase Order* definida en el protocolo *Forecast-Based VMI*. El código correspondiente a la colaboración binaria anidada que es generada a partir del

protocolo *Blanket Purchase Order* no es mostrado aquí. Sólo se muestra el código generado en la colaboración binaria que referencia a la colaboración binaria anidada.

```
<BinaryCollaboration name = "BC:Forecast-based VMI">
    ...
    <CollaborationActivity name = "CA:Blanket Purchase Order"
          binaryCollaboration = "BC:Blanket Purchase Order"
          fromRole = "Customer" toRole = "Supplier"/>
    ...
</BinaryCollaboration>
```

6.2.1.8. Transformación de los Eventos de Terminación

Los eventos que representan la terminación de la ejecución de un protocolo de interacción, son transformados en los correspondientes estados de finalización provistos en BPSS. La regla *Success-to-Success* (Figura 6-21) indica que un evento *Success* representando la terminación exitosa de un protocolo es transformado en un estado de negocio *Success* en la correspondiente colaboración binaria, el cual indica una terminación exitosa de esta última. Al estado *Success* generado en el patrón de salida se le asocia el estado previo, el cual es pasado como parámetro a la regla.

La regla *Failure-to-Failure* indica que un evento *Failure* representando la terminación fallida de un protocolo es transformado en un estado de negocio *Failure* en la correspondiente colaboración binaria, el cual indica una terminación fallida de esta última. La semántica operacional de esta regla es similar a la regla *Success-to-Success*, excepto que para un objeto de tipo *Failure* en el patrón LHS, se genera un estado *Failure* en la colaboración binaria.

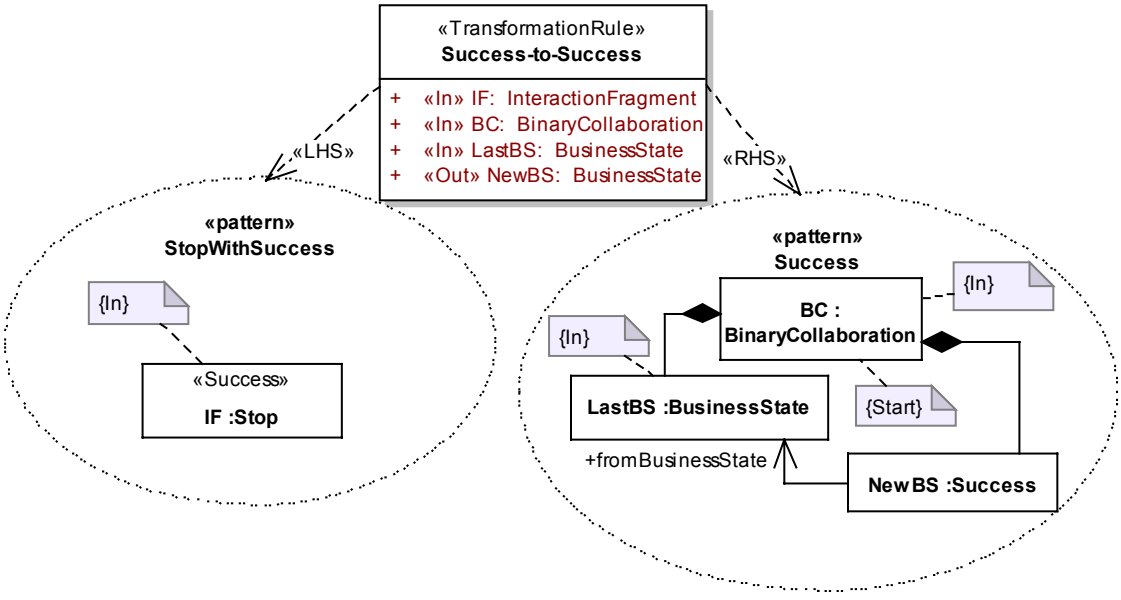


Figura 6-21. Regla de transformación *Success-to-Success*

6.2.1.9. Transformación de Segmentos de Flujo de Control

En UP-ColBPIP, la coreografía de los protocolos de interacción no sólo es definida a través de la secuencia ordenada de mensajes, sino fundamentalmente a través de los segmentos de flujo de control que contienen los caminos de interacción (*InteractionOperands*), los cuales a su vez pueden estar compuestos de otros fragmentos ordenados. Por el contrario, en BPSS, la coreografía de las colaboraciones binarias es definida a través de los estados de negocio y las transiciones entre los mismos, de manera similar a máquinas de estado. Por lo tanto, es necesario transformar la coreografía de mensajes de los protocolos de interacción en una coreografía de estados de negocio de las colaboraciones binarias.

Para llevar a cabo esto, todo segmento de flujo de control es transformado en una colaboración binaria. Existen varias razones para esto:

- BPSS no soporta la definición explícita de flujos de control complejos, tales como bucles.
- Los operadores utilizados en los segmentos de flujo de control no tienen una contraparte directa en BPSS.
- Los segmentos de flujo de control pueden contener segmentos de flujo de control anidados. Por lo tanto, la transformación de ellos a colaboraciones binarias reduce la complejidad de la colaboración binaria que está representando a un protocolo de interacción.

La transformación de un segmento de flujo de control a una colaboración binaria es realizada de acuerdo a su operador. Por lo tanto, para cada tipo de operador de control se deben definir las reglas de transformación específicas. A nivel conceptual, en la tabla 6-1 se resume cada una de las posibles transformaciones.

Por ejemplo, un segmento de flujo de control con un operador *Xor* es convertido en una colaboración binaria, la cual posee un estado *Fork* con un valor “XOR” en el atributo *type*, y además posee un estado *Join* con un valor “False” en su atributo *waitForAll*. El estado *Fork* debe ser el primer estado de la colaboración binaria luego del estado *Start*. El estado *Join* debe ser el estado previo al estado final *Success*. Como otro ejemplo, un segmento de flujo de control con un operador *Loop* es convertido en una colaboración binaria con un estado *Decision*, el cual debe ser el primer estado de

dicha colaboración, luego del estado de comienzo, en el caso que la condición del bucle sea "(0,n)". De lo contrario, el estado *Decision* debe ser definido al final, antes de los estados *Success* y *Failure* de la colaboración binaria. Segmentos con los operadores *Transaction*, *Exception*, y *Stop* son transformados sin la necesidad de generar estados *Fork*, *Join* o *Decisión*. La excepción a la regla de transformar segmentos en colaboraciones binarias, son aquellos definidos con el operador *Cancel*. En este caso no es posible establecer una correspondencia entre un segmento y una colaboración binaria, como se discute luego.

Segmento de flujo de control	**Colaboración binaria con los siguientes estados de negocio**		
operator	***Fork.type***	***Join.waitForAll***	***Decision***
Xor	XOR	False	--
Or	OR	False	--
And	OR	True	--
If	--	--	Este estado es definido después del estado *Start*
Loop (0,n)	--	--	Este estado es definido después del estado *Start*
Loop (1,n)	--	--	Este estado es definido antes de los estados de terminación
Exception	--	--	--
Transaction	--	--	--
Stop	--	--	--
Cancel	x	x	x

Tabla 6-1. Transformación de un segmento de flujo de control en una colaboración binaria

Por otra parte, un segmento de flujo de control posee caminos de interacción, los cuales a su vez poseen un conjunto ordenado de fragmentos (elementos), al igual que un protocolo de interacción. Los caminos de interacción no tienen una contraparte en BPSS. Por lo tanto, para representar los posibles caminos de un segmento como así también el orden de los elementos de un camino, es necesario derivar las transiciones correspondientes entre los estados de negocio que representan a dichos elementos.

Las diferentes reglas de transformación para cada tipo de segmento de control son descritas en el Anexo B. A continuación, como ejemplo, se describen las reglas involucradas en el proceso de transformación de un segmento de flujo de control con el operador *Xor* (Figura 6-22).

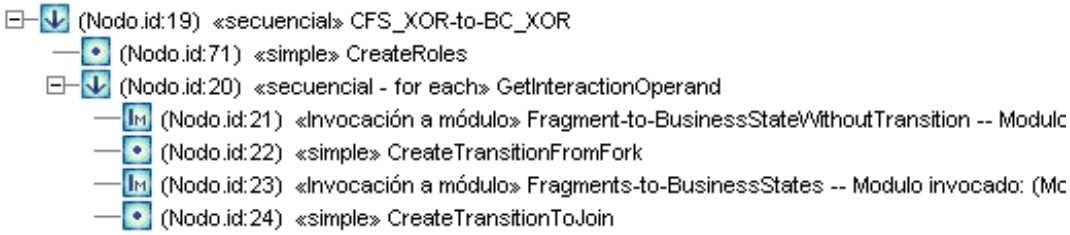


Figura 6-22. Subárbol de reglas para la transformación de segmento *Xor*

En primer lugar, la regla *CFS_XOR-to-BC_XOR* es la encargada de crear la colaboración binaria correspondiente. La Figura 6-23 muestra la semántica operacional de la regla. El patrón LHS indica que el objeto *IF* (pasado como parámetro) debe ser un segmento de flujo de control con el operador *Xor*. El patrón RHS indica que para representar este tipo de segmento se debe generar: una actividad de colaboración, dentro de la colaboración binaria (pasada como parámetro), la cual representa a un protocolo u a otro segmento (cuando existe anidamiento de segmentos); y la nueva colaboración binaria que representa al segmento. Además, en esta nueva colaboración que representa al segmento, se genera el estado de comienzo de la misma, el estado *Fork* con el atributo *type* establecido en "XOR", y el estado *Join* con el atributo *waitForAll* en "False". Esto permite representar la misma semántica de un segmento con el operador *Xor*. También, se agrega un estado *Success* cuya transición indica que luego del estado *Join* se finaliza la colaboración binaria, para representar el fin del segmento.

Una vez ejecutada la regla anterior, en la colaboración binaria generada que corresponde al segmento se crean los roles (Regla *CreateRoles*), de acuerdo a los roles del protocolo. Luego se transforman los elementos que componen los caminos de interacción del segmento. Como se indicó anteriormente, el concepto de camino de interacción (*InteractionOperand*) en UP-ColBPIP no tiene una contraparte directa en BPSS. Estos caminos deben ser expresados a través de las transiciones entre los estados correspondientes. En el caso de este tipo de segmento, cada camino encontrado (regla *GetInteractionOperand*) es expresado de la siguiente forma:

1. Se transforma el primer fragmento del camino y se genera un estado de negocio correspondiente. Esto se realiza invocando el módulo *Fragment-to-BusinessStateWithoutTransition*.
2. Se crea una transición, en donde el estado *Fork* es el origen y el estado de negocio generado anteriormente es el destino (Regla *CreateTransitionFromFork*).
3. Se transforman los restantes elementos del camino. Esto se realiza invocando en forma recursiva al módulo *Fragments-to-BusinessStates*. En este caso se pasan como parámetros de entrada al módulo el camino de interacción y la colaboración binaria que representa el segmento. El módulo retorna como parámetro de salida el último estado de negocio generado a partir del último fragmento del camino.
4. Se crea la transición entre este último estado de negocio generado y el estado *Join*, donde éste último es el destino de la transición (Regla *CreateTransitionToJoin*). Esto representa la terminación de dicho camino.

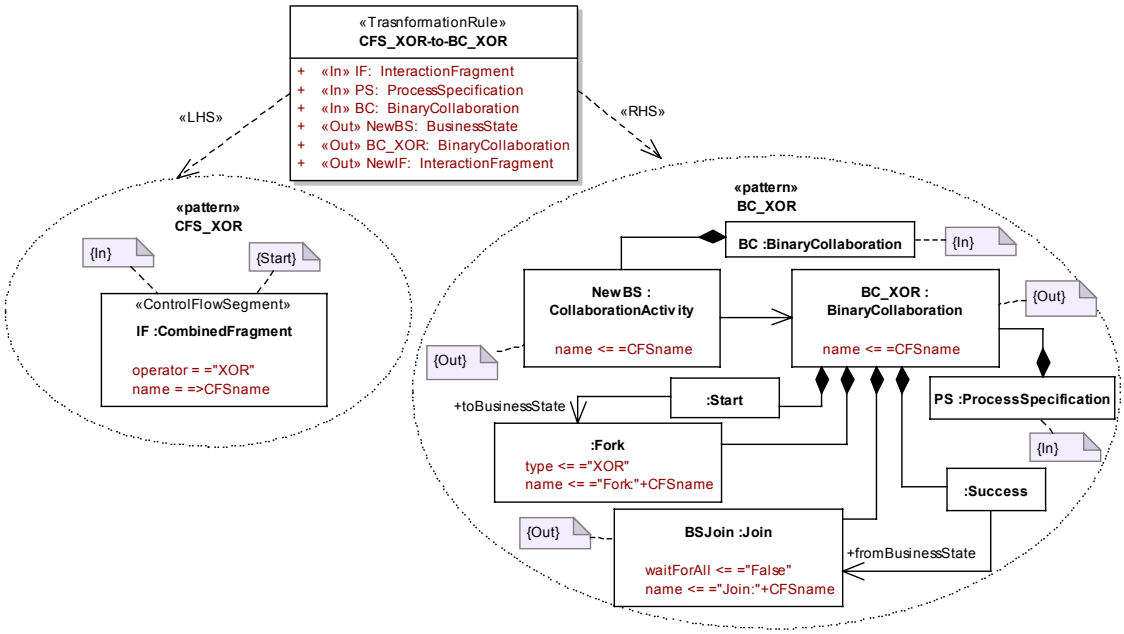


Figura 6-23. Regla de transformación *CFS_XOR-to-BC_XOR*

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

A continuación se muestra el código generado en la transformación del segmento *BPO Changes*, del protocolo de interacción *Blanket Purchase Order*. Sólo el código resaltado es generado por las reglas descritas anteriormente. El restante código es generado en las invocaciones al módulo *Fragment-to-BusinessStateWithoutTransition* y en las invocaciones recursivas al módulo *Fragments-to-BusinessStates*.

```
<BinaryCollaboration name = "BC:BPO Changes(XOR)" initiatingRoleID = "01B">
      <Role name = "Customer" nameID = "01A"/>
      <Role name = "Supplier" nameID = "01B"/>
      <Start toBusinessState = "Fork:BPO Changes(XOR)"/>
      <Fork name = "Fork:BPO Changes(XOR)"/>
      <!--This represents the First interaction operand of the Segment BPO Changes-->
      <BusinessTransactionActivity name = "BTA:propose_BPOChange"
              businessTransaction = "propose_BPOChange"
              fromRole = "Supplier" toRole = "Customer"/>
      <Transition fromBusinessState = "Fork:BPO Changes(XOR)"
              toBusinessState = "BTA:propose_BPOChange">
          <ConditionExpression expressionLanguage = "NL"
              conditionExpression = "ChangeRequired"/>
      </Transition>
      <CollaborationActivity name = "CA:BPO Change Responses(XOR)"
              fromRole = "Customer" toRole = "Supplier"/>
      <Transition fromBusinessState = "BTA:propose_BPOChange"
              toBusinessState = "CA:BPO Change Responses(XOR)"/>
      <Transition fromBusinessState = "CA:BPO Change Responses(XOR)"
              toBusinessState = "Join:BPO Changes(XOR)"/>
      <!--This represents the Second interaction operand of the Segment BPO Changes-->
      <BusinessTransactionActivity name = "BTA:accept-proposal_BPOResponse"
              businessTransaction = "accept-proposal_BPOResponse"
              fromRole = "Supplier" toRole = "Customer"/>
      <Transition fromBusinessState = "Fork:BPO Changes(XOR)"
              toBusinessState = "BTA:accept-proposal_BPOResponse">
          <ConditionExpression expressionLanguage = "NL"
              conditionExpression = "ChangeIsNotRequired"/>
      </Transition>
      <CollaborationActivity name = "CA:BPO Changes By Customer(If)"
              binaryCollaboration = "BC:BPO Changes By Customer(If)"
              fromRole = "Customer" toRole = "Supplier"/>
      <Transition fromBusinessState = "BTA:accept-proposal_BPOResponse"
              toBusinessState = "CA:BPO Changes By Customer(If)"/>
      <Transition fromBusinessState = "CA:BPO Changes By Customer(If)"
              toBusinessState = "Join:BPO Changes(XOR)"/>
      <Join name = "Join:BPO Changes(XOR)" waitForAll = "False"/>
      <Success fromBusinessState = "Join:BPO Changes(XOR)"/>
   </BinaryCollaboration>
```

6.2.1.17. Creación de Transiciones en BPSS

En UP-ColBPIP, el orden de los elementos en la coreografía de los protocolos está dado por el conjunto ordenado de fragmentos de interacción que lo componen. En cambio, en BPSS, el orden de los elementos está dado por las transiciones entre los estados que componen una colaboración binaria. El concepto de transición de estados en BPSS no tiene una contraparte directa en UP-ColBPIP.

Por lo tanto, para representar el orden de los elementos de un protocolo, en BPSS es necesario generar las transiciones correspondientes entre los estados de negocio generados. El módulo *CreateTransitions* consiste de un conjunto de reglas que realizan estas transformaciones (Figura 6-24). La regla *CreateTransition* no produce ningún resultado en el modelo de salida. La misma sólo es agregada para representar un nodo de tipo *Or*, dado que sólo una de las reglas evaluadas en los subnodos retornará verdadero.

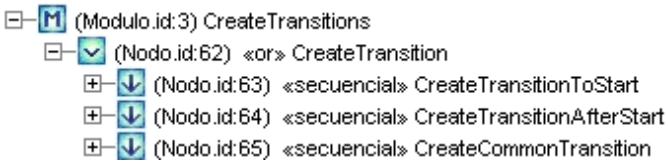


Figura 6-24. Módulo *CreateTransitions*

La regla *CreateTransitionToStart* (Figura 6-25) es utilizada para crear una transición entre el estado de comienzo de una colaboración binaria y el primer estado de negocio generado en la misma. Por lo tanto, el patrón LHS de la regla indica que la colaboración binaria no debe poseer un estado de comienzo. En el patrón RHS, el estado de comienzo es creado, como así también la asociación del mismo al estado de negocio, la cual indica la transición entre dichos estados.

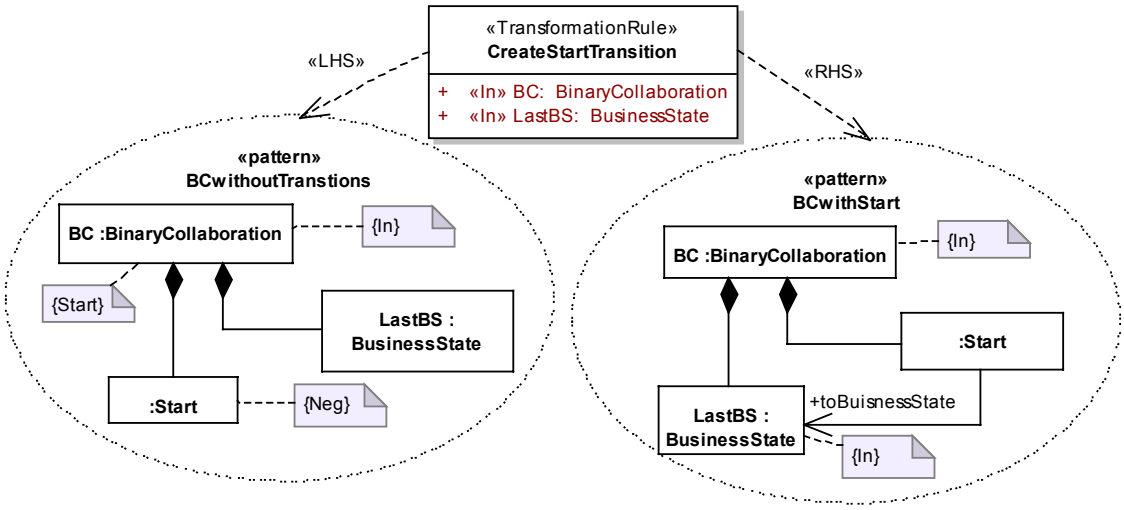


Figura 6-25. Regla de Transformación *CreateTransitionToStart*

La regla *CreateTransitionAfterStart* (Figura 6-26) es utilizada para crear una transición entre dos estados de negocio, cuando aún no se ha creado un objeto *Transition* en la colaboración. Esto significa que sólo debe existir un estado *Start* con

una asociación a un primer estado de negocio representando la transición entre dichos estados. Esta condición es representada en la restricción OCL asociada al patrón LHS. La semántica de dicho patrón es la siguiente. Se debe encontrar un estado de negocio (*PreviousBS*) en la colaboración binaria pasada como parámetro, la cual debe contener el último estado de negocio generado. Además, se debe satisfacer la expresión OCL, para establecer que un objeto *PreviousBS* encontrado debe ser el estado de negocio anterior al último estado generado. La expresión indica que, en la colaboración binaria, el estado de comienzo debe tener una transición al objeto *PreviousBS* encontrado, y no debe existir una transición que tenga como entrada al objeto *PreviousBS* encontrado. El patrón RHS crea el objeto *Transition* y establece el estado origen y destino de la transición.

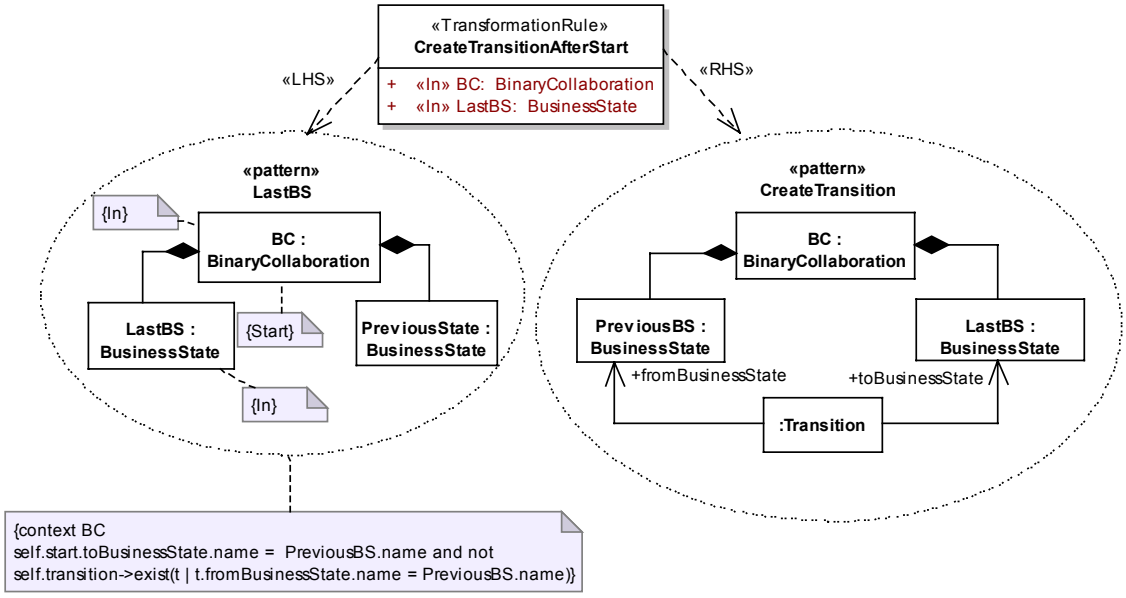


Figura 6-26. Regla de Transformación *CreateTransitionAfterStart*

La regla *CreateCommonTransition* (Figura 6-27) es utilizada para crear una transición entre el último estado de negocio generado y el estado de negocio anterior, cuando ya existen objetos *Transition* creados en la colaboración. Esta condición es representada en la restricción OCL asociada al patrón LHS. La semántica de dicho patrón es la siguiente. Se debe encontrar un estado de negocio (*PreviousBS*) en la colaboración binaria pasada como parámetro, la cual debe contener el último estado de negocio generado. Además, se debe satisfacer la expresión OCL, que establece que el objeto *PreviousBS* encontrado debe ser el estado de negocio anterior al último estado generado. La expresión indica que, en la colaboración binaria, debe existir una

transición en donde tenga como salida de la transición al objeto *PreviousBS* encontrado, y no debe existir una transición que tenga como entrada al objeto *PreviousBS* encontrado.

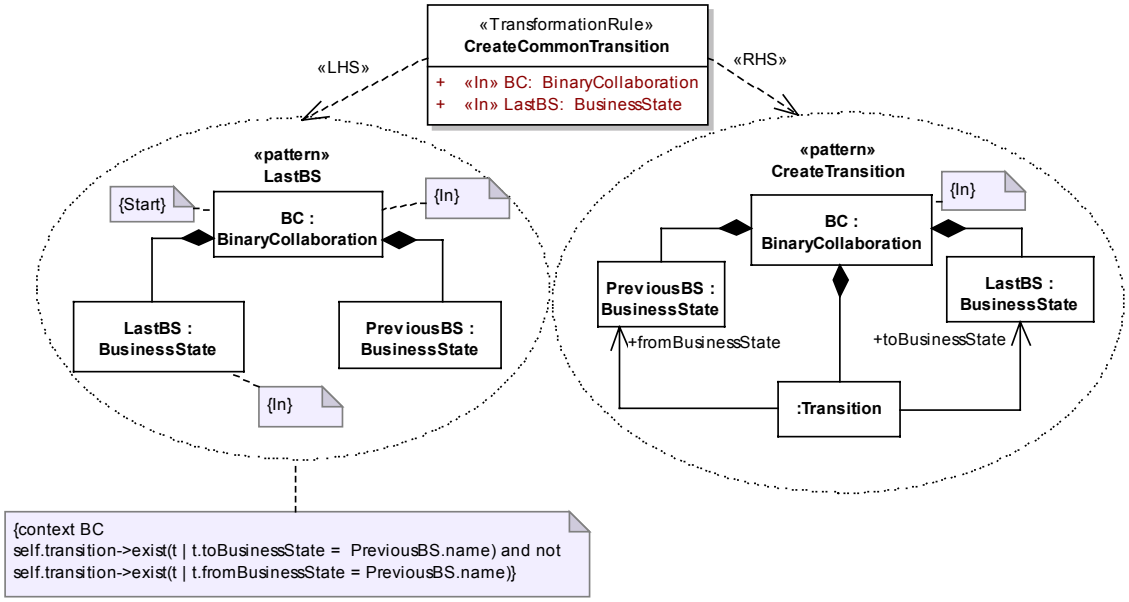


Figura 6-27. Regla de Transformación *CreateCommonTransition*

Debido a que los mensajes de negocio pueden tener adicionados condiciones que restringen el envío de dicho mensaje, las mismas también deberían ser expresadas en BPSS. Estas pueden ser expresadas estableciendo una expresión condicional en la transición que tiene como destino al último estado de negocio generado. Esto es realizado a través de la regla *CreateConditionOnTransition*.

Todas las condiciones expresadas en UP-ColBPIP, tanto las definidas sobre los mensajes como aquellas asociadas a los caminos de interacción de los segmentos, son definidas en lenguaje natural o en OCL. No obstante, BPSS requiere que dichas condiciones (definidas en las transiciones o en los estados *Success* y *Failure*) sean expresadas en el lenguaje XPath (W3C, 1999b). Las reglas de transformación definidas anteriormente no realizan la conversión de las condiciones definidas en lenguaje natural o en OCL a XPath. Por lo tanto, luego de generada la especificación en BPSS, los desarrolladores deben modificar las condiciones generadas e incluir la definición de las mimas usando la sintaxis de XPath.

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

A continuación se muestra el código generado en la generación de las transiciones correspondientes a los primeros dos fragmentos transformados en el protocolo *BlanketPurchaseOrder*. Sólo el código resaltado es generado por las reglas descritas anteriormente.

```
<BinaryCollaboration name = "BC:Blanket Purchase Order"
            timeToPerform = "P6D">
      <Role name = "Customer" nameID = "01A"/>
      <Role name = "Supplier" nameID = "01B"/>
      <BusinessTransactionActivity name = "BTA:propose_BlanketPurchaseOrder"
            fromRole = "Customer" toRole = "Supplier"
            businessTransaction = "propose_BlanketPurchaseOrder"/>
      <Start toBusinessState = "BTA:propose_BlanketPurchaseOrder"/>
      <CollaborationActivity name = "CA:BPO Negotiation(For)"
            binaryCollaboration = "BC:BPO Negotiation(For)"
            fromRole = "Customer" toRole = "Supplier"/>
      <Transition fromBusinessState = "BTA:propose_BlanketPurchaseOrder"
            toBusinessState = "CA:Negotiation(For)"/>
      <Success fromBusinessState = "CA:Negotiation(For)"
            ConditionGuard = "Success"/>
   </BinaryCollaboration>
```

6.2.2. Transformación de UP-ColBPIP a CPPA

Para poder llevar a cabo la ejecución de los protocolos de interacción generados en BPSS, es necesario definir las interfaces de los SAPC de los socios con el lenguaje CPPA. Para cada socio de negocio involucrado en los procesos, se crea una especificación CPP de su interfaz.

Estas especificaciones podrían ser generadas de dos maneras diferentes: a partir de un modelo BPSS o a partir de un modelo UP-ColBPIP. Aquí se describe este último caso. Para ello, se debe tener en cuenta tanto la vista de las interfaces de negocio como la vista de los protocolos de interacción de un modelo UP-ColBPIP. Las especificaciones CPP de los socios no pueden ser derivadas en forma directa desde la vista de las interfaces de negocio, debido a que es necesario indicar en las mismas los procesos en los cuales el socio puede enviar y recibir mensajes. Por lo tanto, estas especificaciones también son obtenidas analizando las responsabilidades de los roles que cumplen los socios en los protocolos de interacción.

A continuación se describe en términos conceptuales las reglas utilizadas en esta transformación y el orden en que las mismas son ejecutadas. Para ejemplificar la salida final de esta transformación, se utiliza también el modelo UP-ColBPIP del caso de estudio descrito en el capítulo 4 (Sección 4.4). En este modelo, la colaboración B2B consiste de dos socios de negocio “Pincelap S.A” y “Tractores Puma S.A.”, los cuales

desempeñaban los roles de *Customer* y *Supplier* respectivamente. Como ejemplo, sólo se muestran algunas porciones del código XML correspondiente a la especificación CPP del socio "Pincelap S.A".

Los pasos de la transformación son los siguientes:

1. Regla *GetB2BCollaboration*. Esta regla obtiene el nombre de la colaboración B2B.
2. Regla *CreateRole*: Esta regla genera la información del socio, la especificación de procesos de BPSS que contiene los procesos en los cuales está involucrado el socio, y el rol que éste desempeña en dichos procesos. El rol del socio es obtenido a partir de la colaboración B2B definida en el modelo UP-ColBPIP. El nombre del socio debe ser pasado como información de entrada al momento de ejecución de la transformación. El código correspondiente que es generado en la especificación CPP del socio "Pincelap S.A." es el siguiente:

```
<CollaborationProtocolProfile...>
    <PartInfo partyName="Pincelap S.A." ...>
    ...
    <CollaborationRole>
        <ProcessSpecification version="1.0"
               name="VMI-based Collaborative Replenishment"
               xlink:href="VMI-based_Collaborative_Replenishment.xml"/>
        <Role name="Customer"
               xlink:href="VMI-based_Collaborative_Replenishment.xml#CustomerID">
```

3. Regla *GetInteractionProtocol*: Esta regla obtiene un protocolo de interacción. Los mensajes que un socio puede enviar o recibir serán generados a partir de los mensajes definidos en los protocolos. Esta regla es ejecutada mientras existan protocolos definidos. Cada vez que esto ocurra, se ejecutarán las siguientes reglas.
   3.1. Regla *CreateCanSend*: Esta regla es la encargada de crear los mensajes que el socio puede enviar en las colaboraciones binarias que representan los protocolos de interacción en la especificación de procesos BPSS. En UP-ColBPIP, cada uno de estos mensajes es obtenido observando la *lifeline* del rol en el protocolo de interacción obtenido en la regla anterior. Más específicamente, un mensaje *CanSend* es creado a partir de un mensaje que un rol envía, el cual es obtenido a partir de una ocurrencia de evento definida sobre la *lifeline* del rol que representa el envío del mensaje. Esta regla es ejecutada mientras se encuentre un mensaje de negocio en el protocolo de interacción. Como ejemplo, el siguiente código muestra el código generado para el mensaje *accept-proposal(BPOChangeResponse)* enviado por el rol *Customer* en el protocolo *BlanketPurchaseOrder*.

```
<ServiceBinding>
    <CanSend>
        <ThisPartyActionBinding id="Customer" action="accept-proposal" ...>
            <ActionContext
                 binaryCollaboration="BC:Blanket Purchase Order"
                 businessTransactionActivity="BTA:accept-proposal_BPOChangeResponse"
                 requestOrResponseAction="propose">
            ...
        </ThisPartyActionBinding>
    </CanSend>
```

3.1.1. Regla *CreateActionContextForSend*. Cada uno de los mensajes que un socio puede enviar, se corresponden con una actividad de solicitud en una transacción de negocio, ejecutada dentro de una colaboración binaria por una actividad que tiene al rol del socio como iniciador de la misma. Por lo tanto es necesario crear una referencia a cada uno de estos elementos. Esta regla realiza esta función. Estos elementos son generados a partir del nombre del protocolo de interacción y del mensaje de negocio enviado por el rol. En el código anterior, esta regla es la encargada de crear los atributos del elemento *ActionContext* para la acción *accept-proposal* creada con la regla anterior.

3.2. Regla *CreateCanReceive*. Esta regla es la encargada de crear los mensajes que el socio puede recibir en las colaboraciones binarias que han sido generadas en la especificación de procesos BPSS. Esta regla es similar a la regla *CreateCanSend*. La diferencia consiste en que un *CanReceive* es creado a partir de un mensaje que un rol recibe, el cual es obtenido a partir de una ocurrencia de evento definida sobre la *lifeline* del rol que representa la recepción del mensaje. Como ejemplo, el siguiente código muestra el código generado a partir del primer mensaje *propose(BlanketPurchaseOrder)* del protocolo *BlanketPurchaseOrder*.

```
<CanReceive>
       <ThisPartyActionBinding id="Customer" action="propose" ...>
           <ActionContext binaryCollaboration="BC:Blanket Purchase Order"
                businessTransactionActivity="BTA:propose_BlanketPurchaseOrder"
                requestOrResponseAction="propose">
           ...
       </ThisPartyActionBinding>
   </CanReceive>
```

3.2.1. Regla *CreateActionContexForReceive*. Esta regla es similar a la regla *CreateActionContextForSend*. La diferencia consiste en que cada uno de los mensajes que un socio puede recibir, se corresponden con una actividad de solicitud en una transacción de negocio, ejecutada dentro de una colaboración binaria por una actividad que tiene al rol del socio como el rol

que responde. En el código anterior, regla es la encargada de los atributos del elemento *ActionContext*, para la acción *propose* creada con la regla anterior.

Las especificaciones CPP de cada socio, generadas con esta transformación, no son completas, como en el caso del modelo de transformación de UP-ColBPIP a BPSS, en donde las especifiaciones de procesos en BPSS son completas. Esto quiere decir que cada especificación CPP generada para un socio, es sólo un esqueleto. Los restantes elementos que definen los aspectos operacionales de más bajo nivel, deberán ser adicionados por los desarrolladores.

6.3 Conclusiones

En este capítulo se han descrito las transformaciones involucradas en el método de desarrollo conducido por modelos de procesos colaborativos, para llevar a cabo la generación automática de especificaciones de procesos e interfaces de los socios basadas en ebXML, a partir de modelos independientes de la tecnología basados en UP-ColBPIP. La principal ventaja de estas transformaciones es la garantía de que la solución tecnológica generada en ebXML es consistente y se corresponde con lo definido en el nivel de negocio a través de modelos basados en UP-ColBPIP.

Los tiempos y costos que los socios de negocio requieren para generar una solución tecnológica basada en ebXML pueden ser reducidos significativamente a través del método de desarrollo conducido por modelos, debido al soporte para la generación automática de las especificaciones en ebXML a partir de modelos UP-ColBPIP. Además de disminuir los errores en el código XML de estas especificaciones, la complejidad de trasladar la solución alcanzada en el nivel de negocio a una solución tecnológica basada en ebXML también es disminuida, ya que dicha complejidad es salvada y ocultada en las transformaciones definidas.

La generación de una solución tecnológica en ebXML consiste en generar especificaciones de procesos colaborativos utilizando el lenguaje BPSS y las especificaciones de los SAPC de los socios utilizando el lenguaje CPPA. Por lo tanto, en este capítulo se han desarrollados y descritos los principales componentes y técnicas que soportan este método de desarrollo conducido por modelos: las transformaciones de modelos UP-ColBPIP a BPSS, las transformaciones de modelos UP-ColBPIP a CPPA,

y los metamodelos de BPSS y CPPA que permiten construir modelos basados en dichos lenguajes. Ambas transformaciones han sido definidas utilizando el método y la herramienta de transformación de modelos propuestas en el capítulo 5.

Con respecto a la transformación de UP-ColBPIP a BPSS, por un lado, se han definido las diferentes reglas de transformación, a través de las cuales se establece la correspondencia entre los conceptos provistos por UP-ColBPIP y los conceptos provistos por BPSS. Para cada regla se ha definido su semántica operacional en un alto nivel de abstracción utilizando el lenguaje de modelado de reglas. Estas definiciones de reglas constituyen la gramática de grafos definida para transformaciones de modelos UP-ColBPIP a modelos BPSS.

Por otro lado, los patrones de dichas reglas han sido implementados utilizando la API provista por la herramienta de transformación. A través del lenguaje de transformación de modelos soportado por esta herramienta se ha definido el modelo de transformación que establece cómo dichas reglas son compuestas en un orden lógico adecuado, para poder llevar a cabo la ejecución de las transformaciones de UP-ColBPIP a BPSS. Como se puede observar en los árboles de reglas que componen los módulos definidos en dicho modelo, existe un considerable número de reglas que han sido reutilizadas. Más aún, la posibilidad de separar las reglas en módulos y realizar invocaciones recursivas a los mismos ha posibilitado un proceso de transformación más modular y manejable. De lo contrario, sin estas reutilizaciones de módulos y reglas, dicha transformación sería aún mas complicada y requeriría de un mayor esfuerzo.

Como resultado de las transformaciones definidas, es posible inferir que la mayoría de los elementos que componen una especificación de procesos en BPSS pueden ser derivados a partir de modelos UP-ColBPIP. Esto significa que el lenguaje UP-ColBPIP provee los elementos de modelado conceptual requeridos no sólo para diseñar procesos colaborativos en un alto nivel de abstracción sino también para derivar especificaciones ejecutables en un estándar B2B basado en transacciones de negocio, como ebXML. Los únicos elementos que no pueden ser derivados son aquellos que constituyen detalles de bajo nivel que corresponden a los aspectos de configuración necesarios para la ejecución de los procesos.

Por otra parte, dado que el lenguaje utilizado en UP-ColBPIP para definir las condiciones (en mensajes y caminos de interacción) es diferente al requerido en BPSS,

los desarrolladores deberán modificar dichas condiciones una vez generada la especificación BPSS con la herramienta de transformación de modelos. No obstante, estas condiciones podrían ser modificadas en el modelo UP-ColBPIP antes de su transformación. De esta manera, es posible realizar dichas modificaciones en un alto nivel de abstracción y no en el código XML. Otra alternativa es realizar estas modificaciones cuando se genera el modelo basado en BPSS, previo a su transformación a código XML. Esto también es factible. No obstante, actualmente la herramienta no genera el archivo XMI correspondiente a este modelo para que el mismo pueda luego ser modificado.

Con respecto a la transformación de UP-ColBPIP a CPPA, esta posibilita la generación de un esqueleto de la especificación CPP que define la interfaz de cada socio. Esto se debe a que en un CPP se definen no solo los aspectos de las interfaces correspondientes al soporte de las colaboraciones y transacciones de negocio definidas en una especificación en BPSS, sino que también se definen los aspectos técnicos de dichas interfaces correspondientes a parámetros de configuración de bajo nivel. Estos últimos no son derivados desde modelos UP-ColBPIP. No obstante, los otros aspectos referidos a una especificación BPSS pueden ser derivados en forma completa desde un modelo UP-ColBPIP.

En resumen, en este capítulo se ha demostrado cómo a partir de una solución en el nivel de negocio es posible generar una solución tecnológica basada en ebXML, en el marco de un proceso de desarrollo conducido por modelos para procesos colaborativos, utilizando el lenguaje de modelado UP-ColBPIP y el método de transformación de modelos definidos en los capítulos anteriores.

Capítulo

7

TRANSFORMACIÓN DE MODELOS UP-COLBPIP A BPEL

En este capítulo, en primer lugar se describen los lenguajes BPEL y WSDL. Luego como parte del método de desarrollo conducido por modelos de procesos colaborativos se describen: la transformación de modelos UP-ColBPIP a BPEL y la transformación de modelos UP-ColBPIP a WSDL. Finalmente se presentan las conclusiones.

7.1. Composición de Servicios Web

Soluciones tecnológicas basadas en servicios web están teniendo cada vez más interés para la implementación de sistemas de información B2B. Esto se debe a que los servicios web proveen varias ventajas para llevar a cabo interacciones entre componentes de diferentes empresas. Un servicio web es un componente autónomo y distribuido, el cual es definido e implementado en forma completamente independiente por una empresa.

Existen varios lenguajes que han surgidos para definir procesos de negocio basados en servicios web. Por lo tanto, otra solución tecnológica disponible para dar soporte a la ejecución de los procesos colaborativos es el uso de procesos que involucran un conjunto de servicios web. Esto es conocido como *Composición de Servicios Web*, en donde la composición es definida en base a un proceso de negocio. A través de servicios web, las empresas exponen funciones de negocio que pueden ser invocadas por las aplicaciones de sus socios vía Internet. Para establecer el orden en que dichas invocaciones deben ser realizadas se define un proceso como un servicio web compuesto. Ejemplo de lenguajes que dan soporte a la especificación de procesos basados en servicios web son: *Business Process Execution Languaje (BPEL)* y *Web Service Choreography Interface (WSCI)*. Estos lenguajes se basan en el *Web Service Description Language (WSDL)*, el cual permite definir un servicio web. Este capítulo se enfoca en la generación de soluciones tecnológicas basadas en BPEL. Por lo tanto, primero se describe WSDL y luego se describe BPEL.

7.1.1. El lenguaje Web Service Description Language

Web Service Description Language (WSDL) (W3C, 2001) permite definir las interfaces de los socios como Servicios Web desacoplados utilizando una sintaxis XML.

Los elementos utilizados en WSDL son los siguientes:

- Tipos (Types). Un contenedor de definiciones de tipos de datos usando algún sistema de representación, tales como esquemas XML.
- Mensaje (Message). Una definición abstracta de la información a ser intercambiada.
- Operación (Operation). Una descripción abstracta de una acción soportada por un servicio.
- Tipo de Puerto (Port Type). Un conjunto abstracto de operaciones soportadas por uno o más puertos.
- Binding. Una especificación del formato de dato y el protocolo de comunicación concreto a ser utilizados.
- Puerto (Port). Un punto final simple definido como una combinación de un binding y una dirección de red.
- Servicio (Service). Una colección de puertos relacionados.

Este lenguaje separa la definición abstracta de las interfaces, de los detalles técnicos que deben ser configurados para implementar luego los servicios. La definición abstracta de las interfaces incluye los tipos de puertos, las operaciones definidas en dichos tipos y los mensajes intercambiados a través de dichas operaciones. Los lenguajes de procesos basados en servicios web sólo hacen referencia a estos elementos abstractos de las interfaces WSDL.

7.1.2. El Lenguaje Business Process Execution Languaje

Business Process Execution Languaje (BPEL) (BEA y otros, 2003) permite especificar procesos de negocio expresados en términos de un flujo de control que involucra operaciones provistas por uno o varios servicios web. Este lenguaje está basado en XML. Según la terminología utilizada en BPEL, dos tipos de procesos pueden ser definidos: procesos *abstractos* y procesos *ejecutables*.

Un proceso abstracto es utilizado para definir los denominados *protocolos de negocio o de conversación*. Estos protocolos se refieren al intercambio de mensajes entre un servicio web compuesto y un cliente del servicio, sin definir la lógica de negocio interna del servicio. En otras palabras, un protocolo de negocio define el orden en el cual un socio particular envía mensajes hacia o espera recibir mensajes desde sus socios para alcanzar una meta de negocio (Leymann y Roller, 2004). En un proceso abstracto sólo se definen los datos relevantes al protocolo de negocio, el cual describe los aspectos públicos de las interacciones entre un participante y sus socios.

Por otra parte, un proceso ejecutable define la lógica de negocio interna que modela el comportamiento de un participante en una interacción con otros socios. En esta lógica se incluye el manejo de los datos privados del socio, las reglas de negocio y actividades internas que determinan los caminos alternativos a seguir, y las invocaciones a operaciones de servicios web privados que exponen las funciones de las aplicaciones internas del socio.

BPEL se basa en un modelo conceptual básico que permite describir tanto procesos abstractos como procesos ejecutables. De esta manera, un proceso ejecutable puede ser visto como el refinamiento de uno abstracto. Este último representa el esqueleto a ser tenido en cuenta para definir el proceso ejecutable que soporte la interacción de una empresa con sus socios. Por lo tanto, un proceso ejecutable es una mezcla de aspectos privados y públicos de un socio.

Una característica importante de BPEL es que, tanto el proceso abstracto como el proceso ejecutable son definidos desde el punto de vista de una única empresa. Por lo tanto, un protocolo de negocio describe el comportamiento de un único rol desempeñado por uno de los socios en un proceso colaborativo.

Dado que estos conceptos de procesos en BPEL, utilizados en el nivel tecnológico, son similares a los conceptos utilizados y propuestos en esta tesis para definir procesos colaborativos en el nivel de negocio, es necesario establecer la correspondencia entre los mismos. La Figura 7-1 muestra dicha correspondencia. Un proceso abstracto BPEL define los aspectos públicos de la relación de un socio con otros. Luego se pueden utilizar procesos abstractos para implementar el comportamiento de los procesos colaborativos. La semántica de un proceso colaborativo puede ser representada con BPEL definiendo un proceso abstracto por cada socio involucrado en el mismo. De esta

manera, un proceso colaborativo en el nivel tecnológico es definido como un conjunto de procesos abstractos, cada uno describiendo el comportamiento de un socio.

Los procesos privados que dan soporte al rol que el socio desempeña en los procesos colaborativos son especificados a través de procesos ejecutables BPEL, de acuerdo a lo definido en los respectivos procesos abstractos.

Por lo tanto, para especificar procesos colaborativos a través de BPEL, este capítulo se enfoca en la generación automática de procesos abstractos.

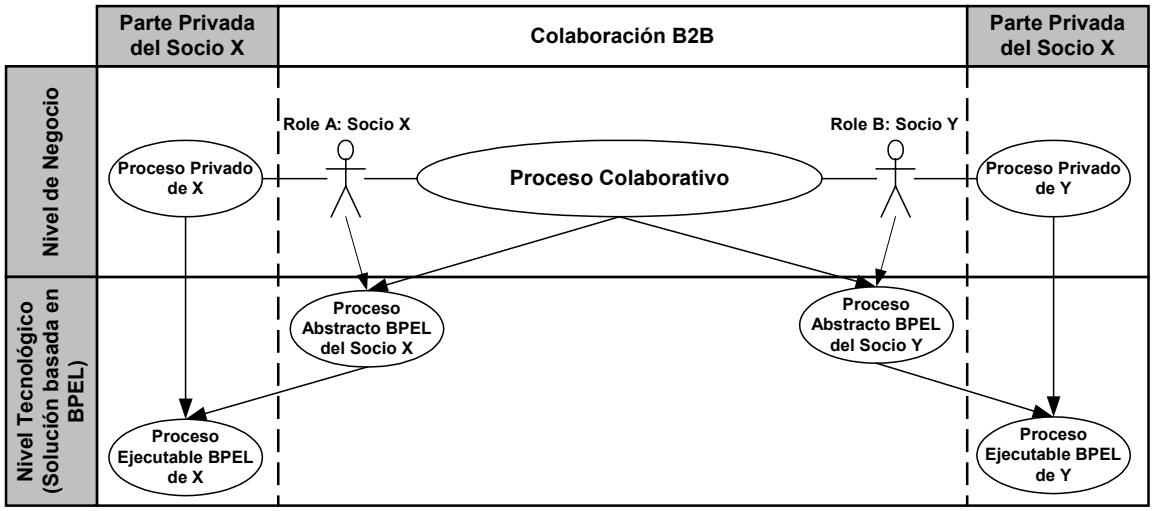


Figura 7-1. Correspondencia entre los tipos de procesos del nivel de negocio y los tipos de procesos BPEL

7.1.2.1. Conceptos de BPEL para Especificar Procesos de Negocio

Los principales conceptos provistos por BPEL para especificar procesos abstractos se describen a continuación. Los procesos en BPEL están basados en el concepto de *actividad*, a través del cual se definen las invocaciones a las operaciones de los tipos de puerto de los servicios web. Una actividad en un proceso BPEL es *primitiva* o *estructurada*. Las actividades primitivas son:

- invoke, permite invocar una operación en un tipo de puerto ofrecido en el servicio web de un socio. La operación puede ser sincrónica (solicitud/respuesta) o asincrónica (en un solo sentido).
- receive, permite esperar la recepción de un mensaje desde un socio. Esta actividad especifica el tipo de puerto y operación que se espera que los socios invoquen. Un proceso de negocio en BPEL provee servicios a sus socios a

través de actividades receive. Además, este tipo de actividad es utilizado para instanciar un proceso, cuando el atributo createInstance tiene asignado el valor "yes".

- reply, permite enviar una respuesta a una solicitud previamente aceptada a través de una actividad receive. Estas respuestas sólo tienen significado para interacciones sincrónicas.
- throw, permite disparar y señalar explícitamente una falla o excepción en el proceso.
- wait, permite especificar una demora para un cierto período de tiempo o hasta que cierta fecha límite sea alcanzada.
- assign, permite copiar datos desde una variable a otra.
- empty, permite expresar una actividad en donde no se debe realizar acción alguna. Utilizada para sincronización de actividades concurrentes.

Las actividades estructuradas son definidas usando los siguientes constructores de flujos de control:

- sequence, permite definir un conjunto de actividades a ser ejecutadas en forma secuencial.
- switch, permite representar un comportamiento condicional con varias alternativas posibles.
- while, permite representar la repetición de varias actividades mientras se cumpla un criterio dado.
- pick, permite representar un conjunto de alternativas, de las cuales sólo una puede ser ejecutada. Esta actividad es definida como un conjunto de eventos, los cuales si ocurren ejecutan la actividad asociada a los mismos. El proceso se bloquea y espera la ocurrencia de uno de estos eventos. Dos tipos de eventos pueden ocurrir: un mensaje (elemento onMessage) o una alarma basada en un intervalo o duración de tiempo. El evento onMessage es equivalente a una actividad receive.
- flow, permite representar la ejecución en paralelo de actividades. Esta actividad también puede ser utilizada para establecer dependencias entre las actividades

definidas dentro de esta actividad estructurada. Para establecer dichas dependencias se utilizan los vínculos (links) de control. En la actividad de origen se indica el vínculo que representa la dependencia a través del elemento source de la actividad. En la actividad de destino se indica el mismo vínculo a través del elemento target de dicha actividad. De esta manera, la actividad de destino podría sólo comenzar cuando la actividad de origen ha finalizado.

- scope, permite definir una actividad anidada en donde se agrupan un conjunto de actividades con sus propias variables, con un mismo manejador de compensación, y con sus propios manejadores de fallas.

Los restantes conceptos utilizados en BPEL son descritos en la transformación de modelos UP-ColBPIP a BPEL.

7.2. Transformación de UP-ColBPIP a BPEL y WSDL

Siguiendo los lineamientos del método propuesto basado en el desarrollo conducido por modelos de procesos colaborativos (Sección 3.3.2), la generación de especificaciones de procesos basados en BPEL, a partir de modelos conceptuales definidos con UP-ColBPIP, es realizada a través de los componentes y técnicas mostrados en la Figura 7-2.

Los modelos independientes de la plataforma son aquellos modelos de procesos colaborativos definidos con UP-ColBPIP. Los modelos dependientes de la plataforma son aquellos basados en BPEL y WSDL. El primero representa los modelos de procesos dependientes de la tecnología. El segundo representa los modelos de las interfaces dependientes de la tecnología de los SAPC de los socios.

Varios modelos BPEL deben ser generados a partir de un modelo UP-ColBPIP. Esto se debe a dos razones. Por un lado, un modelo BPEL especifica un único proceso. Entonces, cada modelo BPEL generado corresponderá a un proceso abstracto y debido a que dicho proceso es generado a partir de un protocolo de interacción de un modelo UP-ColBPIP, existirá en principio un proceso abstracto por cada protocolo definido. Por otro lado, debido a que cada proceso abstracto BPEL representa el comportamiento del rol que un socio desempeña en un proceso colaborativo, por cada protocolo de interacción se generarán varios procesos abstractos, uno por cada rol del protocolo. En

resumen, para un modelo UP-ColBPIP, el número de procesos abstractos *PA* a ser generados está dado por la siguiente fórmula:

$$PA = \sum_{pi=1}^{CantPI} R_{pi}$$

en donde *CantPI* es el número de protocolos de interacción definidos en el modelo UP-ColBPIP y R_{pi} es el número de roles involucrados en el protocolo *pi*.

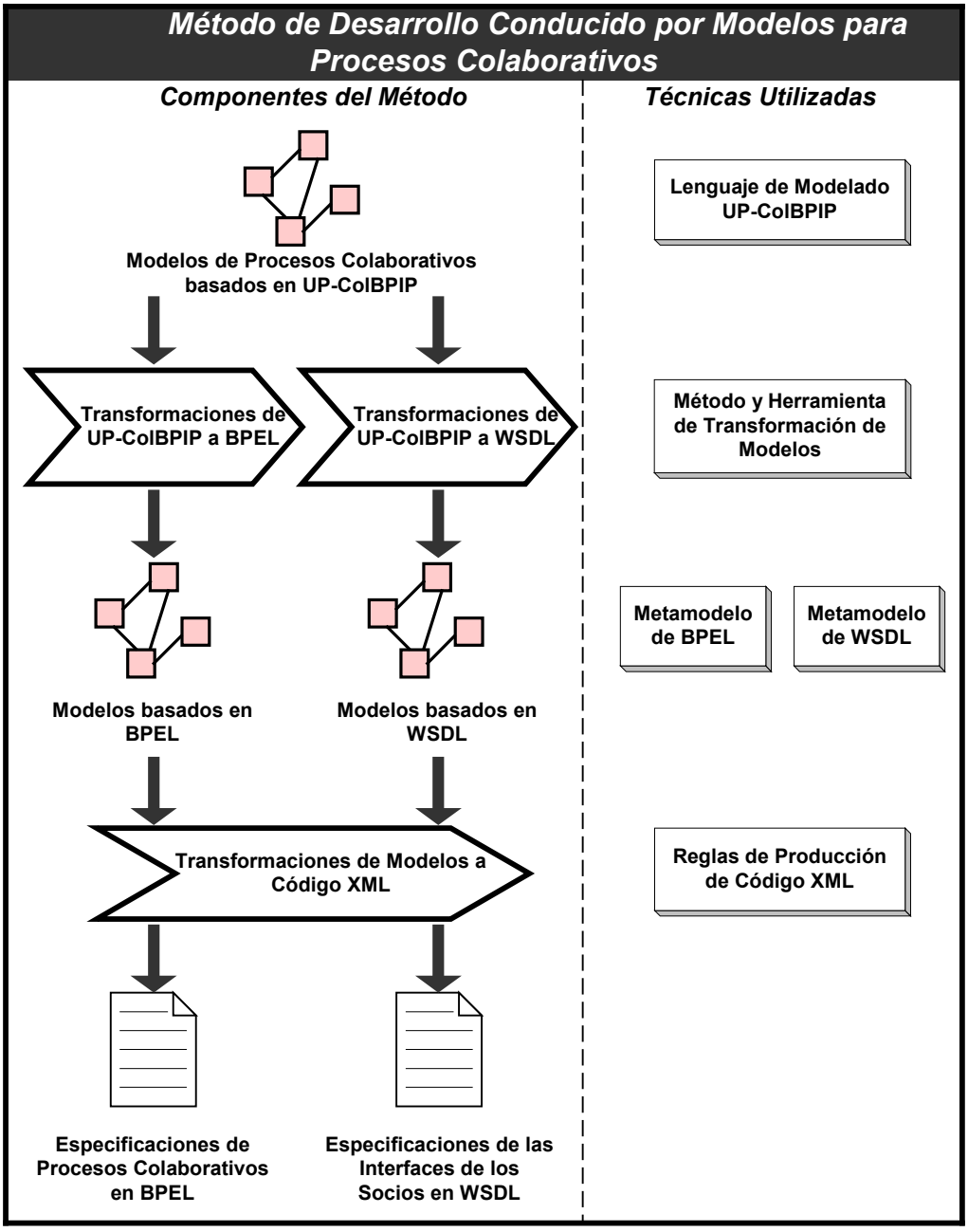


Figura 7-2. Técnicas y componentes del método de desarrollo conducido por modelos para generar especificaciones de procesos en BPEL/WSDL

Por lo tanto, se deben ejecutar *PA* transformaciones por cada modelo UP-ColBPIP, con el objetivo de obtener como salida final las especificaciones de los procesos abstractos BPEL correspondientes. Estas transformaciones son ejecutadas en base a un mismo modelo, cuyo módulo principal de transformación de UP-ColBPIP a BPEL tendrá dos parámetros de entrada: el protocolo a ser transformado y el rol para el que se generará el proceso BPEL. En cada ejecución, se deberá asignar valor a dichos parámetros.

Para el caso de modelos WSDL, se debe generar uno por cada socio definido en un modelo UP-ColBPIP, ya que las interfaces WSDL de los SAPC de los socios son derivadas desde las interfaces de negocio de los roles. Por lo tanto, el número de transformaciones a ser ejecutadas para obtener como salida final las especificaciones WSDL, es igual al número de socios definidos en el modelo UP-ColBPIP. Dichas transformaciones también son ejecutadas a través de un mismo modelo.

La aplicación de este método de desarrollo conducido por modelos posibilita:

- La generación de una solución tecnológica basada en servicios web para ejecutar procesos colaborativos, a partir de modelos conceptuales basados en UP-ColBPIP.
- La garantía de que la solución tecnológica basada en BPEL es consistente y se corresponde con lo definido en el nivel de negocio.
- La reducción de la distancia semántica entre los conceptos provistos por UP-ColBPIP en el nivel de negocio y los conceptos provistos por BPEL en el nivel tecnológico.

Por otra parte, este método permite garantizar que los procesos abstractos de cada uno de los socios sean compatibles entre sí, ya que son generados a partir del mismo modelo conceptual y de la misma definición de transformación. Esto es un requerimiento muy importante cuando se generan soluciones basadas en BPEL. Si los socios definen en forma separada sus procesos abstractos no existe garantía de que dichos procesos sean compatibles. Por lo tanto, este método contribuye a crear soluciones consistentes basadas en BPEL para soportar la ejecución de procesos colaborativos.

A continuación se describe a nivel conceptual las correspondencias entre los conceptos provistos por UP-ColBPIP y los provistos por BPEL y WSDL. A través de dichas correspondencias es posible especificar los modelos de transformación de UP-ColBPIP a dichos lenguajes. Para ejemplificar la salida final de algunas de estas transformaciones se utiliza nuevamente el modelo UP-ColBPIP descrito en el caso de estudio del capítulo 4.

7.2.1. Transformación de Modelos UP-ColBPIP a WSDL

La generación de las interfaces WSDL de cada uno de los socios puede ser derivada en forma casi directa desde la vista de las interfaces de negocio de un modelo UP-ColBPIP (Tabla 7-1). Para cada rol se define un servicio en WSDL. A partir de una interfaz de negocio pública del rol que cada socio cumple en la colaboración B2B, se deriva un tipo de puerto en WSDL. En UP-ColBPIP la interfaz pública tiene definida una interfaz provista y una interfaz requerida. En WSDL, un tipo de puerto representa sólo la interfaz provista. Por lo tanto, las operaciones del tipo de puerto son derivadas a partir de los servicios de negocio definidos en la interfaz provista del rol.

Concepto de UP-ColBPIP	Concepto de WSDL
Role	Service
PublicBusinessInterface	PortType
BusinessService (correspondiente a la interfaz provista)	Operation
BusinessDocument	Message

Tabla 7-1. Correspondencia entre conceptos de UP-ColBPIP y de WSDL

Debido a que en UP-ColBPIP los mensajes de negocio son asincrónicos, los servicios de negocio sólo tienen un parámetro de entrada que representa el documento de negocio a ser intercambiado. Por lo tanto, en WSDL las operaciones son definidas en forma asincrónica con un mensaje de entrada. Operaciones de tipo solicitud/respuesta no son generadas. El documento de negocio en UP-ColBPIP es transformado en un mensaje en WSDL.

Como ejemplo, el siguiente código muestra un fragmento del documento WSDL que se corresponde con la interfaz provista del rol *Customer* definido en el modelo UP-

ColBPIP del caso de estudio. Esta interfaz deberá ser soportada por el SAPC del socio que cumple con el rol *Customer*.

```
<wsdl:definitions ...>
<message name="Msg:BlanketPurchaseOrder"/>
<message name="Msg:BPOChange"/>
<message name="Msg:BPOResponse"/>
....
<portType name="PT:Customer_InterfaceWithSupplier">
   <operation name="propose_BPOChange">
        <input message="Msg:BPOChange"/>
   </operation>
   <operation name="accept-proposal_BPOResponse">
        <input message="Msg:BPOResponse"/>
   </operation>
   <operation name="inform_ShipNotice">
        <input message="Msg:ShipNotice"/>
   </operation>
   ....
   <operation name="AckReceipt">
        <input message="Msg:ReceiptAcknowledgement"/>
   </operation>
   <operation name="AckRead">
        <input message="Msg:ReadAcknowledgement"/>
   </operation>
</portType>
```

Existen otras operaciones que son generadas en la interfaz WSDL, cuando se transforman modelos UP-ColBPIP a BPEL, las cuales permiten representar otros conceptos manejados en la vista de los protocolos de interacción. Por ejemplo, para representar los acuses de recibo de los mensajes de los protocolos de interacción, se definen dos operaciones en WSDL correspondientes a los tipos de acuses de recibo. Además, para cada protocolo se definen dos operaciones en los tipos de puerto de cada rol: una operación “de comienzo” y otra “de fin”. La creación de estas operaciones es discutida luego en la transformación a BPEL. Por lo tanto, en dicha transformación, no sólo se debe considerar como modelo de entrada el modelo UP-ColBPIP, sino también el modelo WSDL que contiene la interfaz del rol para el que se genera el modelo BPEL.

7.2.2. Transformación de Modelos UP-ColBPIP a BPEL

Los procesos abstractos BPEL de cada socio pueden ser derivados a partir de la vista de los protocolos de interacción de un modelo UP-ColBPIP. Más precisamente, a partir de un protocolo de interacción es posible generar el proceso abstracto de cada socio involucrado en dicho protocolo. El proceso abstracto BPEL define el comportamiento y responsabilidades del rol que cumple el socio en el proceso colaborativo. En un protocolo de interacción, el comportamiento y responsabilidades de los roles son definidos en las *lifelines* que representan a cada uno de ellos. Por lo tanto, los elementos que componen un proceso abstracto que describe el rol de un socio,

pueden ser derivados recorriendo los elementos definidos sobre la *lifeline* que representa a dicho rol en el protocolo. En otras palabras, un proceso abstracto BPEL de un socio puede ser visto como la implementación de la *lifeline* del rol que dicho socio desempeña en un protocolo de interacción.

A continuación se describe la transformación de cada uno de los conceptos de UP-ColBPIP a BPEL.

7.2.2.1. Estableciendo la relación entre los roles del proceso BPEL y las interfaces WSDL

El primer paso en la transformación, previo a la generación de los procesos abstractos, es establecer la relación entre los roles de los procesos BPEL y las interfaces WSDL. Aunque un proceso BPEL representa uno de los roles de un protocolo de interacción, es necesario indicar en el mismo las interfaces WSDL de los otros socios que serán utilizadas en el proceso. Para ello, BPEL provee el elemento *PartnerLinkType*. El mismo no es adicionado como parte de la definición de un documento BPEL, sino que debe ser adicionado en los documentos WSDL que contienen las interfaces definidas. En dicho elemento se definen las relaciones entre los servicios de los socios y los roles de cada socio. Cada rol hace referencia a un tipo de puerto definido en WSDL. Estos roles son utilizados en cada proceso BPEL. La definición de un *PartnerLinkType* es la misma para todos los procesos abstractos a ser generados.

El elemento *partnerLinkType* y los elementos que lo componen pueden ser derivados a partir de los roles y socios de negocio definidos en la colaboración B2B del modelo UP-ColBPIP, y a partir de las interfaces de negocio públicas de cada rol.

Para el caso de estudio, el siguiente código muestra la salida correspondiente. Dos roles han sido definidos de acuerdo a los involucrados en la colaboración B2B. Cada rol hace referencia al tipo de puerto correspondiente.

```
<partnerLinkType name="PLT:CustomerSupplier">
   <role name="Customer">
      <portType name="PT:Customer_InterfaceWithSupplier"/>
   </role>
   <role name="Supplier">
      <portType name="PT:Supplier_InterfaceWithCustomer"/>
   </role>
</partnerLinkType>
```

7.2.2.2. Transformación de Protocolos de Interacción

Como se indicó anteriormente, por cada socio involucrado en un protocolo de interacción, se genera el proceso abstracto BPEL que representa el comportamiento del rol que el socio cumple en dicho protocolo. A partir de ahora el socio y el rol que el mismo desempeña, será referido como el socio y el rol propietario del proceso abstracto BPEL, para diferenciarlos de los restantes socios y roles.

Para el caso de estudio, a continuación se muestra el fragmento de código del proceso abstracto BPEL que representa al rol *Customer,* desempeñado por uno de los socios, en el protocolo de interacción *Blanket Purchase Order*. El nombre dado al proceso es el nombre del protocolo seguido del nombre del rol que desempeña el socio.

```
<process name="BlanketPurchaseOrder_Customer" ....
       abstractProcess="yes">
```

Al comienzo de la definición de un proceso, se deben definir los vínculos de socio (*partnerLink*), los cuales especifican las relaciones del propietario del proceso con sus socios. Se debe definir un *partnerLink* por cada socio. En el mismo se debe indicar el tipo de vinculo de socio (atributo *partnerLinkType*), el rol que el propietario del proceso cumple (atributo *myRole*) y el rol del socio (atributo *partnerRole*).

Para el caso de estudio, en el proceso del cliente que representa su parte en el protocolo de interacción *Blanket Purchase Order*, el *partnerLink* definido establece que el rol del propietario del proceso es el *Customer* y el rol del socio con el que interactúa es el *Supplier*.

```
<partnerLinks>
   <partnerLink name="LinkToSupplier"
      partnerLinkType="PLT:CustomerSupplier"
      myRole="Customer"
      partnerRole="Supplier" />
</partnerLinks>
```

7.2.2.3. Transformación de Mensajes de Negocio

Los mensajes de negocio que componen a un protocolo de interacción pueden ser transformados en tres tipos de actividades: *receive* o *pick*, e *invoke*. Cuando un mensaje de negocio es enviado por el rol propietario del proceso, una actividad *invoke* debe ser generada. Dicha actividad contiene la operación correspondiente al mensaje, definida en el tipo de puerto del rol que recibe el mensaje. Actividades de tipo *reply* no son

generadas debido a que los mensajes de negocio en UP-ColBPIP representan una comunicación asincrónica.

Cuando un mensaje de negocio es recibido por el rol propietario del proceso, una actividad *receive* es generada. En este caso, la actividad *receive* define la operación a través de la cual un socio recibe información. Dicha operación forma parte del tipo de puerto del propietario del proceso. Una actividad *receive* es generada cuando el mensaje no tiene asociado una restricción de tiempo. Cuando dicha restricción está presente, el mensaje de negocio es transformado en una actividad *pick*, con dos eventos: un evento *onMessage* que indica la recepción del mensaje a través de la correspondiente operación del tipo de puerto del propietario del proceso; y un evento *onAlarm*, el cual define la restricción de tiempo. Sólo uno de estos eventos puede ocurrir en una actividad *pick*. Por lo tanto, si el mensaje es recibido la actividad *pick* finaliza. De lo contrario, si el mensaje no es recibido dentro del tiempo establecido en el evento *onAlarm*, se ejecutan las acciones definidas en el mismo y luego la actividad *pick* finaliza.

Para representar la excepción de tiempo, en los tipos de puerto de los socios se define una operación *timeException*, ya que BPEL no maneja dicha excepción en forma implícita y automática. Dicha operación debe ser invocada para que informe un rol a otro la situación de que no ha recibido el mensaje en el tiempo solicitado. Esto se realiza a través de una actividad *invoke* definida en el evento *onAlarm*. Los desarrolladores deben luego agregar el código que representa el manejo de dicha excepción. Por ejemplo, pueden agregar una actividad *terminate* para indicar que la instancia del proceso debe finalizar.

El valor del atributo *for* en el evento *onAlarm* debe ser modificado por el desarrollador luego de la generación del código BPEL. Esto se debe a que BPEL sólo acepta la definición de fechas absolutas en un formato específico. En cambio, las restricciones de tiempo en un mensaje de negocio pueden ser definidas con fechas relativas.

Las actividades *invoke*, *receive* y *pick* de un proceso abstracto BPEL pueden ser generadas a partir de las ocurrencias de eventos que representan el envío y recepción de los mensajes en la *lifeline* del rol para el que se está generando el proceso.

Como ejemplo, para el protocolo *Blanket Purchase Order* del caso de estudio, el siguiente código del proceso BPEL del cliente muestra las actividades generadas que

corresponden al primer y segundo mensaje del protocolo. Para el mensaje *propose(BlanketPurchaseOrder)* se genera la actividad *invoke* correspondiente que representa el envío del mensaje por parte del cliente. Para el mensaje *propose(BPOChange)* se genera la actividad *pick* que representa la recepción del mensaje por parte del cliente.

```
   <invoke partnerLink="LinkToSupplier"
           portType="PT:Supplier_InterfaceWithCustomer"
           operation="propose_BlanketPurcahseOrder"
                  inputVariable="BlanketPurcahseOrder"/>
....
   <pick>
      <onMessage partnerLink="LinkToSupplier"
           portType="PT:Customer_InterfaceWithSupplier"
           operation="propose_BPOChange"
                  variable="BPOChange">
      <onAlarm for="t..t+2"/>
   </pick>
```

Por otra parte, no es posible expresar de forma directa en BPEL los acuses de recibo que un mensaje de negocio puede tener definidos. Para ello, en el tipo de puerto de cada rol se adicionan dos operaciones: *acknowledgmentReceipt* y *acknowledgmentRead*. Cuando el rol propietario del proceso es quien envía el mensaje de negocio, luego de la actividad *invoke* correspondiente, se adiciona una actividad *receive* con la operación *acknowledgmentReceipt* y otra igual con la operación *acknowledgmentRead*. Estas actividades permiten representar las recepciones de los correspondientes acuses de recibo. En el caso que el rol propietario del proceso es quien recibe el mensaje de negocio, luego de la actividad *receive* correspondiente, se adiciona una actividad *invoke* con la operación *acknowledgmentReceipt* y otro con la operación *acknowledgmentRead*. Ambas son operaciones del tipo de puerto del otro rol.

Como ejemplo, para el caso que un rol *A* envía un mensaje *m1* a otro rol *B* en un protocolo *X*, en donde dicho mensaje tiene acuses de recibo, se genera el siguiente código:

```
process="X_A"
   ...
   <invoke ...
      portType="B"
      operation="m1"/>
   <receive ...
      portType="A"
      operation="acknowledgementReceipt"/>
   <receive ...
      portType="A"
      operation="acknowledgementRead"/>
   ...
</process>
```

```
<process="X_B"
   ...
   <receive ...
      portType="B"
      operation="m1"/>
   <invoke ...
      portType="A"
      operation="acknowledgementReceipt"/>
   <invoke ...
      portType="A"
      operation="acknowledgementRead"/>
   ...
</process>
```

7.2.2.4. Generación del Comienzo de un Proceso BPEL

En BPEL, un proceso debe comenzar con una actividad *receive* o una actividad *pick*, las cuales deben tener establecido el atributo *createInstance* con un valor "yes". Esto indica que cuando la operación definida en dichas actividades sea invocada, se creará una instancia del proceso.

Aunque estas actividades de inicio podrían ser derivadas desde los mensajes definidos en el protocolo, es necesario adicionar actividades específicas que representen el comienzo del proceso. Esto se debe a que para representar el comienzo de un protocolo de interacción en BPEL, es necesario sincronizar el comienzo de los procesos abstractos que representan al protocolo. Debido a que no existe forma de correlacionar las instancias de dichos procesos que representan la instancia de un protocolo, esta solución propuesta sólo es aplicable cuando existe una única instancia de ambos procesos. Si más de dos instancias están ejecutándose al mismo tiempo, esta solución puede producir estados no deseados en dichas instancias.

Para representar el comienzo de un protocolo en los procesos correspondientes, en primer lugar se adiciona una operación "de comienzo" en cada tipo de puerto de los roles del protocolo. El nombre de dicha operación es "start_" seguido del nombre del protocolo. Luego, antes de la transformación se debe indicar explícitamente el rol iniciador del protocolo. En el proceso BPEL de dicho rol se adiciona una actividad *receive* con la operación "de comienzo" definida en el tipo de puerto del rol. De esta manera, cuando el propietario del proceso debe iniciar una instancia del proceso debido a un evento ocurrido en su lógica interna, esta operación será invocada desde alguna de sus aplicaciones internas. Además se adiciona una actividad *invoke* con una operación "de comienzo" definida en el tipo de puerto del otro rol, la cual permitirá instanciar el proceso del otro rol. En el caso que el rol no sea el iniciador del protocolo, entonces en su proceso BPEL sólo se adiciona una actividad *receive* con la operación "de comienzo" correspondiente.

Como ejemplo, a continuación se muestra el código que representa el inicio del proceso BPEL del cliente en el protocolo *Blanket Purchase Order*, en donde dicho rol es el iniciador.

```
<receive partnerLink="LinkToSupplier"
       portType="PT:Customer_InterfaceWithSupplier"
       operation="start_BlanketPurchaseOrder"
              inputVariable="ProtocolInstanceNumber"
              CreateInstance="Yes"/>
   <invoke partnerLink="LinkToSupplier"
       portType="PT:Supplier_InterfaceWithCustomer"
       operation="start_BlanketPurcahseOrder"
              inputVariable="ProtocolInstanceNumber"/>
```

El código que representa el comienzo del proceso BPEL del proveedor es el siguiente:

```
<receive partnerLink="LinkToCustomer"
     portType="Supplier_InterfaceWithCustomer"
     operation="start_BlanketPurchaseOrder"
            CreateInstance="Yes"/>
```

7.2.2.5. Transformación de Eventos de Terminación

La terminación de un protocolo, ya sea implícita o explícita, debe ser representada a través de una terminación sincronizada de los procesos BPEL de los roles del protocolo, de manera similar a como se sincroniza el inicio de estos procesos. Para ello, en primer lugar se adiciona una operación "de fin" en cada tipo de puerto de los roles del protocolo. El nombre de dicha operación es "end_" seguido del nombre del protocolo.

Luego, en todo proceso abstracto se adiciona un manejador de eventos, en donde se define que uno de los eventos que puede ocurrir en cualquier punto del proceso es la recepción de un mensaje a través de la operación "de fin". La actividad asociada a dicho evento es la actividad *terminate*, la cual expresa la terminación de la instancia del proceso.

Cuando el protocolo no tiene una terminación explícita, y el proceso BPEL corresponde al rol iniciador, se asume que este rol es quien finaliza el proceso. Esto se realiza adicionando una actividad *invoke* con la operación "de fin" del tipo de puerto del otro rol, seguida de una actividad *terminate*. La primera actividad es utilizada para que el proceso del otro rol sea finalizado. La segunda es utilizada para finalizar la instancia del proceso del rol propietario del mismo. De lo contrario, si el rol del proceso no es el iniciador en el protocolo, no se adiciona actividad alguna, ya que el manejador de evento será el encargado de finalizar la instancia del proceso cuando se reciba el mensaje correspondiente.

Para el caso de un evento de terminación definido en un protocolo, si éste está definido en la *lifeline* del rol propietario del proceso, se procede de la misma manera

como si éste fuera el iniciador del proceso. De lo contrario, no se adiciona otro código más que el incluido en el manejador de eventos. No obstante, BPEL no permite discriminar entre una terminación exitosa de un proceso (como consecuencia de la ejecución de ciertos caminos) o de una terminación sin éxito. Por lo cual, tanto un evento *Success* como un *Failure* de un protocolo es transformado de la misma manera.

Como ejemplo, el siguiente código representa la terminación del proceso BPEL del cliente que representa al protocolo *Blanket Purchase Order*, en donde dicho rol es el iniciador.

```
<invoke partnerLink="LinkToSupplier"
        portType="PT:Supplier_InterfaceWithCustomer"
        operation="terminate_BlanketPurcahseOrder"/>
<terminate/>
```

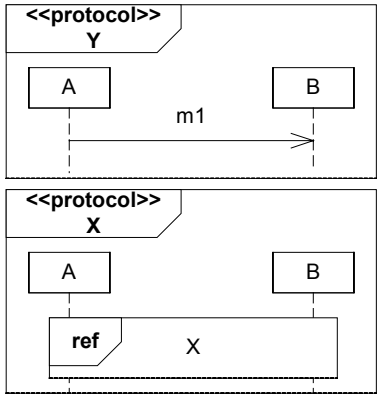
El siguiente código representa la terminación del proceso BPEL del proveedor.

```
<eventHandlers>
   <onMessage partnerLink="LinkToCustomer"
              portType="Supplier_InterfaceWithCustomer"
              operation="end_BlanketPurchaseOrder">
      <terminate/>
   </onMessage>
   ...
</eventHnadlers>
```

7.2.2.6. Transformación de Referencias a Protocolos Anidados

El lenguaje BPEL no soporta la definición de procesos anidados. No obstante es posible representar la invocación de un proceso dentro de otro, a través de la invocación de las operaciones “de comienzo” que representen el comienzo de los procesos BPEL correspondientes al protocolo anidado.

Para expresar esta transformación, utilizamos un ejemplo sencillo. Dos roles, *A* y *B*, están involucrados en un protocolo *X* que tiene como referencia a otro protocolo *Y*. En el protocolo *Y*, el rol *A* es el iniciador. En el proceso BPEL *X_A*, es decir el proceso del rol *A* que representa su parte en el protocolo *X*, se adiciona una actividad *invoke* con la operación *start_Y* sobre el tipo de puerto del rol A. Esto indica que se iniciará una instancia del proceso *Y_A*, es decir del proceso del rol *A* que representa su parte en el protocolo *Y*.




```
<process="X_A"
   <invoke ...
         portType="B"
         operation="start_Y"/>
   <receive ...
         portType="A"
         operation="terminate_Y">
</process>

<process="X_B"
   <receive ...
         portType="B"
         operation="start_Y"/>
   <receive ...
         portType="A"
         operation="terminate_Y">
</process>
```

En el proceso *X_B*, es decir en el caso que el rol no sea el iniciador del protocolo, también es necesario expresar que se inicia dicho proceso. Para ello se adiciona una actividad *receive* con la operación *start_Y* sobre el tipo de puerto de *B*, pero sin el atributo *createInstance*.

No sólo es necesario indicar en los procesos padres (*X_A* y *X_B*) el inicio de los procesos que corresponden al protocolo anidado, sino también su terminación. En el proceso *X_A*, luego de la actividad *invoke* generada, se adiciona una actividad *receive* con la operación *terminate_Y* sobre el tipo de puerto de *A*. En el proceso *X_B*, también se adiciona una actividad *receive* con la operación *terminate_Y* sobre el tipo de puerto de *A*. Estas dos actividades en ambos procesos permiten representar el bloqueo de los mismos hasta que las instancias de los subprocesos sean finalizadas. Estas dos actividades serán realizadas ya que cuando se finalicen los procesos *Y_A* y *Y_B*, se invocarán las operaciones de las mismas.

7.2.2.7. Transformación de Segmentos de Control con el Operador *Loop*.

Un segmento de control con el operador *loop* puede ser transformado en forma directa en una actividad estructurada *while*. BPEL no distingue entre iteraciones de tipo "for" o de tipo "while", sólo incluye esta última. Por lo tanto, todo segmento con el operador *loop*, independientemente de si la condición asociada al camino del segmento es "(0,n)" o "(1,n)" es transformado en un actividad *while*. La condición asociada al camino del segmento es asignada al atributo *condition* de la actividad *while*. Debido a que en UP-ColBPIP, la condición de un camino de un segmento puede ser definida usando OCL o lenguaje natural, para representar a la misma en BPEL se crea una variable, la cual contendrá el valor final de la condición.

Para el protocolo de ejemplo del caso de estudio, el segmento *BPO Negotiation* es transformado en el siguiente código correspondiente al proceso del cliente. En este caso se muestra la variable creada para representar la condición.

```
<while condition="bpws:getVariableData('NegotiationIsRequired')="TRUE">
    ....
</while
```

7.2.2.8. Transformación de Segmentos de Control con el Operador *Xor*

Un segmento de control *Xor* es transformado en una actividad *switch*, en donde cada camino alternativo del segmento es representado por un elemento *case* de dicha actividad. Para cada elemento *case* que representa un camino se adiciona una actividad *sequence*, dentro de la cual se adicionarán luego las actividades que representen los elementos del camino.

La condición de cada elemento *case* debe representar la condición del camino alternativo. Para ello, se adiciona una variable que representa dicha condición. Según el valor de las variables, se invocará la actividad contenida en el elemento *case* correspondiente. El valor de estas variables debe ser obtenido de la ejecución de actividades que representen la lógica interna del rol. Estas actividades deben ser adicionadas luego por el socio cuando derive, a partir del proceso abstracto, el proceso ejecutable.

Este tipo de segmento también podría ser transformado utilizando la actividad *pick*. No obstante, ésta sólo es aplicable cuando el primer elemento de todos los segmentos es un mensaje de negocio. Además, sólo puede ser definida en el proceso BPEL del rol que recibe los mensajes. En el proceso BPEL del otro rol se debe adicionar una actividad *switch* como se explicó anteriormente.

Para el protocolo *BlanketPurchaseOrder* del caso de estudio, el siguiente código representa el segmento *BPO Changes* en el proceso BPEL del cliente.

```
<switch>
   <case condition="bpws:getVariableData('ChangeRequired')=TRUE"
      <sequence>
         ...
      </sequence>
   </case>
   <case condition="bpws:getVariableData('ChangeNotRequired')=TRUE"
      <sequence>
         ...
      </sequence>
   </case>
</switch>
```

7.2.2.9. Transformación de Segmentos de Control con el Operador *And*

Un segmento de control con el operador *And* puede ser transformado en una actividad estructurada *flow* en los procesos BPEL de los roles. Para cada camino del segmento se debe generar una actividad *sequence* dentro de la actividad *flow*. Dentro de cada actividad *sequence* se adicionarán luego las actividades que representen los elementos del camino correspondiente.

7.2.2.10. Transformación de Segmentos de Control con el Operador *Or*

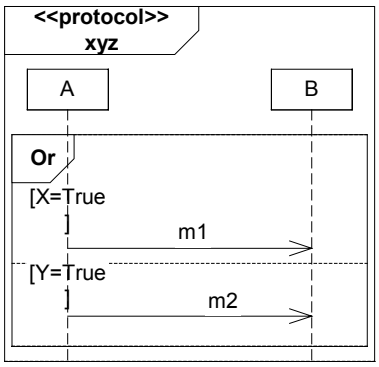
Un segmento de control con el operador *Or* no puede ser transformado en forma directa en BPEL, ya que éste no posee constructores que den soporte a la semántica de este tipo de segmento. No obstante, el mismo puede ser representado a través de una actividad estructurada *flow* en los procesos BPEL de los roles. Una actividad *flow* permite representar actividades concurrentes y también dependencias entre actividades, a través del uso de vinculos (*links*). Con un *link* es posible representar la transición entre una actividad de origen y otra de destino, y adicionarle una condición a la transición.

Para representar el segmento *Or* se define en primer lugar la actividad *flow*. Dentro de la misma se definen varios *links*, uno por cada camino del segmento. Además, se adiciona una actividad *empty* y luego una actividad *sequence* por cada camino del segmento. En la actividad *empty* se adiciona un elemento *source* por cada camino del segmento. En dicho elemento se indica el *link* correspondiente y se adiciona la condición de transición, a partir de la condición definida en el camino del segmento. Luego, en cada actividad *sequence* se adiciona un elemento *target* con el *link* correspondiente al camino representado por dicha actividad. De esta manera se expresa el comienzo de cada camino. Es decir, que cuando una condición definida en la transición de un elemento *source* de la actividad *empty* sea evaluada y se cumpla, entonces comenzará la actividad *sequence* que tiene en el elemento *target* el mismo *link* que el elemento *source* cuya condición fue evaluada.

Para este tipo de segmento, también es necesario expresar la terminación de los caminos del segmento. Para ello se adiciona otra actividad *empty* al final de la actividad *flow*. Además se adicionan otros *links* por cada camino. Estos son definidos como el origen de la transición en las actividades *sequence* de los caminos. En la actividad *empty* se define un elemento *target* por cada camino, con el *link* correspondiente. En

dicha actividad se adiciona el atributo *joinCondition*. En éste se adiciona una expresión condicional que retorna verdadero cuando una de las actividades *sequence* fue completada. Esta condición es construida con la función *bpws:getLinkStatus*, la cual establece si el estado de un *link* entrante en una actividad es positivo (se ejecutó dicha transición) o es negativo (no se ejecutó dicho transición).

Como ejemplo, a continuación se muestra un protocolo con un segmento *Or* y el correspondiente código generado para el proceso BPEL del rol *A*. Para el rol *B* el procedimiento es similar.




```
<process name="xyz_A"
<flow>
   <link name="Start_Path1"/>
   <link name="Start_Path2"/>
   <link name="End_Path1"/>
   <link name="End_Path2"/>
   <empty>
      <source linkName="Start_Path1"
         transitionCondition=
         "bpws:getVariableData('X')=TRUE"/>
      <source linkName="Start_Path2"
         transitionCondition=
         "bpws:getVariableData('Y')=TRUE"/>
   </empty>
   <sequence>
      <target linkName="Start_Path1"/>
      <invoke ....
         operation="m1"/>
      <source linkName="End_Path1"/>
   </sequence>
   <sequence>
      <target linkName="Path2"/>
      <invoke ....
         operation="m2"/>
      <source linkName="End_Path2"/>
   </sequence>
   <empty
      joinCondition="bpws:getLinkStatus('End_Path1') or
                     bpws:getLinkStatus('End_Path2')">
      <target linkName="End_Path1"/>
      <target linkName="End_Path2"/>
   </empty>
</flow>
</process>
```

7.2.2.11. Transformación de Segmentos de Control con el Operador *If*

Un segmento de control con el operador *If* puede ser transformado de manera similar a un segmento de control con el operador *Xor*. La única diferencia consiste en que si el segmento posee el camino con la condición “Else”, entonces en el elemento *otherwhise* se adiciona una actividad *sequence*, en la cual luego se crearán las actividades que correspondan a dicho camino.

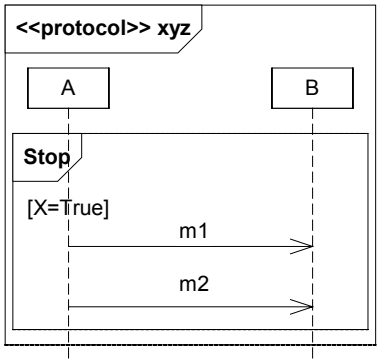
Para el protocolo *Blanket Purchase Order* del caso de estudio, el siguiente código representa al segmento *BPO Changes by Customer* en el proceso BPEL del cliente.

```
<swith>
   <case condition="bpws:getVariableData('ChangeRequired')=TRUE">
      <sequence>
           ....
      </sequence>
   </case>
   <otherwise>
      <sequence>
           ....
      </sequence>
   </otherwise>
</swith>
```

7.2.2.12. Transformación de Segmentos de Control con el Operador *Stop* o *Exception*

Estos tipos de segmentos pueden ser transformados de manera similar como se ha indicado en la transformación de segmentos con el operador *If*. Los mismos sólo contienen un camino y por lo tanto, la actividad *switch* tendrá sólo un elemento *case* con la condición correspondiente. En el caso del segmento con el operador *stop*, primero se debe adicionar una actividad *sequence* y dentro de esta actividad se debe adicionar la actividad *switch* seguida de las actividades que representan la finalización del proceso, como se ha descrito en la sección 7.2.2.5. Esto se debe a que luego de la ejecución del segmento con el operador *stop*, la instancia del protocolo debe finalizar.

Como ejemplo, a continuación se muestra un protocolo con un segmento *stop* y el código correspondiente para el proceso BPEL del rol *A*, el cual es el encargado de sincronizar la terminación de los procesos.




```
<process="xyz_A"
   ...
<segment>
   <switch>
      <case condition=
         "bpws:getVariableData('X')=TRUE"/>
         <sequence>
            ...
         </sequence>
      </case>
   </switch>
   <invoke ...
      portType="B"
      operation="terminate_xyz"/>
   <terminate/>
</segment>
   ...
</process>
```

7.2.2.13. Transformación de Segmentos de Control con el Operador *Transaction*

Este tipo de segmento no puede ser representado en BPEL, debido a que cada proceso es gestionado por un único socio. Dentro del mismo, BPEL sí permite un manejo transaccional de ciertas partes del proceso, utilizando los conceptos de manejadores de fallas y manejadores de compensación. Pero este manejo transaccional es puramente local y ocurre dentro de la instancia de un único proceso. No existe una coordinación distribuida respecto a transacciones entre servicios de múltiples participantes. Esto es reconocido en BPEL y para dar soporte al mismo se propone la utilización de protocolos descritos con el lenguaje *WS-Transaction* (BEA y otros, 2003). No obstante, aún no está claro cómo es posible combinar las definiciones de procesos BPEL con dicho lenguaje.

El manejo transaccional entre instancias de diferentes procesos BPEL podría ser realizado creando una actividad *scope* en cada uno de los procesos y las actividades que representen la sincronización del inicio y fin de la transacción. No obstante, esto requiere de un considerable esfuerzo adicional.

7.2.2.13. Transformación de Segmentos de Control con el Operador *Cancel*

Un segmento de control con el operador *cancel* tampoco tiene una contraparte en BPEL. Un camino de este tipo de segmento con la condición *TimeException* es representado en parte, a través de los eventos *onAlarm* que se adicionan a la actividad *pick*. Los otros caminos de este tipo de segmento no pueden ser representados en BPEL.

7.2.2.14. Sincronización de los Procesos BPEL para Representar Segmentos de Control

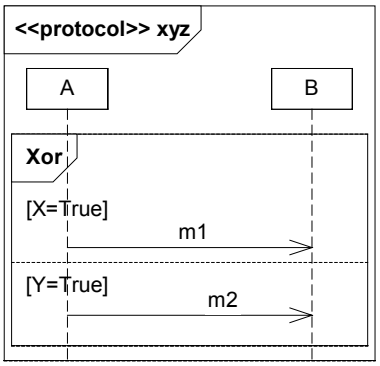
Cuando se transforman segmentos de control en BPEL, se adicionan variables que representan las condiciones que cada camino del segmento posee. El valor de estas variables es utilizado para determinar las actividades a seguir en los procesos BPEL correspondientes a los roles del protocolo. En cada uno de estos procesos se adicionan las mismas variables, cuyo valor debe ser igual en las instancias de los procesos BPEL que representan la instancia de un protocolo. De lo contrario, no existe una sincronización entre las instancias de dichos procesos.

El problema surge debido a que sólo uno de los roles del protocolo (de acuerdo a sus decisiones internas) es quien asignará un valor a dichas variables. Se denomina a este rol, rol sincronizador, y al otro rol, rol sincronizado. En el proceso BPEL del sincronizador, el valor de las variables será obtenido a través de actividades internas, las cuales serán adicionadas cuando se derive de dicho proceso abstracto el proceso ejecutable BPEL. Pero el valor de estas variables debe ser comunicado al proceso BPEL del sincronizado, para mantener sincronizadas las instancias de los procesos y asegurar que ambos seguirán los mismos caminos.

Para dar soporte a esta sincronización se debe realizar lo siguiente. Por cada variable utilizada en una condición se debe adicionar una operación en el tipo de puerto del rol sincronizado. Antes de la actividad que representa al segmento, en el proceso BPEL del sincronizador se debe adicionar una actividad *invoke* por cada variable utilizada en las condiciones de los segmentos. Dicha actividad debe invocar la operación correspondiente creada para dicha variable en el tipo de puerto del sincronizado.

En el proceso BPEL del sincronizado, antes de la actividad que representa al segmento, se debe adicionar una actividad *receive* por cada variable, con la operación correspondiente a la variable.

Como ejemplo, se muestra el código de los procesos BPEL de los roles involucrados en un protocolo *A*. El rol *A*, basado en su lógica interna es quien asigna valor a la variable *X*. El rol *B*, basado en su lógica interna es quien asigna valor a la variable *Y*.




```
<process="xyz_A"
   ...
<invoke ...
      portType="B"
      operation="var_X"
inputVariable="X"/>
<receive ...
      portType="A"
      operation="var_Y" Variable="Y"/>
<switch>
   <case condition=
      "bpws:getVariableData('X')=TRUE"/>
      <sequence>
            ...
      </sequence>
   </case>
   <case condition=
      "bpws:getVariableData(Y')=TRUE"/>
      <sequence>
            ...
      </sequence>
   </case>
</switch>
   ...
</process>
```

```
<process="xyz_B"
   ...
<receive ...
      portType="B"
      operation="var_X" Variable="X"/>
<invoke ...
      portType="A"
      operation="var_Y" inputVariable="Y"/>
<switch>
   <case condition=
      "bpws:getVariableData('X')=TRUE"/>
      <sequence>
            ...
      </sequence>
   </case>
   <case condition=
      "bpws:getVariableData(Y')=TRUE"/>
      <sequence>
            ...
      </sequence>
   </case>
</switch>
   ...
</process>
```

De esta manera, a través de la sincronización de los valores de estas variables, se realiza la sincronización de los estados de los procesos BPEL que representan a un protocolo.

7.3. Conclusiones

A través de la composición de servicios web (o procesos de negocio basados en servicios web) es posible generar soluciones tecnológicas que den soporte a la ejecución de procesos colaborativos. Por lo tanto, debido al interés en este tipo de tecnología, en este capítulo se ha descrito cómo, a través del método de desarrollo conducido por modelos de procesos colaborativos, es posible generar una solución tecnológica basada en composición de servicios web a partir de modelos independientes de la tecnología basados en UP-ColBPIP. En particular, se ha descrito la generación de especificaciones de procesos basados en servicios web con el lenguaje BPEL.

La principal ventaja de dicho método es la garantía de que la solución tecnológica

basada en BPEL es consistente y se corresponde con lo definido en el nivel de negocio a través de modelos basados en UP-ColBPIP. Además, a través del mismo es posible disminuir la complejidad, los tiempos y costos en la generación de dicha solución tecnológica.

El lenguaje BPEL permite definir procesos colaborativos sólo desde el punto de vista de uno de los socios, a través de la especificación de procesos abstractos. Estos procesos definen el comportamiento del rol que desempeña un socio en un proceso colaborativo. Por lo tanto, la semántica de un proceso colaborativo debe ser representada en el nivel tecnológico con BPEL a través de varios procesos abstractos BPEL, uno por cada socio involucrado en dicho proceso.

Lo anterior adiciona otra complejidad a este tipo de solución tecnológica de procesos colaborativos, ya que para poder representar la semántica completa de cada uno de estos procesos, es necesario que los procesos abstractos BPEL correspondientes sean consistentes. Dicha consistencia se refiere a que los procesos abstractos de cada socio deben ser definidos de acuerdo a lo establecido en el proceso colaborativo. Si tal consistencia no puede ser alcanzada, no es posible garantizar que un conjunto de procesos abstractos BPEL representen la semántica de un proceso colaborativo. Este requerimiento de consistencia entre procesos abstractos BPEL puede ser alcanzado y garantizado a través del método propuesto para el desarrollo conducido por modelos de procesos colaborativos. Esto se debe a que los procesos abstractos que representan un proceso colaborativo son generados a partir de un único modelo conceptual de dicho proceso y utilizando el mismo modelo de transformación.

Los modelos UP-ColBPIP, los cuales pueden contener un conjunto de procesos colaborativos, no tienen una relación uno a uno con los modelos BPEL, sino una relación uno a muhcos. Para un modelo UP-ColBPIP, varios modelos BPEL deben ser generados y por lo tanto, la transformación de modelos UP-ColBPIP a BPEL debe ser ejecutada varias veces. Esto se debe a dos razones: debido a que en un modelo BPEL sólo es posible definir un proceso abstracto; y debido a que para un proceso colaborativo basado en un protocolo de interacción, se debe generar un proceso abstracto por cada socio involucrado en el proceso.

En este capítulo se han definido a nivel conceptual las transformaciones de modelos UP-ColBPIP a especificaciones en BPEL y WSDL. Las correspondencias entre los

conceptos de dichos lenguajes han sido señaladas y definidas. Además, se ha definido cómo cada concepto de UP-ColBPIP es transformado en BPEL y WSDL. A partir de estas definiciones, luego es posible definir un modelo de transformación que soporte la ejecución automática de dichas transformaciones de modelos y permita la generación de las especificaciones en BPEL y WSDL.

Como resultado de las transformaciones definidas, es posible inferir que la mayoría de los conceptos provistos por UP-ColBPIP para definir procesos colaborativos basados en protocolos de interacción pueden ser representados en BPEL. No obstante, la representación de muchos de ellos no es directa y se requiere un esfuerzo adicional para representarlos.

Por otra parte, BPEL no provee soporte para sincronizar dos procesos abstractos. Por lo tanto, se debe adicionar código extra, el cual no pertenece a la lógica del proceso, pero es necesario para representar la sincronización de las instancias de los procesos que representan la instancia de un protocolo.

En resumen, en este capítulo se ha descrito cómo a partir de una solución en el nivel de negocio utilizando el lenguaje de modelo UP-ColPIP, es posible generar una solución tecnológica basada en BPEL, en el marco de un proceso de desarrollo conducido por modelos de procesos colaborativos.

Capítulo

8

CONCLUSIONES Y TRABAJOS FUTUROS

Este capítulo tiene como objetivo destacar las principales contribuciones de esta tesis, y describir los aspectos de la misma que constituyen el punto de partida para el desarrollo de trabajos futuros.

8.1. Principales Contribuciones

A través del trabajo desarrollado en esta tesis, se ha contribuido a diferentes áreas de interés, las cuales van desde la gestión de negocios, a la ingeniería de sistemas de información e ingeniería de software. En particular se han realizado contribuciones en los siguientes dominios: modelos de colaboración para la gestión de la cadena de suministro, colaboraciones B2B, modelado y especificación de procesos de negocio, sistemas de información B2B, métodos de desarrollo conducido por modelos, y transformaciones de modelos y generación automática de código.

Este trabajo de investigación comenzó desde la observación que, la colaboración entre empresas es el principal factor de éxito para que las mismas puedan disminuir los costos y mejorar los beneficios en la gestión de la cadena de suministro. La colaboración entre empresas implica que las mismas son integradas a través de la definición y ejecución en forma conjunta de procesos de negocio colaborativos.

A partir de este punto se ha identificado que para dar soporte a dicha colaboración se deben aplicar modelos descentralizados. Para ello se han estudiado diferentes modelos de colaboración y se ha propuesto el modelo de colaboración socio-a-socio, el cual surge con el objetivo de gestionar en forma descentralizada relaciones de colaboración entre empresas de producción. Para dar soporte a estos modelos se ha concluido que las empresas deben establecer colaboraciones B2B basadas en sistemas de información peer-to-peer. De esta manera, las empresas podrán alcanzar altos niveles de integración, manteniendo sus autonomías y gestionando en forma descentralizada los procesos colaborativos.

Actualmente, el desarrollo de colaboraciones B2B es complejo y demanda demasiado tiempo, lo cual está limitando su aplicación por los altos costos. El principal objetivo de esta tesis ha sido proveer un conjunto de métodos, lenguajes de modelado, técnicas y herramientas que permiten salvar estos inconvenientes, de manera tal que las empresas cuenten con los instrumentos adecuados y necesarios para establecer colaboraciones B2B.

En primer lugar se ha destacado la importancia de considerar dos niveles de abstracción en el desarrollo de relaciones de colaboraciones B2B: el nivel de negocio y el nivel tecnológico. El nivel de negocio (el dominio del problema) refiere al diseño conceptual de los procesos colaborativos y las interfaces de negocio de los socios, sin considerar la tecnología de implementación. El nivel tecnológico (el dominio de la solución) refiere al diseño de protocolos B2B, los cuales implican la especificación de: los procesos colaborativos en un lenguaje ejecutable de procesos de negocio, y las interfaces de los componentes de los socios que conforman el sistema de información B2B. A través de estas especificaciones en el nivel tecnológico es posible dar soporte a la ejecución de los procesos colaborativos. Actualmente, dos tipos de tecnologías están disponibles para definir estos protocolos B2B: estándares basados en transacciones de negocio (tal como ebXML) y estándares basados en composición de servicios web (tales como BPEL y WSCI). Las empresas primero deben enfocarse en el nivel de negocio y luego en el nivel tecnológico.

La generación automática de las especificaciones de los protocolos B2B basados en un estándar particular, a partir de los modelos de procesos colaborativos independientes de la tecnología, es de suma importancia para disminuir la complejidad, el tiempo y los costos en la construcción de colaboraciones B2B. Para dar soporte a esto y a las diferentes etapas del desarrollo de procesos colaborativos en ambos niveles, la filosofía propuesta por el desarrollo conducido por modelos ha sido identificada como un habilitador clave. Por lo tanto, se ha propuesto un método basado en el desarrollo conducido por modelos para procesos colaborativos.

A continuación se discuten las restantes contribuciones de esta tesis teniendo en cuenta dicho método y las principales técnicas, lenguajes y herramientas propuestas que forman parte del mismo.

8.1.1. Método de Desarrollo Conducido por Modelos para Procesos Colaborativos

Como se ha discutido en el capítulo 3, aunque los beneficios del desarrollo conducido por modelos en el dominio de procesos colaborativos son claros, actualmente el mismo no ha sido completamente explotado en este dominio. A partir de lo discutido en los trabajos relacionados, si bien existen algunas aproximaciones, se ha concluido que ninguno de estos trabajos está basado en un método de desarrollo conducido por modelos bien definido, que provea las técnicas y métodos necesarios para definir cada uno de los componentes que se requieren en este tipo de métodos.

El método propuesto está basado en los principios, componentes y estándares propuestos por la Arquitectura Conducida por Modelos (MDA). Por lo tanto, se ha propuesto una completa aplicación de MDA para el dominio de procesos colaborativos y sistemas de información B2B. Además, se provee un método robusto a través de la utilización de estándares extensamente aceptados tanto en la academia como en la industria. Siguiendo los lineamientos de este método, las empresas podrán generar soluciones que les permitan construir colaboraciones B2B, orientadas a la gestión integrada de la cadena de suministro.

Este método provee los lineamientos y etapas ha seguir para generar en forma automática las especificaciones de los protocolos B2B basados en un estándar particular, a partir de los modelos de procesos colaborativos independientes de la tecnología. Los principales beneficios de este método para el dominio de procesos colaborativos y sistemas de información B2B son:

- Incremento del nivel de abstracción en el desarrollo de colaboraciones B2B, debido a que los principales productos del desarrollo son los modelos de procesos colaborativos independientes de la tecnología.
- Disminución del tiempo de desarrollo, ya que la generación de la solución es realizada en forma automática a partir de los modelos conceptuales de procesos colaborativos.
- Disminución de los costos como consecuencia de la automatización en la generación de especificaciones B2B.

- Garantía de que la solución tecnológica alcanzada se corresponde con lo definido en el nivel de negocio.
- Independencia de los modelos de procesos colaborativos de las nuevas o diferentes tecnologías, incrementando la reutilización de los mismos para generar diferentes soluciones tecnológicas.
- Soporte al problema de interoperabilidad en tiempo de diseño, como consecuencia de que una empresa puede implementar procesos colaborativos con diferentes socios usando diferentes tecnologías.
- Promueve la idea de que la solución tecnológica utilizada para implementar los procesos colaborativos debe ser decidida posteriormente.

El desarrollo conducido por modelos a través de MDA es sólo una filosofía para llevar a cabo procesos de desarrollo. No obstante, para hacer efectivo los beneficios de esta filosofía se requiere de un conjunto de lenguajes de modelado, y técnicas de transformación de modelos y de generación de código. En esta tesis se han provistos los lenguajes y técnicas que dan soporte al desarrollo de procesos colaborativos siguiendo una arquitectura conducida por modelos. A continuación se describen cada uno de ellos.

8.1.2. UP-ColBPIP: Lenguaje de Modelado de Procesos de Negocio Colaborativos basados en Protocolos de Interacción

El diseño de procesos colaborativos en el nivel de negocio es una tarea clave en el desarrollo de colaboraciones B2B, principalmente porque la forma en que los socios colaborarán es formalizada en dichos procesos. Como se ha discutido en el capítulo 3, no existe un lenguaje de modelado que soporte el requerimiento de modelar procesos colaborativos independientes de la tecnología representando: la vista global de las interacciones peer-to-peer, y las responsabilidades de los roles que desempeñan los socios en los procesos.

Por lo tanto, se ha propuesto el Lenguaje de Modelado de Procesos de Negocio Colaborativos basados en Protocolos de Interacción (UP-ColBPIP). Este lenguaje da soporte al modelado visual de procesos colaborativos independiente de la tecnología.

El lenguaje UP-ColBPIP soporta la definición de cuatro vistas de los procesos colaborativos. Cada vista ofrece una perspectiva diferente, como así también resalta

propiedades diferentes de los procesos. A través de la definición de estas vistas, da soporte a las etapas de análisis y diseño de los procesos colaborativos, como así también a la derivación a partir de los mismos, de las interfaces de los socios en el nivel de negocio. Con UP-ColBPIP, el modelado conceptual de los procesos colaborativos sigue un proceso de diseño top-down, en donde cada vista es un refinamiento de la anterior.

La base teórica de UP-ColBPIP está sustentada por el uso del formalismo de protocolos de interacción para diseñar procesos colaborativos, junto con la aplicación de los principios de la teoría de actos de comunicación y la teoría LAP (Language/Action Perspective). De esta manera, puede ser considerado como un tipo de lenguaje de procesos colaborativos orientado a la comunicación. A través de la definición de protocolos de interacción, los diseñadores se enfocan en modelar no sólo el intercambio de información, sino también los aspectos de comunicación requeridos en los procesos colaborativos.

El uso de actos de comunicación para describir los mensajes o interacciones entre las partes posibilita:

- expresar las intenciones de las partes cuando intercambian información en un proceso colaborativo, a través de las cuales las partes crean, modifican, cancelan o cumplimentan los compromisos.
- simplificar el diseño de los procesos colaborativos, ya que los diseñadores pueden utilizar conceptos intuitivos más cercanos al lenguaje natural, pudiendo hacer analogías con las interacciones sociales entre humanos.

Además, el uso de protocolos de interacción posibilita satisfacer los diferentes requerimientos que se han identificado con respecto al modelado conceptual de procesos colaborativos en relaciones de colaboraciones B2B: vista global de las interacciones entre los socios, autonomía de las empresas, descentralización e interacciones peer-to-peer, procesos colaborativos como procesos abstractos, derivación de las interfaces de los socios a partir de los procesos, soporte a las perspectivas de los modelos de procesos y soporte a procesos con negociaciones. Principalmente, a través de la definición de protocolos de interacción los diseñadores se enfocan en describir la coreografía global de las interacciones entre los socios, sin la necesidad de recurrir a otra vista para entender las responsabilidades de los roles en el proceso. Esto y la

posibilidad de expresar intenciones a través de los actos de comunicación, permiten la representación de negociaciones complejas dentro de los procesos colaborativos.

El lenguaje UP-ColBPIP ha sido definido como un perfil UML, basado en UML2, siguiendo los lineamientos propuestos por MDA para diseñar lenguajes independientes de las plataformas. El lenguaje reutiliza la mayor parte de la notación de UML2, con el propósito de ofrecer a los analistas de negocio y desarrolladores de sistemas una notación intuitiva y extensamente conocida, para modelar procesos colaborativos. Esto permite a los usuarios del lenguaje acelerar el tiempo de aprendizaje del mismo. Además, el uso de un perfil UML para modelar procesos colaborativos permite: proveer un vocabulario más adecuado que el original de UML para modelar dichos procesos; adicionar semántica y restricciones al metamodelo de UML para diseñar procesos colaborativos; y reutilizar las herramientas Case UML existentes, sin la necesidad de contar con una herramienta específica para ello. Otra ventaja de definir al lenguaje UP-ColBPIP como un perfil UML, es que el mismo también puede ser extendido y personalizado por los usuarios finales.

Para derivar el perfil UML, se ha definido un metamodelo del lenguaje. Este metamodelo también es adecuado para ser utilizado como una ayuda de referencia para entender la semántica de los elementos conceptuales del mismo, y las restricciones de cada uno de ellos como así también sus relaciones. Además, a partir de este metamodelo es posible la construcción de herramientas específicas que implementen el lenguaje UP-ColBPIP.

Como demostración de conceptos, la aplicación del lenguaje UP-ColBPIP a un caso de estudio también ha sido ilustrada. Por lo tanto, el objetivo de dar soporte al análisis y diseño de procesos colaborativos en el nivel de negocio ha sido cumplimentado.

8.1.3. El Método de Transformación de Modelos

Para hacer realidad transformaciones de modelos en el dominio de procesos colaborativos, se han identificado los requerimientos particulares para tal tipo de transformaciones y se ha desarrollado un método que satisface a los mismos. El objetivo de este método es dar soporte al proceso completo requerido para realizar el desarrollo conducido por modelos de procesos colaborativos. Como parte del método se ha

propuesto: el *Lenguaje de Transformación de Modelos*, el *Lenguaje de Modelado de Reglas* y las *Reglas de Producción de Código XML*.

El lenguaje de transformación de modelos da soporte a la composición, ordenamiento y reutilización de reglas de transformación, y por lo tanto a través del mismo es posible definir modelos de transformación. Permite declarar las reglas con sus patrones, y componer y ordenar a las mismas dentro de módulos de reglas utilizando una notación de árbol. Dicha notación facilita la comprensión de la transformación completa. La principal ventaja de este lenguaje es la separación de la definición de las reglas, de la composición y estructura de control de las mismas. Esto posibilita la reutilización de las reglas en varias partes de la transformación. Por otra parte, la capacidad para componer las reglas simples en una jerarquía de nodos dentro de los módulos de reglas incrementa la legibilidad, modularidad y el mantenimiento de transformaciones complejas. Además, ofrece un mecanismo de ordenamiento explícito.

El lenguaje de modelado de reglas permite definir las reglas con sus patrones utilizando una notación gráfica conforme a UML. Este lenguaje está basado en la teoría de gramáticas y transformación de grafos, lo cual permite que la semántica operacional de una regla pueda ser entendida en forma intuitiva y clara. De esta manera, es posible implementar los patrones siguiendo la semántica operacional que ofrece el mismo.

Los lenguajes y técnicas propuestas por el método de transformación de modelos están acoplados a los principios establecidos en MDA. El lenguaje de transformación de modelos ha sido construido a través de un metamodelo basado en MOF. El lenguaje de modelado de reglas permite definir los patrones de las mismas utilizando diagramas de UML. Para definir las reglas de transformación se requiere la aplicación de la técnica de metamodelación, ya que los patrones son definidos teniendo en cuenta la sintaxis abstracta de los lenguajes de modelado.

Aunque este método de transformación de modelos ha sido diseñado para llevar a cabo transformaciones de modelos en el dominio de procesos colaborativos, el mismo es bastante general y en principio podría ser utilizado para realizar transformaciones en otros dominios.

8.1.4. La Herramienta de Transformación de Modelos

Se ha desarrollado un prototipo de la herramienta que implementa el metamodelo y la semántica del lenguaje de transformación de modelos. Los componentes de la herramienta soportan la definición y ejecución de modelos de transformación utilizando dicho lenguaje.

La herramienta desarrollada es completamente independiente de las herramientas Case UML, como así también de los ambientes de metamodelación. La herramienta permite transformar modelos UP-ColBIP almacenados en un formato XMI. De esta manera, estos modelos a transformar pueden ser construidos con diferentes herramientas Case UML. Por otra parte, se han incorporado a la herramienta las APIs necesarias para la manipulación de los modelos de entrada y salida, las cuales fueron generadas con otro ambiente de metamodelación.

El método y la herramienta de transformación de modelos también podrían ser utilizados para realizar refinamientos de modelos UP-ColBPIP, con el objetivo de derivar una vista de dicho modelo a partir de otra. Esto permite proveer capacidades de automatización a las actividades de diseño conceptual, ya que con el uso de un perfil UML esto no es posible.

La aplicabilidad tanto del método como de la herramienta ha sido probada en las transformaciones de modelos de procesos de UP-ColBPIP a especificaciones de procesos basadas en ebXML BPSS.

8.1.5. Transformaciones de Modelos UP-ColBPIP a ebXML

En esta tesis se ha descrito cómo, a través del método de desarrollo conducido por modelos para procesos colaborativos, es posible generar soluciones tecnológicas basadas en ebXML, a partir de modelos independientes de la tecnología basados en UP-ColBPIP.

La generación de una solución tecnológica en ebXML consiste en generar especificaciones de procesos colaborativos utilizando el lenguaje BPSS y especificaciones de las interfaces de los socios utilizando el lenguaje CPPA. Por lo tanto, se han desarrollado las técnicas y componentes requeridos en el método de desarrollo conducido por modelos de procesos colaborativos: las transformaciones de modelos UP-ColBPIP a BPSS, las transformaciones de modelos UP-ColBPIP a CPPA,

y los metamodelos de BPSS y CPPA que permiten construir modelos basados en dichos lenguajes. La principal ventaja de estas transformaciones es la garantía de que la solución tecnológica generada en ebXML es consistente y se corresponde con lo definido en el nivel de negocio a través de modelos basados en UP-ColBPIP.

Como resultado de las transformaciones definidas, es posible inferir que la mayoría de los elementos que componen una especificación en BPSS pueden ser derivados a partir de modelos UP-ColBPIP. Esto significa que el lenguaje UP-ColBPIP provee los elementos de modelado conceptual requeridos no sólo para diseñar procesos colaborativos en un alto nivel de abstracción sino también para derivar especificaciones ejecutables en un estándar B2B basado en transacciones de negocio, como ebXML. Los únicos elementos que no pueden ser derivados son aquellos que constituyen detalles de bajo nivel que corresponden a los aspectos de configuración necesarios para la ejecución de los procesos.

Con respecto a la transformación de UP-ColBPIP a CPPA definida, la misma posibilita la generación de un esqueleto de la especificación CPP que define la interfaz de cada socio.

8.1.6. Transformaciones de Modelos UP-ColBPIP a BPEL/WSDL

Debido al interés en soluciones tecnológicas basadas en composición de servicios web, en esta tesis se ha descrito cómo, a través del método de desarrollo conducido por modelos de procesos colaborativos, es posible generar una solución tecnológica de este tipo a partir de modelos independientes de la tecnología basados en UP-ColBPIP. En particular, se ha descrito la transformación de modelos UP-ColBPIP a especificaciones de procesos basados en servicios web con los lenguajes BPEL y WSDL. La principal ventaja de estas transformaciones es la garantía de que la solución tecnológica basada en estos lenguajes es consistente y se corresponde con lo definido en el nivel de negocio a través de modelos basados en UP-ColBPIP.

La semántica de un proceso colaborativo debe ser representada en el nivel tecnológico con BPEL a través de varios procesos abstractos BPEL, uno por cada socio involucrado en dicho proceso. Esto se debe a que BPEL sólo permite definir procesos colaborativos desde el punto de vista de uno de los socios, a través de la especificación de procesos abstractos. Por lo tanto, estos procesos abstractos deben ser consistentes

entre sí para representar la semántica de los procesos colaborativos. Este requerimiento de consistencia para este tipo de solución tecnológica puede ser alcanzado y garantizado a través del método propuesto para el desarrollo conducido por modelos de procesos colaborativos. Esto se debe a que los procesos abstractos que representan un proceso colaborativo son generados a partir de un único modelo conceptual de dicho proceso y utilizando las mismas definiciones de transformaciones.

Se ha definido a nivel conceptual las transformaciones de modelos UP-ColBPIP a especificaciones en BPEL y WSDL. Las correspondencias entre los conceptos de dichos lenguajes han sido señaladas y definidas. Además, se ha definido cómo cada concepto de UP-ColBPIP es transformado en BPEL y WSDL. A partir de estas definiciones, luego es posible definir un modelo de transformación que soporte la ejecución automática de dichas transformaciones de modelos y permita la generación de las especificaciones en BPEL y WSDL.

Como resultado de las transformaciones definidas, es posible inferir que la mayoría de los conceptos provistos por UP-ColBPIP para definir procesos colaborativos basados en protocolos de interacción pueden ser representados en BPEL. No obstante, la representación de muchos de ellos no es directa y requiere un esfuerzo adicional para representarlos. Por otra parte, BPEL no provee soporte para sincronizar dos procesos abstractos. Por lo tanto, también se debe adicionar código extra, el cual no pertenece a la lógica del proceso, pero que es necesario para representar la sincronización de las instancias de los procesos que representan la instancia de un protocolo.

8.1.7. Resumen de las Contribuciones de la Tesis

En resumen, las contribuciones realizadas en esta tesis son las siguientes:

- El **método de desarrollo conducido por modelos para el dominio de procesos colaborativos**, el cual define los lineamientos y etapas para generar soluciones tecnológicas, utilizando como principales productos de desarrollo los modelos conceptuales de procesos colaborativos.
- El **lenguaje de modelado UP-ColBPIP**, el cual soporta el análisis y diseño de procesos colaborativos independientes de la tecnología.
- El **método de transformación de modelos**, el cual da soporte a la transformación de modelos UP-ColBPIP a modelos basados en un estándar

B2B, como así también la generación del código XML a partir de estos últimos. Como parte de este método se ha propuesto:

  - El **lenguaje de transformación de modelos**, el cual soporta la definición de transformaciones de modelos.
  - El **lenguaje de modelado de reglas**, el cual soporta la definición de la semántica operacional de las reglas de transformación, que luego son implementadas e integradas a las transformaciones definidas con el lenguaje anterior.
  - Las **Reglas de Producción de Código XML**, las cuales permiten generar código XML a partir de modelos de objetos.

- La **herramienta de transformación de modelos**, la cual soporta la definición, implementación y ejecución de transformaciones de modelos y generación de código XML.
- La **definición de transformaciones de modelos UP-ColBPIP a especificaciones en ebXML**, la cual permite la generación de soluciones tecnológicas basadas en el estándar ebXML.
- La **definición de transformaciones de modelos UP-ColBPIP a especificaciones en BPEL/WSDL**, la cual permite la generación de soluciones tecnológicas basadas en composición de servicios web.

8.2. Trabajos Futuros

8.2.1. Extensiones al Lenguaje UP-ColBPIP

Una de las principales bases teóricas del lenguaje UP-ColBPIP es el uso de actos de comunicación para representar las interacciones entre las partes en los procesos colaborativos definidos a través de protocolos de interacción. A pesar de los beneficios de utilizar actos de comunicación para definir protocolos de interacción, la correcta utilización de los mismos es responsabilidad del diseñador. Esto se refiere a que el diseñador debe tener en cuenta la semántica de cada uno de ellos, según la librería de actos de comunicación utilizada. No obstante, los mismos pueden no ser interpretados de igual manera por diferentes personas, debido a que en algunos casos la semántica está descrita informalmente o sólo se provee un significado individual para cada acto,

sin indicar la relación con los restantes. Esto podría causar que el orden en que los actos de comunicación son utilizados no sea el correcto.

Por lo tanto, sería conveniente incluir un mecanismo que asista al diseñador para definir protocolos de interacción coherentes y bien formados. No desde el punto de vista del flujo de control, sino desde el punto de vista del uso de la teoría de actos de comunicación. Es decir, un mecanismo que indique qué actos de comunicación podrían ser utilizados en cada paso del protocolo, de tal manera de establecer compromisos en forma coherente.

Un posible mecanismo podría ser adicionar y definir compromisos en forma explícita en las diferentes partes de los protocolos de interacción, donde corresponda, para establecer la semántica de uso de los actos de comunicación. Dado que un compromiso puede ser creado, modificado, cancelado, o cumplido (Verdicchio y Colombetti, 2002), éste puede ser visto como un objeto que posee las operaciones de creación, modificación, cancelación y cumplimiento. Luego, para cada una de estas operaciones se debe indicar el o los actos que pueden ser utilizados y están relacionados con cada una de las mismas.

De esta manera, cuando se define un proceso basado en protocolos de interacción, debería indicarse los compromisos que las partes asumen en el mismo, y según las operaciones a ser aplicadas a esos compromisos, el diseñador podrá saber qué tipos de actos de comunicación debería aplicar para llevar a cabo estas operaciones.

8.2.2. Verificación y Validación de Modelos de Procesos Colaborativos

Dado que un proceso colaborativo describe el comportamiento de las partes en la colaboración, como así también la forma en que los sistemas de dichas partes interactuarán, la verificación de requerimientos funcionales de dichos procesos, tales como la ausencia de “deadlocks” o que ciertos estados sean alcanzados, debería ser realizada. Actualmente, existen diferentes propuestas que soportan la verificación de procesos en lenguajes como BPEL (Ferrara, 2002) (Fu y otros, 2004). Por lo tanto, luego de la generación de una solución tecnológica, a partir de la aplicación del método de desarrollo conducido por modelos propuesto, es posible verificar las especificaciones generadas. No obstante, esto tiene como desventaja que si se han encontrado defectos en dichas especificaciones, las mismas deberían ser modificadas en la solución generada.

Otra alternativa sería incluir dichas modificaciones en los modelos UP-ColBPIP y generar nuevamente la solución. Esto tiene la desventaja que quien diseña y corrige los procesos debería manejar ambos niveles de abstracción, es decir conocer la correspondencia entre los conceptos de UP-ColBPIP y los lenguajes de especificación de procesos que se utilicen.

Para salvar los anteriores inconvenientes, sería adecuado dar soporte a la verificación de los procesos, cuando estos son definidos en el nivel de negocio usando el lenguaje UP-ColBPIP, previo a la generación de las soluciones tecnológicas. De esta manera los diseñadores de los procesos podrán modificar y decidir los cambios que consideren necesarios cuando detecten errores en el diseño de los procesos. Para ello, las técnicas de transformación de modelos que han sido propuestas podrían ser utilizadas. En este caso, el objetivo sería transformar modelos de procesos definidos con UP-ColBPIP a modelos basados en algún lenguaje formal, para luego llevar a cabo la verificación de dichos procesos a través de herramientas específicas. Por ejemplo, una definición de protocolo de interacción podría ser traducida a una definición en algún tipo de redes de Petri, y luego utilizar alguna de las herramientas disponibles para llevar a cabo el análisis de la misma.

8.2.3. Integración de Procesos Colaborativos con Procesos Privados

Como se ha discutido en esta tesis, el concepto principal para llevar a cabo la colaboración entre empresas es el de proceso de negocio colaborativo. Dentro de un proceso colaborativo, cada socio cumple un rol específico. La definición de procesos colaborativos a través de protocolos de interacción posibilita definir en forma explícita el comportamiento público de cada rol del proceso, expresado en términos de la secuencia de envío y recepción de mensajes. No obstante, cada socio debe dar soporte internamente a los roles que cumple en los procesos colaborativos, con el objetivo de: decidir qué caminos alternativos seguir en dichos procesos, generar la información a enviar a sus socios, y procesar la información recibida de sus socios. Estas actividades son soportadas dentro de las empresas a través de sus procesos de negocio privados. Debido a que dichos procesos generalmente ya existen, es necesario integrar los mismos con los procesos colaborativos.

Para integrar estos procesos es posible utilizar nuevamente un enfoque de integración basado en procesos. Esto significa que la integración entre un proceso

colaborativo y los proceso privados que soportan al mismo puede ser realizada definiendo otro proceso, que se denomina proceso de interfaz. Este último será el encargado de gestionar el rol que un socio desempeña en un proceso colaborativo, como así también de gestionar la invocación y ejecución de aquellas actividades o procesos internos requeridos para cumplir con lo acordado con los socios. Obviamente, este proceso de interfaz debe ser definido en base al proceso colaborativo a dar soporte.

Las técnicas y métodos propuestos en esta tesis para el desarrollo conducido por modelos de proceso colaborativos, pueden ser utilizados como base para llevar a cabo dicha integración. Los modelos de procesos de interfaz pueden ser definidos usando diagramas de actividades de UML. Por lo tanto, a partir de modelos de protocolos de interacción es posible derivar modelos de procesos de interfaz, utilizando el método y la herramienta de transformación de modelos propuestos. Más específicamente, a partir de las *lifelines* que representan los roles que cada socio cumple en un protocolo de interacción, es posible derivar los correspondientes diagrama de actividades. Luego, cada empresa deberá adicionar a su proceso de interfaz generado, la lógica interna que indiquen los procesos privados a ser ejecutados.

Debido a que un proceso de interfaz es un proceso de negocio, el mismo podrá ser ejecutado por cualquiera de los sistemas de gestión de proceso de negocio existentes. Por lo tanto, a partir de dicho proceso se deberá generar en forma automática la especificación del proceso según el lenguaje soportado por el sistema de gestión de procesos de negocio. Más aún, un proceso de interfaz puede ser visto como un proceso ejecutable de BPEL, el cual consiste de una parte pública y una privada. De esta manera también es posible implementar un proceso de interfaz utilizando BPEL.

De esta manera, un trabajo futuro es extender el método de desarrollo conducido por modelos para procesos colaborativos, con el objetivo de dar soporte a la integración de estos procesos con los procesos privados.

8.2.4. Sistema de Administración de Procesos Colaborativos basado en Agentes de Software

Aunque las empresas pueden implementar en forma independiente su sistema de administración de procesos colaborativos, es necesario determinar también la tecnología de software más adecuada para implementar estos componentes de los sistemas de

información B2B. Para ello, se deberían considerar algunos de los requerimientos apuntados en el capítulo 3 para el modelado de procesos colaborativos. Una tecnología que permite dar soporte a estos requerimientos es la tecnología de agentes de software (Willmott y otros, 2002) (Jennings, 2001).

De esta manera, es necesario contar con herramientas que permitan el desarrollo de sistemas basados en agentes a partir de modelos de procesos colaborativos. En este sentido, el uso del enfoque de procesos colaborativos basados en protocolos de interacción posibilita esta derivación. Esto se debe a que dos conceptos claves en las organizaciones de sistemas multi-agentes son: el concepto de rol y el de protocolo de interacción (Wooldridge y otros, 2000). Por lo tanto, los roles de los agentes pueden ser derivados a partir de los roles identificados en los procesos colaborativos. Luego, los protocolos de interacción que definen a estos procesos deben ser considerados como los protocolos de interacción que los agentes van a ejecutar para comunicarse entre sí. Debido a esto último, los sistemas de información B2B basados en agentes no pueden ser diseñados usando el enfoque tradicional de diseño de sistemas multi-agentes, en el cual los protocolos de interacción son definidos e incorporados en tiempo de diseño. Esto no permite ejecutar nuevos protocolos para que un agente pueda gestionar nuevos procesos colaborativos y comunicarse con otros nuevos agentes. Para ello, los agentes deben poder interpretar y ejecutar procesos colaborativos basados en protocolos de interacción, los cuales están definidos en lenguajes ejecutables de especificación de procesos, como por ejemplo ebXML BPSS.

Por lo tanto, un trabajo futuro es la definición de un modelo de ejecución a ser incorporado en los agentes, para que los mismos interpreten protocolos de interacción en tiempo de ejecución y en forma dinámica.

8.2.5. Extensiones al Método y a la Herramienta de Transformación de Modelos

Actualmente, el lenguaje de modelado de reglas no está soportado por la herramienta de transformación de modelos y por lo tanto, el mismo es declarativo pero no es ejecutable. Por lo tanto, el objetivo final es incorporar un soporte a la herramienta para poder definir los patrones de las reglas en un alto nivel de abstracción utilizando el lenguaje de modelado de reglas, como una alternativa a la implementación de dichos patrones usando el lenguaje de programación Java. Además, esto permitiría resolver el

problema de garantizar que la definición de un patrón es consistente con un metamodelo y está libre de errores, ya que tal definición podría ser chequeada por la herramienta.

En particular, dos soluciones son posibles. Una alternativa es incorporar un soporte gráfico a la herramienta para el lenguaje de modelado de reglas. Ya que el lenguaje de modelado de reglas es también un perfil UML, otra alternativa es definir las reglas usando una herramienta Case UML, exportar las definiciones en un archivo XMI, procesar a las mismas, y generar en forma automática una representación interna de la gramática de grafos de las reglas.

ANEXO

A

LIBRERÍA DE ACTOS DE COMUNICACIÓN DE FIPA ACL

Este anexo documenta los actos de comunicación provistos por el estándar FIPA ACL (The Foundation for Intelligent Physical Agents – Agent Communication Language), los cuales son utilizados en los ejemplos de protocolos de interacción provistos en esta tesis. Para cada acto de comunicación se describe su semántica de uso, la cual expresa el significado o intención de la acción que un rol comunica a otro rol como parte del envío de un mensaje. Además se describe una clasificación de los mismos, según el tipo de interacciones a realizar en un protocolo.

A.1. Semántica de los Actos de Comunicación

Accept-Proposal

Representa la acción de aceptar una propuesta para realizar una acción. La propuesta debe haber sido previamente presentada (generalmente a través de un mensaje con el acto de comunicación *propose*). El rol que envía la aceptación comunica al rol receptor que, en algún momento del futuro, éste entiende que el receptor realizará la acción comprometida en la propuesta que está aceptando.

Agree

Representa la acción de acordar ejecutar alguna acción, posiblemente en el futuro. Esta acción expresa un acuerdo a una solicitud previamente presentada para ejecutar una acción. El rol que envía el acuerdo comunica al rol receptor que éste tiene la intención de ejecutar la acción solicitada previamente.

Cancel

Representa la acción de un rol *i* informando a otro rol *j*, que *i* no tiene más interés en que *j* realice alguna acción. Esto no representa que *i* desea que *j* no ejecute la acción o pare de realizarla. *Cancel* es simplemente utilizado para permitir a un rol conocer que otro rol no está más interesado en una acción particular y sirve de esta manera para cancelar compromisos previamente asumidos entre las partes.

Call-for-Proposal

Representa la acción de un rol *i* solicitando a otro rol *j* una propuesta para realizar alguna acción. El acto *cfp* puede ser utilizado para iniciar un proceso de negociación realizando un llamado solicitando propuestas a otros roles para que realicen una determinada acción. Este acto supone que quien recibe el mensaje luego enviará una propuesta al iniciador del *cfp*, a través de un mensaje con el acto *propose*.

Confirm

Representa la acción de un rol *i* de confirmar a otro rol *j* que una proposición es verdadera, cuando el rol *j* tiene incertidumbre o no conoce el estado de dicha proposición.

Disconfirm

Representa la acción de un rol *i* de informar a otro rol *j* que una determinada proposición es falsa, cuando *j* no está seguro del valor de la proposición o cree que la misma es verdadera.

Failure

Representa la acción de un rol que comunica a otro que se ha intentado realizar una determinada acción pero se ha fallado en el intento. Este acto es utilizado para informar que una acción previa (generalmente una solicitud realizada a través del acto *request*) fue considerada factible de realizar por el rol remitente, pero la misma no ha sido realizada o completada por alguna razón determinada. La razón de la falla es indicada en la proposición del mensaje.

Inform

Representa la acción de un rol de informar a otro que una proposición determinada es verdadera. Este es utilizado para que un rol remitente, el cual tiene un conocimiento acerca de una determinada proposición, la comunique al rol receptor para que este también pueda tener conocimiento de dicha proposición.

Not-Understood

Representa la acción de un rol *i* de informar al rol *j* que el acto de comunicación que previamente recibió de *j* no lo ha podido entender. Esto significa que el rol *i* informa a un rol *j* que percibió que *j* realizó una acción, pero que *i* no pudo entender lo

que *j* dijo. Existen varias razones posibles para esto: el remitente no ha sido diseñado para procesar un cierto acto o proposición del acto, o bien el remitente estaba esperando un mensaje diferente.

Propose

Representa la acción de enviar una propuesta para realizar una cierta acción, dadas ciertas precondiciones. Este acto es utilizado para hacer una propuesta o responder a una propuesta existente durante un proceso de negociación, proponiendo realizar una determinada acción, la cual puede estar sujeta a que ciertas condiciones sean verdaderas. Este acto supone que quien realiza la proposición informa al receptor que adoptará la intención de ejecutar una acción una vez que el receptor notifica al remitente de su intención que éste último ejecute la acción propuesta.

Query-If

Representa la acción de preguntar a otro rol si una proposición es o no verdadera. El rol remitente no tiene conocimiento acerca del valor de la proposición.

Query-Ref

Representa la acción de solicitar a otro rol por un determinado tipo de objeto que éste posee, sobre el cual el remitente sólo tiene referencia. Este acto también implica que el remitente está solicitando al receptor que le envíe un acto *inform* conteniendo el objeto solicitado.

Refuse

Representa la acción de negarse a realizar una acción dada y explicar las razones del rechazo. Este acto es utilizado cuando un rol no puede cumplir con una acción dada (generalmente solicitada a través de un acto *request*), debido a que no reúne los requerimientos necesarios para realizar la acción o porque no cree conveniente realizarla, según sus propios objetivos. El agente que recibe el acto *refuse* tiene derecho a creer que: la acción no ha sido realizada y que la acción no es factible. Las razones del rechazo son indicadas en la proposición asociada al acto.

Reject-Proposal

Representa la acción de rechazar una propuesta (previamente presentada) para realizar una acción durante una negociación. El rol que envía el rechazo informa al

receptor que no desea que el receptor realice una acción dada, la cual fue propuesta previamente por el receptor. La proposición asociada al acto indica las razones por las cuales la propuesta es rechazada.

Request

Representa la acción de un rol de solicitarle a otro que realice una acción. La proposición asociada al acto describe la acción a ser ejecutada.

Request-When

Representa la acción de un rol de solicitarle a otro que realice una acción cuando una proposición sea verdadera. Este acto supone que quien recibe el acto *request-when* debería rechazar el compromiso, o bien debería comprometerse a ejecutar la acción cuando la proposición sea verdadera. Este acto significa que el compromiso persistirá hasta que: la proposición sea verdadera, el rol que envió la solicitud cancele la misma (acto *cancel*), o bien el receptor decide que no puede comprometerse más con esta solicitud para lo cual debería enviar un mensaje *refuse* al rol que envió la solicitud.

Request-Whenever

Representa la acción de un rol de solicitarle a otro que realice una acción mientras alguna proposición sea verdadera, y a partir de entonces, cada vez que la proposición sea verdadera. Esto significa que luego de que la acción solicitada se realizó por primera vez debido a que la proposición fue verdadera, aunque luego la proposición sea falsa, la acción solicitada debería volver a realizarse si la proposición es nuevamente verdadera. Este acto representa un compromiso persistente para reevaluar una proposición dada y tomar una acción cuando sus valores cambien. El rol que originó la solicitud puede remover posteriormente este compromiso realizando la acción *cancel*. Además, al igual que en el acto *request-when*, el receptor puede desear no comprometerse más con la solicitud, para lo cual debería enviar un mensaje *refuse* al rol que envió la solicitud. La frecuencia de reevaluación de la proposición podría ser acordada por los roles a través de un proceso de negociación realizado previamente al envío del acto *request-whenever*.

A.2. Clasificación de los Actos de Comunicación

Una clasificación de estos actos de comunicación se muestra en la tabla A-1. En dicha tabla los actos son clasificados de acuerdo al tipo de interacciones que podrían

realizarse en un protocolo, como son: el intercambio de información, la solicitud de información, negociaciones, compromisos para llevar a cabo una acción y el manejo de errores. Cada categoría indica los actos están relacionados y deberían utilizarse en cada caso.

Acto de Comunicación	Intercambio de Información	Solicitud de Información	Negociación	Manejo de Errores
Accept-proposal			✓	
Agree		✓		
Cancel				✓
Call-for-Proposal			✓	
Confirm	✓			
Disconfirm	✓			
Failure				✓
Inform	✓			
Not-understood				✓
Propose			✓	
Query-If		✓		
Query-Ref		✓		
Refuse		✓		
Reject-Proposal			✓	
Request		✓		
Request-When		✓		
Request-Whenever		✓		

Tabla A-1. Una clasificación de los actos de comunicación de FIPA ACL.

ANEXO

B

REGLAS DE TRANSFORMACIÓN DE UP-COLBPIP A BPSS

Este anexo describe las reglas de transformación utilizadas en la transformación de modelos definidos con UP-ColBPIP a especificaciones de procesos en el lenguaje BPSS. En particular se describen las transformaciones de cada tipo de segmento de flujo de control de un protocolo de interacción.

B.1. Transformación de Segmentos de Flujo de Control

B.1.1. Transformación de Segmentos con el Operador Xor

A continuación se describen las reglas involucradas en el proceso de transformación de un segmento de flujo de control con el operador *Xor* (Figura B-1).

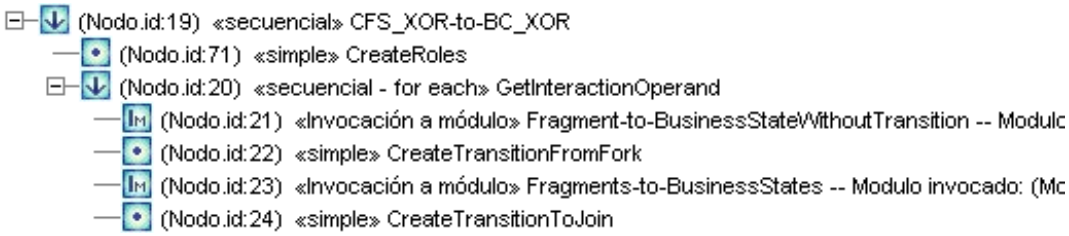


Figura B-1. Subárbol de reglas para la transformación de segmento *Xor*

En primer lugar, la regla *CFS_XOR-to-BC_XOR* es la encargada de crear la colaboración binaria correspondiente. La Figura B-2 muestra la semántica operacional de la regla. El patrón LHS indica que el objeto *IF* (pasado como parámetro) debe ser un segmento de flujo de control con el operador *Xor*. El patrón RHS indica que para representar este tipo de segmento se debe generar: una actividad de colaboración, dentro de la colaboración binaria (pasada como parámetro), la cual representa a un protocolo u a otro segmento (cuando existe anidamiento de segmentos); y la nueva colaboración binaria que representa al segmento. Además, en esta nueva colaboración que representa al segmento, se genera el estado de comienzo de la misma, el estado *Fork* con el atributo *type* establecido en "XOR", y el estado *Join* con el atributo *waitForAll* en "False". Esto permite representar la misma semántica de un segmento con el operador

Xor. También, se agrega un estado *Success* cuya transición indica que luego del estado *Join* se finaliza la colaboración binaria, para representar el fin del segmento.

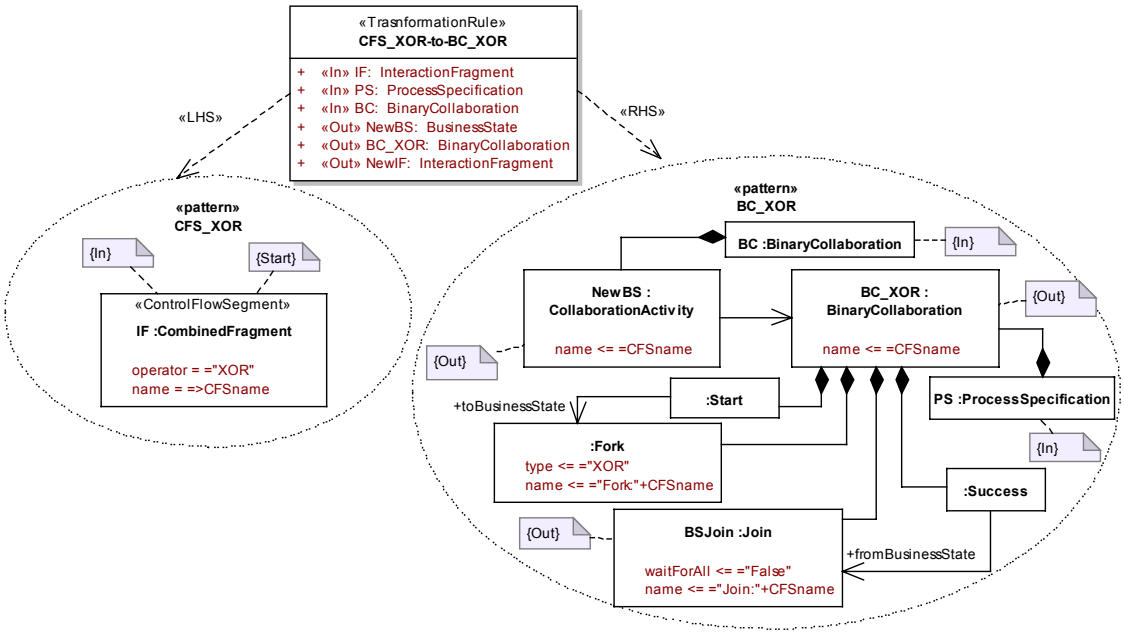


Figura B-2. Regla de Transformación *CFS_XOR-to-BC_XOR*

Una vez ejecutada la regla anterior, en la colaboración binaria generada que corresponde al segmento se crean los roles (Regla *CreateRoles*), de acuerdo a los roles del protocolo. Luego se transforman los elementos que componen los caminos de interacción del segmento. Como se indicó anteriormente, el concepto de camino de interacción (*InteractionOperand*) en UP-ColBPIP no tiene una contraparte directa en BPSS. Estos caminos deben ser expresados a través de las transiciones entre los estados correspondientes. En el caso de este tipo de segmento, cada camino encontrado (regla *GetInteractionOperand*) es expresado de la siguiente forma:

1. Se transforma el primer fragmento del camino y se genera un estado de negocio correspondiente. Esto se realiza invocando el módulo *Fragment-to-BusinessStateWithoutTransition*.
2. Se crea una transición, en donde el estado *Fork* es el origen y el estado de negocio generado anteriormente es el destino (regla *CreateTransitionFromFork*).
3. Se transforman los restantes elementos del camino. Esto se realiza invocando en forma recursiva al módulo *Fragments-to-BusinessStates*. En este caso se pasan

como parámetros de entrada al módulo el camino de interacción y la colaboración binaria que representa el segmento. El módulo retorna como parámetro de salida el último estado de negocio generado a partir del último fragmento del camino.

4. Se crea la transición entre este último estado de negocio generado y el estado *Join*, donde éste último es el destino de la transición (regla *CreateTransitionToJoin*). Esto representa la terminación de dicho camino.

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, a continuación se muestra el código generado en la transformación del segmento de control *BPO Changes* del protocolo de interacción *BlanketPurchaseOrder*. Sólo el código resaltado es generado por las reglas descritas anteriormente. El restante código es generado en las invocaciones al módulo *Fragment-to-BusinessStateWithoutTransition* y en las invocaciones recursivas al módulo *Fragments-to-BusinessStates*.

```
<BinaryCollaboration name = "BC:BPO Changes(XOR)" initiatingRoleID = "01B">
      <Role name = "Customer" nameID = "01A"/>
      <Role name = "Supplier" nameID = "01B"/>
      <Start toBusinessState = "Fork:BPO Changes(XOR)"/>
      <Fork name = "Fork:BPO Changes(XOR)"/>
      <!--This represents the First interaction operand of the Segment BPO Changes-->
      <BusinessTransactionActivity name = "BTA:propose_BPOChange"
              businessTransaction = "propose_BPOChange"
              fromRole = "Supplier" toRole = "Customer"/>
      <Transition fromBusinessState = "Fork:BPO Changes(XOR)"
              toBusinessState = "BTA:propose_BPOChange">
          <ConditionExpression expressionLanguage = "NL"
              conditionExpression = "ChangeRequired"/>
      </Transition>
      <CollaborationActivity name = "CA:BPO Change Responses(XOR)"
              fromRole = "Customer" toRole = "Supplier"/>
      <Transition fromBusinessState = "BTA:propose_BPOChange"
              toBusinessState = "CA:BPO Change Responses(XOR)"/>
      <Transition fromBusinessState = "CA:BPO Change Responses(XOR)"
              toBusinessState = "Join:BPO Changes(XOR)"/>
      <!--This represents the Second interaction operand of the Segment BPO Changes-->
      <BusinessTransactionActivity name = "BTA:accept-proposal_BPOResponse"
              businessTransaction = "accept-proposal_BPOResponse"
              fromRole = "Supplier" toRole = "Customer"/>
      <Transition fromBusinessState = "Fork:BPO Changes(XOR)"
              toBusinessState = "BTA:accept-proposal_BPOResponse">
          <ConditionExpression expressionLanguage = "NL"
              conditionExpression = "ChangeIsNotRequired"/>
      </Transition>
      <CollaborationActivity name = "CA:BPO Changes By Customer(If)"
              binaryCollaboration = "BC:BPO Changes By Customer(If)"
              fromRole = "Customer" toRole = "Supplier"/>
      <Transition fromBusinessState = "BTA:accept-proposal_BPOResponse"
              toBusinessState = "CA:BPO Changes By Customer(If)"/>
      <Transition fromBusinessState = "CA:BPO Changes By Customer(If)"
              toBusinessState = "Join:BPO Changes(XOR)"/>
      <Join name = "Join:BPO Changes(XOR)" waitForAll = "False"/>
      <Success fromBusinessState = "Join:BPO Changes(XOR)"/>
   </BinaryCollaboration>
```

B.1.2. Transformación de Segmentos con el Operador Or *y* And

La transformación de este tipo de segmentos es similar a la transformación de un segmento con el operador *Xor*. En primer lugar, las reglas *CFS-OR-to-BC_OR* y *CFS-AND-to-BC_AND* tienen la misma semántica que la regla *CFS-XOR-to-BC_XOR*, en el sentido que el patrón RHS crea los mismos objetos. La diferencia consiste en el valor asignado al atributo *type* del estado *Fork* y al atributo *waitForAll* del estado *Join*. Para expresar un segmento con el operador "Or", se debe asignar el valor "OR" al atributo *type* del estado *Fork* y se debe asignar el valor "False" al atributo *waitForAll* del estado *Join*. Para expresar un segmento con el operador "And", se debe asignar el valor "OR" al atributo *type* del estado *Fork* y se debe asignar el valor "True" al atributo *waitForAll* del estado *Join*.

El subárbol de reglas utilizado para transformar los elementos de los caminos de este tipo de segmento es el mismo al descrito anteriormente para el segmento con el operador *Xor*.

B.1.3. Transformación de Segmentos con el Operador If

Este tipo de segmento siempre contiene un camino a seguir cuando su condición es satisfecha y opcionalmente, puede contener otro camino alternativo, representando los pasos a seguir cuando dicha condición no es satisfecha. La transformación de este tipo de segmento es realizada de manera diferente, de acuerdo a si el segmento tiene definido el camino alternativo o no.n

La Figura B-3 muestra las reglas involucradas en el proceso de transformación de un segmento de flujo de control con el operador *If*, cuando éste posee un sólo camino.

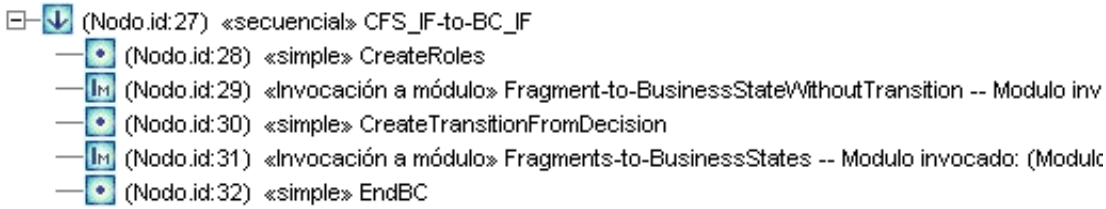


Figura B-3. Subárbol de reglas para la transformación de un segmento *If*

La regla *CFS_IF-to-BC_IF* indica cómo este tipo de segmento es transformado en una colaboración binaria (Figura B-4). El patrón LHS indica que el objeto *IF* debe ser un segmento de flujo de control con el operador *If*. El segmento debe poseer un camino de interacción con una condición asociada. Este camino es un parámetro de salida debido a que será utilizado en las siguientes reglas para transformar los elementos del mismo. Además, el segmento no debe poseer el camino alternativo (con la condición "Else"), como lo indica la restricción negativa asociada al objeto *ElsePath*. La condición definida en el primer camino es asignada al parámetro de salida *Condition*, para su posterior uso.

El patrón RHS indica que se debe generar: una actividad de colaboración, dentro de la colaboración binaria que representa a un protocolo o a otro segmento; y la colaboración binaria que representa el segmento. Además, en la colaboración binaria generada, el patrón indica que se debe adicionar un estado de comienzo y el estado *Decision* que representa la expresión condicional. También se agrega un estado final *Failure* en la colaboración binaria, el cual representará el caso en el que la condición del camino no sea satisfecha. Esto representa la terminación del segmento en la colaboración correspondiente.

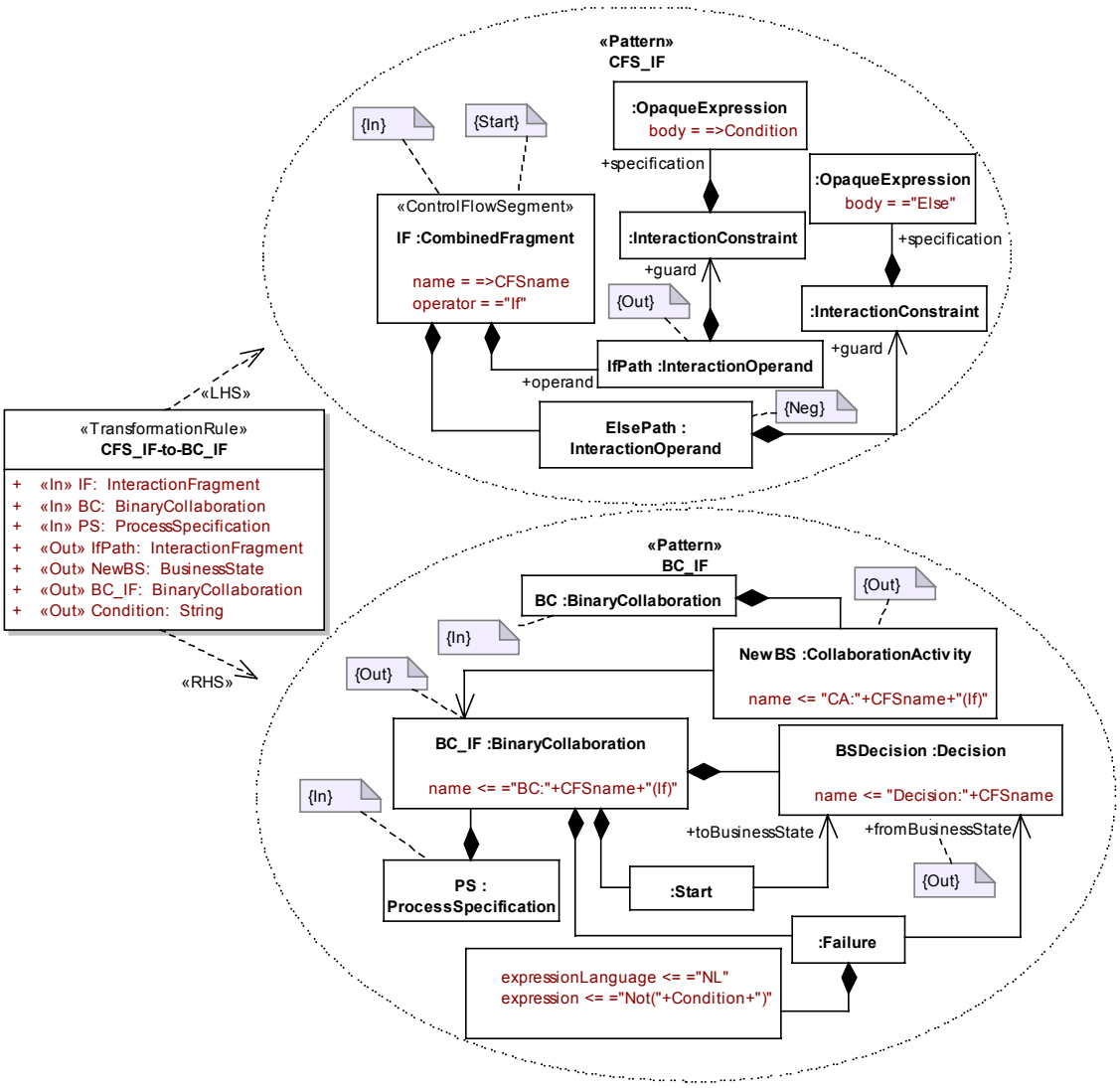


Figura B-4. Regla de Transformación *CFS_IF-to-BC_IF*

Luego de la ejecución de esta regla, se transforman los elementos que componen el camino de interacción del segmento. Para ello se siguen los siguientes pasos de acuerdo a lo definido en el árbol de reglas:

1. Se generan los roles en la colaboración binaria que representa al segmento (regla *CreateRoles*).
2. Se transforma el primer elemento del camino y se genera un estado de negocio correspondiente (módulo *Fragment-to-BusinessStateWithoutTransition*).
3. Se crea una transición desde el estado *Decision* al estado de negocio generado anteriormente. En dicha transición se agrega la condición expresada en el camino de interacción (regla *CreateTransitionFromDecision*).

4. Se transforman los restantes elementos del camino. Esto se realiza invocando en forma recursiva al módulo *Fragments-to-BusinessStates*. En este caso se pasan como parámetros de entrada al módulo el camino de interacción y la colaboración binaria que representa el segmento. El módulo retorna como parámetro de salida el último estado de negocio generado a partir del último elemento del camino.
5. Se agrega el estado de terminación *Success*, el cual expresa el fin de la colaboración que representa al segmento (regla *EndBC*).

La transformación de un segmento con el operador *If*, el cual posee dos caminos de interacción, es realizada a través de las reglas mostradas en la Figura B-5.

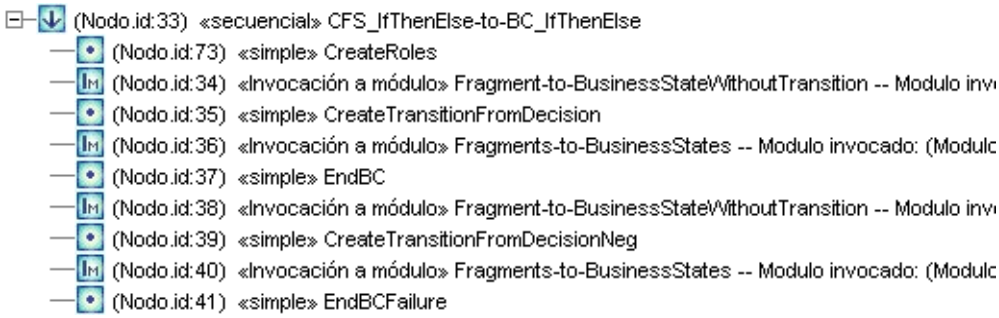


Figura B-5. Subárbol de reglas para la transformación de segmento *If* con dos caminos de interacción

La primer regla a ejecutar es la regla *CFS_IfThenElse-to-BC_IfThenElse*, la cual indica cómo este tipo de segmento es transformado en una colaboración binaria (Figura B-6). Esta regla es similar a la regla *CFS_IF-to-BC_IF*, excepto que en el patrón LHS, la condición negativa sobre el objeto *InteractionOperand* (representando el camino con la condición *Else*) no está presente. Además, el patrón RHS también es similar, excepto que no se crea el estado *Failure*.

Luego de la ejecución de esta regla, el procedimiento para transformar los elementos de los dos caminos que componen este tipo de segmento es el siguiente (Figura B-5):

1. Se generan los roles en la colaboración binaria que representa al segmento (regla *CreateRoles*).
2. Se transforman los elementos del primer camino:

a. Se transforma el primer elemento de este camino y se genera el estado de negocio correspondiente (módulo *Fragment-to-BusinessStateWithoutTransition*).

b. Se crea una transición desde el estado *Decision* al estado de negocio generado anteriormente. En dicha transición se agrega la condición expresada en el camino de interacción (regla *CreateTransitionFromDecision*).

c. Se transforman los restantes elementos del camino. Esto se realiza invocando en forma recursiva al módulo *Fragments-to-BusinessStates*.

d. Se agrega el estado de terminación *Success*, indicando que la transición a dicho estado es realizada desde el último estado de negocio generado por el paso anterior (regla *EndBC*). Esto representa la finalización del primer camino del segmento.

3. Se transforman los elementos del segundo camino:

a. Se transforma el primer elemento de este camino y se genera un estado de negocio correspondiente (módulo *Fragment-to-BusinessStateWithoutTransition*).

b. Se crea una transición desde el estado *Decision* al estado de negocio generado anteriormente. Para representar la condición "Else", en dicha transición se agrega la condición expresada en el primer camino de interacción (regla *CreateTransitionFromDecisionNeg*), pero en forma negada.

c. Se transforman los restantes elementos del camino. Esto se realiza invocando en forma recursiva al módulo *Fragments-to-BusinessStates*.

d. Se agrega el estado de terminación *Failure*, para indicar que se ejecutó el camino contrario. En este estado se indica que la transición a dicho estado es realizada desde el último estado de negocio generado por el paso anterior (regla *EndBCFailure*). Este representa la finalización del segundo camino del segmento.

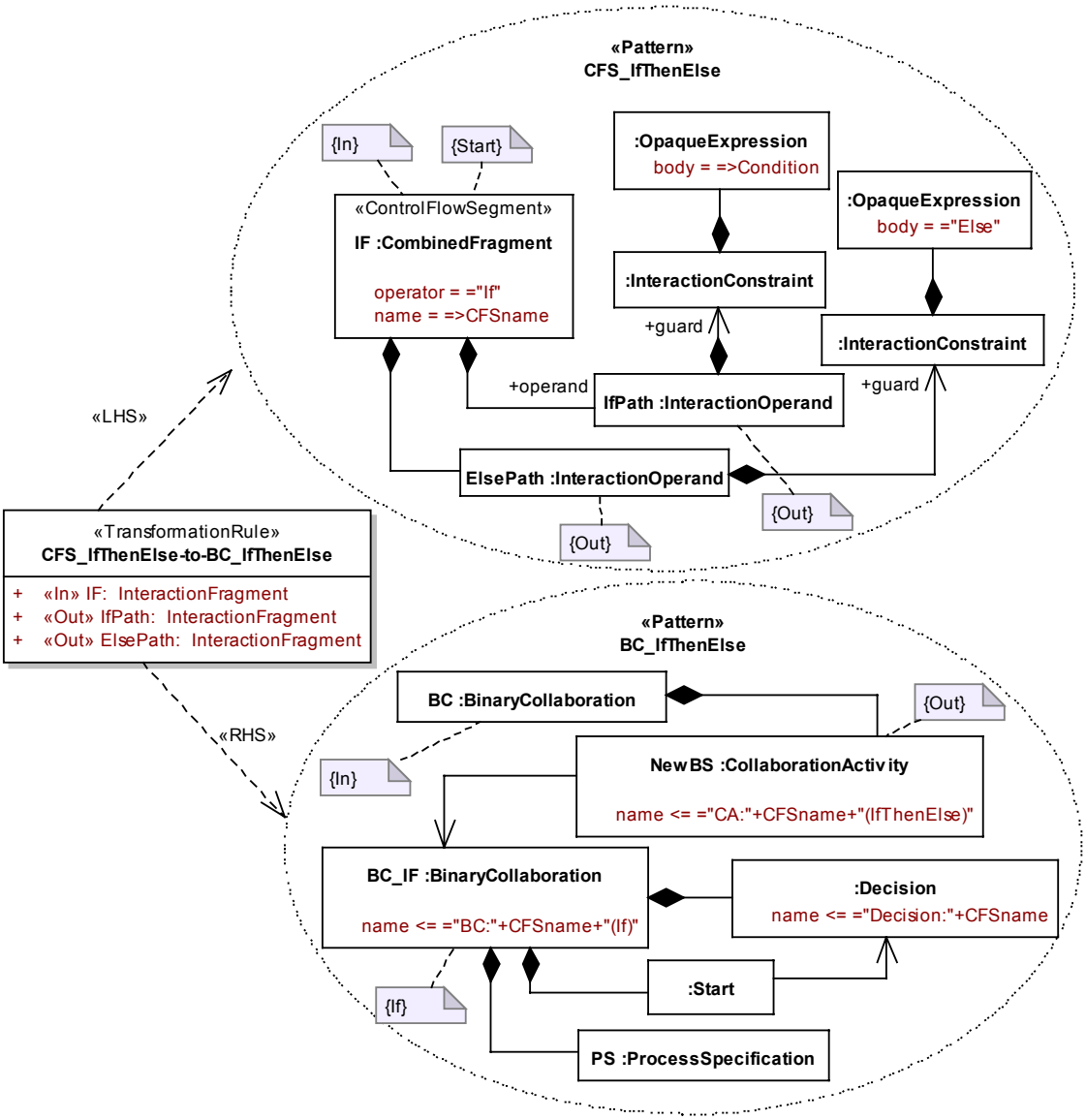


Figura B-6. Regla de Transformación *CFS_IfThenElse-to-BC_IfThenElse*

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, a continuación se muestra el código generado en la transformación del segmento de control *BPO Changes By Customer* del protocolo de interacción *BlanketPurchaseOrder*. Este segmento posee dos caminos de interacción. Sólo el código resaltado es generado por las reglas descritas anteriormente. El restante código es generado en las invocaciones al módulo *Fragment-to-BusinessStateWithoutTransition* y en las invocaciones recursivas al módulo *Fragments-to-BusinessStates*.

```
<BinaryCollaboration name = "BC:BPO Changes(XOR)" initiatingRoleID = "01B">
      ....
      <CollaborationActivity name = "CA:BPO Changes By Customer(If)"
              binaryCollaboration = "BC:BPO Changes By Customer(If)"
              fromRole = "Customer" toRole = "Supplier"/>
      ....
   </BinaryCollaboration>
   <BinaryCollaboration name = "BC:BPO Changes By Customer(If)">
      <Role name = "Customer" nameID = "01A"/>
      <Role name = "Supplier" nameID = "01B"/>
      <Start toBusinessState = "Decision:BPO Changes By Customer"/>
      <Decision name = "Decision:BPO Changes By Customer"/>
      <BusinessTransactionActivity name = "BTA:propose_BPOChange"
              businessTransaction = "propose_BPOChange"
              fromRole = "Customer" toRole = "Supplier"/>
      <Transition fromBusinessState = "Decision:BPO Changes By Customer"
              toBusinessState = "BTA:propose_BPOChange">
          <conditionExpression expressionLanguage = "NL"
              conditionExpression = "ChangeRequired"/>
      </Transition>
      <Success fromBusinessState = "BTA:propose_BPOChange"/>
      <Failure fromBusinessState = "Decision:BPO Changes By Customer">
          <conditionExpression expressionLanguage = "NL"
              conditionExpression = "Not(ChangeRequired)"/>
      </Failure>
   <BinaryCollaboration/>
```

B.1.4. Transformación de Segmentos con el Operador Loop

Existen dos tipos de segmento con el operador *Loop*. Uno representa un bucle de tipo *For*, en donde el camino del segmento es ejecutado al menos una vez. Este tipo de segmento tiene la condición "(1,n)" asociada al camino. El otro representa un bucle de tipo *While*, en donde el camino del segmento puede ser ejecutado cero o más veces. El camino del segmento tiene la condición "(0,n)"

Para representar el segmento con un bucle de tipo *For*, se deben seguir los pasos y las reglas mostradas en la Figura B-7.

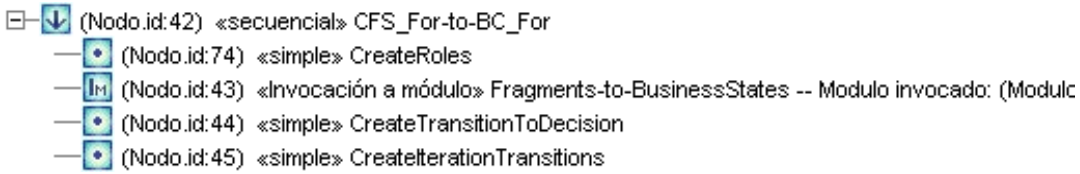


Figura B-7. Subárbol de reglas para la transformación de segmento de tipo *For*

La regla *CFS_For-to-BC_For* indica cómo un segmento de este tipo es transformado en una colaboración binaria (Figura B-8). El patrón LHS indica que el objeto *If* debe ser un segmento de flujo de control con el operador *Loop*. La condición definida en el camino del segmento es asignada al parámetro de salida *ForCondition*, para su posterior uso. Además, la restricción debe tener "1" como el valor de mínima del bucle.

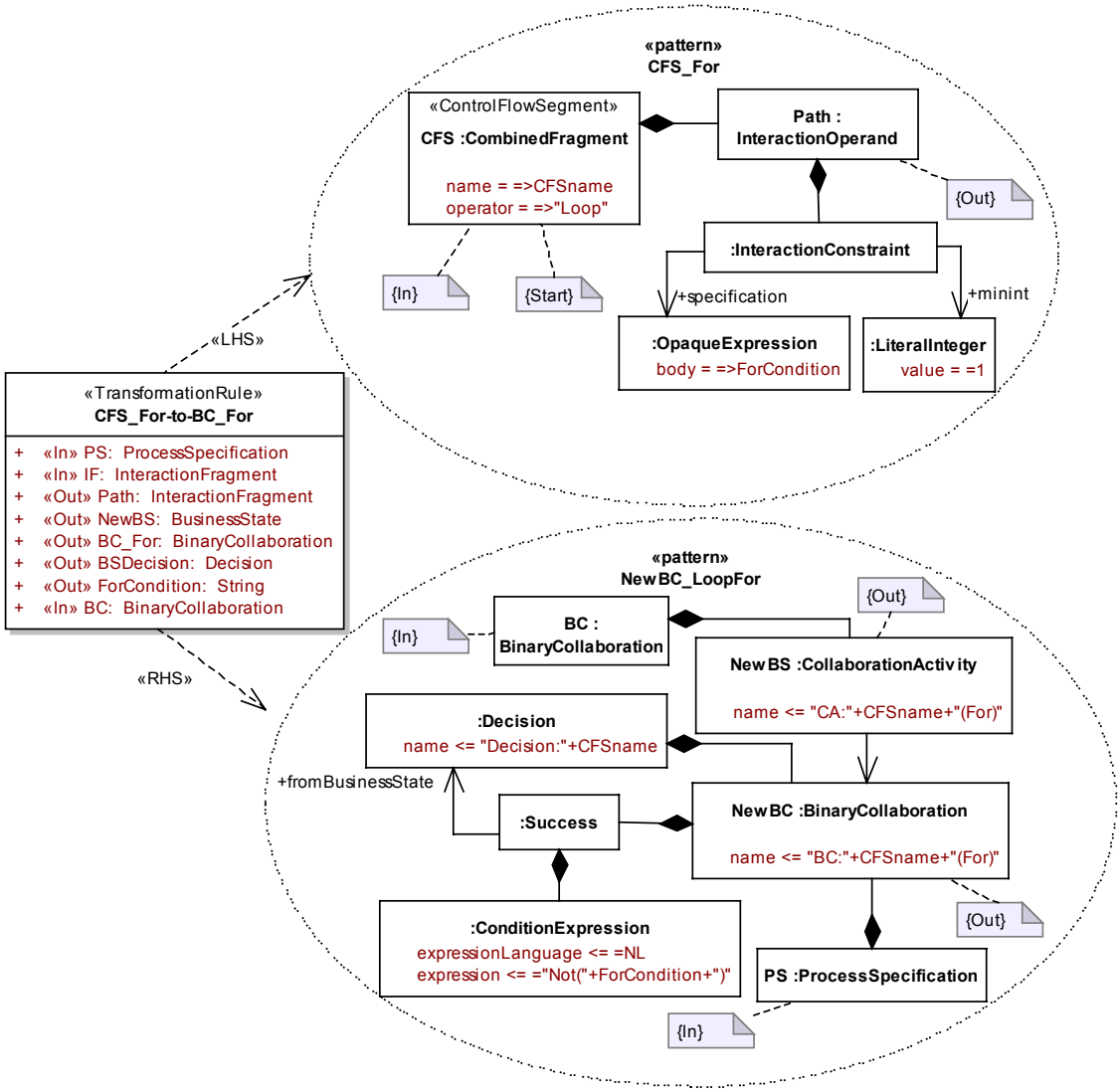


Figura B-8. Regla de Transformación *CFS_For-to-BC_For*

El patrón RHS indica que se debe generar: una actividad de colaboración y la colaboración binaria que representa este tipo de segmento. Además, en la nueva colaboración binaria se adiciona el estado *Decision*, el cual permitirá representar la condición del bucle. En esta colaboración no se adiciona una transición desde un estado de comienzo al estado de decisión, debido a que un bucle *For*, siempre debe ejecutarse al menos una vez.

Luego de la generación de la colaboración binaria representando este segmento, se realizan los siguientes pasos (Figura B-7):

1. Se generan los roles en la colaboración binaria que representa el segmento (regla *CreateRoles*).

2. Se generan los estados de negocio correspondientes a los elementos del camino de interacción. Esto se realiza invocando al módulo *Fragments-to-BusinessStates* en forma recursiva. Este módulo retorna como parámetro de salida el último estado de negocio generado.
3. Este estado de negocio es utilizado como parámetro de entrada en la siguiente regla invocada (Regla *CreateTransitionToDecision*), la cual crea la transición desde dicho estado hacia el estado *Decision* representando la condición del bucle.
4. Se representa la iteración del bucle, a través de la ejecución de la regla *CreateForIteration*, la cual crea una transición entre el estado decisión y el primer estado generado en la colaboración binaria. A esta transición se agrega la condición que representa el bucle. Además, esta regla adiciona un estado final *Success* y una transición desde el estado *Decision* hacia dicho estado final. En esta transición también se agrega la condición que representa el bucle, pero en este caso la misma es definida en forma negada.

La transformación de un segmento con un bucle de tipo *While*, se realiza a través del subárbol de reglas mostrado en la Figura B-9.

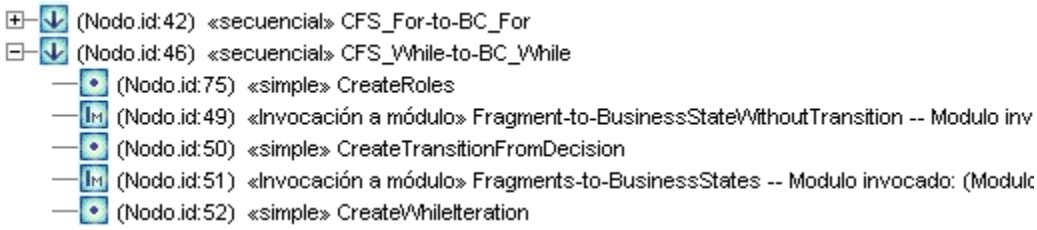


Figura B-9. Subárbol de reglas para la transformación de segmento de tipo *While*.

La regla *CFS_While-to-BC_While* indica cómo un segmento de este tipo es transformado en una colaboración binaria (Figura B-10). El patrón LHS indica que el objeto *IF* debe ser un segmento de flujo de control con el operador *Loop*. La condición definida en el camino asociado es asignada al parámetro de salida *WhileCondition*, para su posterior uso. Además, la restricción debe tener "0" como el valor de mínima del bucle.

El patrón RHS indica que se debe generar: una actividad de colaboración y la colaboración binaria que representa este tipo de segmento. Además, se adicionan a la colaboración el estado de comienzo, el estado *Decision* y la transición entre dichos

estados. El estado *Decision* representa la condición del bucle *While*. También se adiciona el estado *Success*, el cual define una transición a dicho estado a partir del estado *Decision*. Esta estado contiene una condición que representa la salida del bucle, cuando la condición no es más satisfecha.

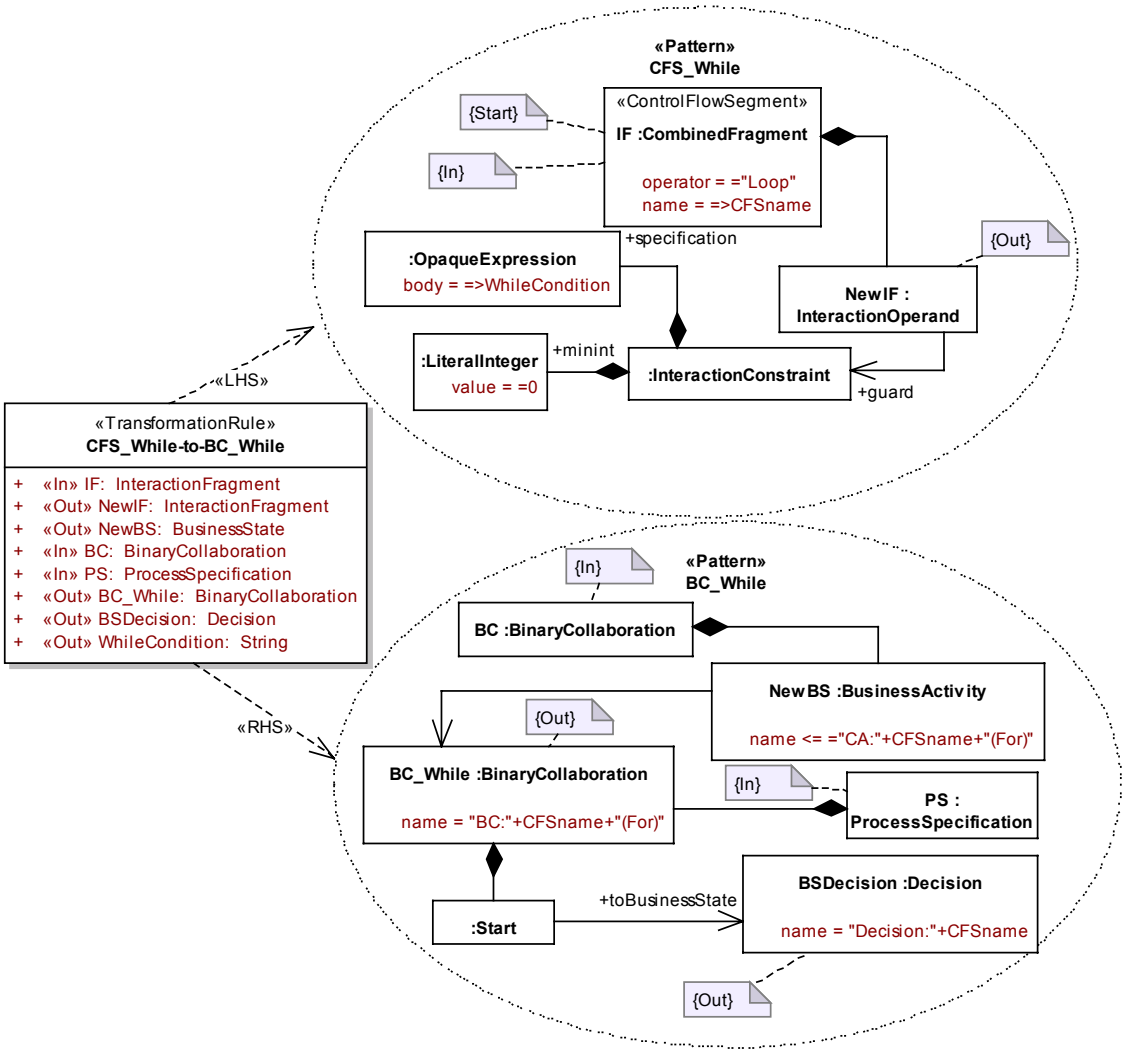


Figura B-10. Regla de Transformación *CFS_While-to-BC_While*

Luego de la generación de la colaboración binaria representando este segmento, se realizan los siguientes pasos (Figura B-9):

1. Se transforma el primer elemento del camino y se genera el estado de negocio correspondiente (regla *Fragment-to-BusinessStateWithoutTransition*).
2. Se crea una transición desde el estado *Decision* y el estado de negocio creado anteriormente. A esta transición se adiciona la condición que debe ser verificada para que el bucle *While* sea ejecutado. Esto es realizado a través de la regla

CreateTransitionFromDecision. Esta regla también es utilizada en la transformación de segmento con el operador *If*.

3. Se transforman los restantes elementos del camino en los estados de negocio correspondientes. Esto se realiza invocando en forma recursiva al módulo *Fragments-to-BusinessStates*.

4. Se crea una transición desde el último estado de negocio generado hacia el estado de decisión, para representar la iteración del bucle (Regla *CreateWhileIteration*).

CÓDIGO GENERADO EN EL EJEMPLO DE TRANSFORMACIÓN

Como ejemplo, a continuación se muestra el código generado en la transformación del segmento de control *BPO Negotiation* del protocolo de interacción *BlanketPurchaseOrder*. Sólo el código resaltado es generado por las reglas descritas anteriormente. El restante código es generado en la recursiva al módulo *Fragments-to-BusinessStates*.

```
<BinaryCollaboration name = "BC:Blanket Purchase Order"
            timeToPerform = "P6D">
      ....
      <CollaborationActivity name = "CA:BPO Negotiation(For)"
            binaryCollaboration = "BC:BPO Negotiation(For)"
            fromRole = "Customer" toRole = "Supplier"/>
      ....

   </BinaryCollaboration>
   <BinaryCollaboration name = "BC:BPO Negotiation(For)">
      <Role name = "Customer" nameID = "01A"/>
      <Role name = "Supplier" nameID = "01B"/>
      <CollaborationActivity name = "CA:BPO Changes(XOR)"
            binaryCollaboration = "BC:BPO Changes(XOR)"
            fromRole = "Supplier" toRole = "Customer"/>
      <Start toBusinessState = "CA:BPO Changes(XOR)"/>
      <Decision name = "Decision:BPO Negotiation"/>
      <Transition fromBusinessState = "CA:BPO Changes(XOR)"
            toBusinessState = "Decision:BPO Negotiation"/>
      <Transition fromBusinessState = "Decision:BPO Negotiation"
            toBusinessState = "CA:BPO Changes(XOR)">
            <ConditionExpression expressionLanguage = "NL"
                conditionExpression = "NegotiationIsRequired"/>
      </Transition>
      <Success fromBusinessState = "Decision:BPO Changes">
            <ConditionExpression expressionLanguage = "NL"
                conditionExpression = "Not(NegotiationIsRequired"/>
      </Success>
   </BinaryCollaboration>
```

B.1.5. Transformación de un Segmento de Control con el Operador Transaction

Conceptualmente, un segmento de este tipo es diferente al de una transacción de negocio en BPSS, debido a que este segmento puede involucrar el intercambio de más de dos mensajes. Por lo tanto, el mismo es transformado en una colaboración binaria, ya que en BPSS, la misma también posee las propiedades de una transacción de negocio. La Figura B-11 muestra el subárbol de reglas que realiza esta transformación.

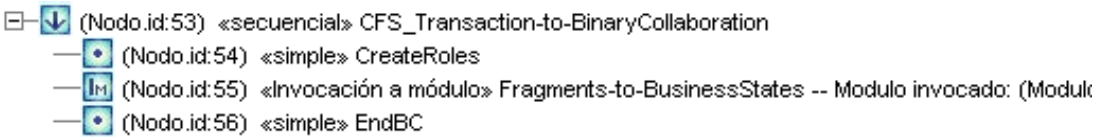


Figura B-11. Subárbol de reglas para la transformación de segmento *Transaction*.

La primer regla a ejecutar es la regla *CFS_Transaction-to-BinaryCollaboration*. La Figura B-12 muestra la semántica operacional de la regla. El patrón LHS indica que el objeto *IF* debe ser un segmento con el operador *Transaction*. Este tipo de segmento siempre tiene un único camino. El patrón RHS genera la actividad de colaboración y la colaboración binaria que esta referencia, la cual representa al segmento. Además se genera el estado de comienzo de la nueva colaboración binaria generada.

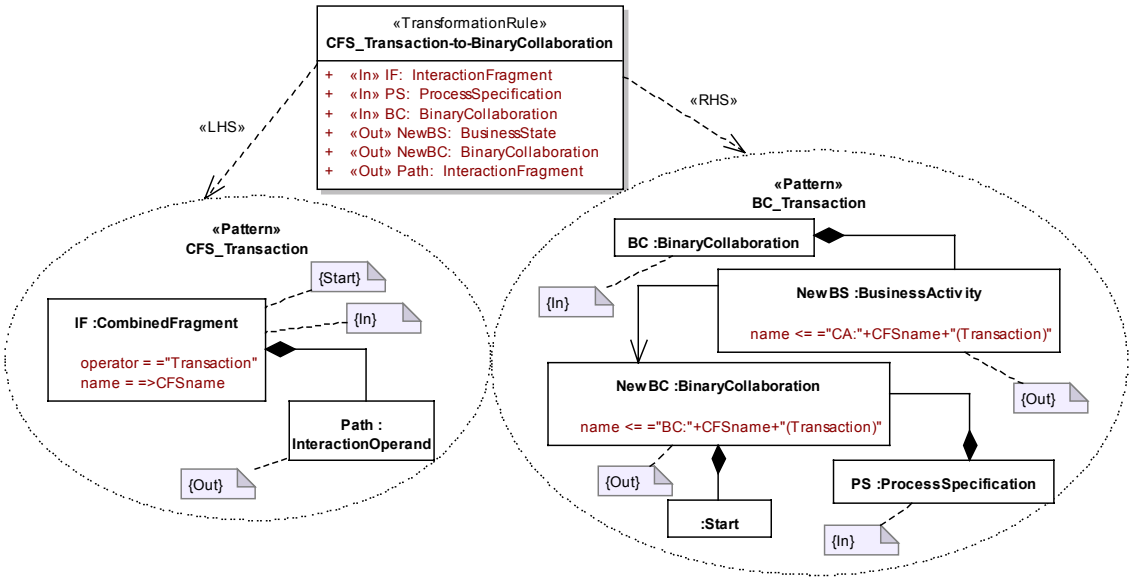


Figura B-12. Regla de Transformación CFS_Transaction-to-BinaryCollaboration

Luego de la generación de la colaboración binaria representando este segmento, se realizan los siguientes pasos (Figura B-11):

1. Se generan los roles en la colaboración binaria que representa este segmento (regla *CreateRoles*).
2. Se generan los estados de negocio correspondientes a los elementos del camino de interacción del segmento. Esto se realiza invocando al módulo *Fragments-to-BusinessStates* en forma recursiva. Este módulo retorna como parámetro de salida el último estado de negocio generado.
3. Se finaliza la colaboración binaria generada (regla *EndBC*).

B.1.6. Transformación de un Segmento con el Operador Exception

El subárbol de reglas utilizado para transformar este tipo de segmento es el mismo que el utilizado para transformar un segmento con el operador *Transaction*, ya que un segmento de este tipo también contiene un único camino. La única diferencia es la regla *CFS_Exception-to-BinaryCollaboration*. No obstante, la semántica de la misma es similar al de la regla *CFS_Transaction-to-BinaryCollaboration*.

B.1.7. Transformación de un Segmento con el Operador Stop

Este tipo de segmento es utilizado para realizar interacciones, luego de las cuales se debe finalizar la ejecución del protocolo que contiene a dicho segmento. Este segmento también contiene un único camino. El subárbol de reglas utilizado para transformar este tipo de segmento es el mismo que el utilizado para transformar un segmento con el operador *Transaction*. La única diferencia es la regla *CFS_Stop-to-BinaryCollaboration*. Aunque la semántica de la misma es similar al de la regla *CFS_Transaction-to-BinaryCollaboration*, excepto que el patrón RHS genera un estado *Failure*, el cual representa el estado final que debe suceder luego de la realización de la actividad de colaboración que invoca a la colaboración binaria que representa el segmento.

B.1.8. Transformación de un Segmento con el Operador Cancel

Un segmento de control con el operador *Cancel* no tiene una contraparte directa en BPSS, ya que en este lenguaje no es posible indicar lo que debe realizarse ante una excepción que puede ocurrir en cualquier punto de la coreografía. Aunque la transformación de este tipo de segmento podría ser representada en cierta forma en BPSS agregando una transición desde cada uno de los estados de negocio generados

hacia la colaboración binaria representando este tipo de segmento, la misma requiere un esfuerzo adicional considerable. Por lo tanto, este tipo de segmento es ignorado en este modelo de transformación.

BIBLIOGRAFÍA

Agrawal y otros (2003)

Agrawal, A., Karsai, G., Ledeczi, A. *A UML-based Graph Transformation Approach for Implementing Domain-Specific Generators*. Second International Conference on Generative Programming and Component Engineering, (GPCE '03) Submitted. Erfurt, Germany, 2003.

Akehurst y Kent (2002)

Akehurst, D. H., Kent, S. *A Relational Approach to Defining Transformations in a Metamodel*. UML 2002 – The Unified Modeling Language 5th International Conference. J.-M. Jézéquel, H. Hussmann, S. Cook (Eds.), LNCS 2460, pp 243-258. Dresden, Germany, 2002.

Alt y otros (2000)

Alt, R., Grünauer, K. M., Reichmayr, C. *Interaction of Electronic Commerce and Supply Chain Management – Insights from 'The Swatch Group'*. Proceedings of The 33rd Hawaii International Conference on System Sciences. Hawai, USA, 2000.

Anderson (2000)

Anderson, G. *From Supply Chain to Collaborative Commerce Networks: The Next Step in Supply Chain*. Achieving Supply Chain Excellence through Technology (ASCET), 2, 2000.

Angelen y Grefen (2002)

Angelen, S., Grefen, P. *Support for B2B e-Contracting - The Process Perspective*. Knowledge and Technology Integration in Production and Services - Proceedings 5th International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, pp. 87-96. Cancún, México, 2002.

Austin (1962)

Austin, J. L. *How to do things with Words*. Harvard University Press, Cambridge MA, 1962.

Aviv (2001)

Aviv, Y. *The Effect of Collaborative Forecasting on Supply Chain Performance*. Management Science, 47, pp 1326-1343, 2001.

Baghdadi (2004)

Baghdadi, Y. *ABBA: an architecture for deploying business-to-business electronic commerce applications*. Electronic Commerce Research and Applications, 3, pp 190-212, 2004.

Baïna y otros (2004)

Baïna, K, Benatallah, B., Cassati, F., Toumani, F. *Model-Driven Web Service*

Development. CAiSE'04, Springer, pp. 290-306, 2004.

Bauer y otros (2001)

Bauer, B., Müller, J.P., Odel, J. *Agent UML: A Formalism for Specifying Multiagent Software Systems*. Int. Journal of Software Engineering and Knowledge Engineering, 11 (3), pp 1-24, 2001.

BEA y otros (2003)

BEA, IBM, Microsoft, SAP, Siebel. *Business Process Execution Language for Web Services (BPEL)*. http://www-106.ibm.com/developerworks/library/ws-bpel/. Mayo, 2003.

Benatallah y otros (2003)

Benatallah, B., Casati, F., Toumani, F., Hamadi, R. *Conceptual Modeling of Web Service Conversations*. CAiSE'2003, Springer, pp. 449–467, 2003.

Bergholtz y otros (2003)

Bergholtz, M.; Jayaweera, P.; Johannesson, P.; Wohed, P. *Reconciling Physical, Communicative, and Social/Institutional Domains in Agent Oriented Information Systems - A Unified Framework*. Proceedings of the Int. Workshop on Agent Oriented Information Systems, ER 2003, LNCS, Vol. 2814, pp 180-194, Springer. Berlin, 2003.

Bernauer y otros (2003)

Bernauer, M., Kappel, G., Kramler, G. *Comparing WSDL-based and ebXML-based Approaches for B2B Protocol Specification*. Proceedings of ISOC'03, 2003.

Bernauer y otros (2003b)

Bernauer, M., Kappel, G., Kramler, G., Retschitzegger, W. *Specification of Interorganizational Workflows - A Comparison of Approaches*. Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003), 2003.

Bezivin y otros (2003)

Bezivin, J., Dupe, G., Jouault, F., Pitette, G., Rougui, J.E. *First experiments with the ATL model transformation language: Transforming XSLT into XQuery*. 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture, 2003.

Bussler (2001)

Bussler, C. *The Role of B2B Protocols in Inter-enterprise Process Execution*. Proceedings of Workshop on Technologies for E-Services (TES 2001). Rome, Italy, September 2001.

Bussler (2002)

Bussler, C. *The Role of B2B Engines in B2B Integration Architectures*. ACM SIGMOD Record, Special Issue on Data Management Issues in E-Commerce,

31(1), 2002.

Caliusco y otros (2004a)

Caliusco, M.L., Villarreal, P., Arredondo, F., Zanel, C., Zucchini, D., Chiotti, O., Galli, M.R. *Decentralized Management Model of a Partner-to-Partner Collaborative Relationship*. Proceedings of The 2nd World POM Conference on POM and 15th annual POM Conference. Cancún, México, 2004.

Caliusco y otros (2004b)

Caliusco, M.L., Galli, M.R., Chiotti, O. *Ontology and XML-based Specifications for Collaborative B2B Relationships*. Special Issue of Best Papers presented at III Jornadas Iberoamericanas de Ingeniería de Software e Ingenierma de Conocimiento (JIISIC), 2004.

Carlson (2001)

Carlson, D. *Modeling XML Applications with UML – Practical e-Business Applications*. Addison-Wesley, 2001.

Carol (2001)

Carol, M. *Collaborative commerce: The next big thing in global manufacturing*. EAI Journal, pp 20-24, 2001.

Cavusoglu y otros (2001)

Cavusoglu, T., Gulledge, T., Kessler, T. *Aligning the supply chain operations reference (SCOR) model with enterprise applications*. Proceedings of The Portland International Conference on Management of Engineering and Technology (PICMET'01). Portland, USA: IEEE, 2001.

Chapman y Petersen (2000)

Chapman, L., Petersen, M. *Demand Activated Manufacturing Architecture (DAMA) - Model for Supply Chain Collaboration*. Proceedings of The International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2000). Temper, Arizona, USA, 2000.

Chen y Hsu (2001)

Chen, Q., Hsu, M. *Inter-Enterprise Collaborative Business Process Management*. Proceedings of the 17th International Conference on Data Engineering (ICDE). Heidelberg, Germany, 2001.

Chen y otros (2000)

Chen, F., Drezner, Z., Ryan, J., Simchi-Levi, D. *Quantifying the bullwhip effect in a simple supply chain: the impact of forecasting, lead times and information*. Management Science, 46, pp 436-443, 2000.

Chen y otros (2001)

Chen, Q., Hsu, M., Dayal, U. *Peer-to-Peer Collaborative Internet Business Servers*. TechnicalReport HPL-2001-14, HP Laboratories Palo Alto, 2001.

Chen y otros (2003)

Chen, Q., Hsu, M., Mehta, V. *How Public Conversation Management Integrated with Local Business Process Management*. Proceedings of the IEEE International Conference on E-Commerce (CEC'03), 2003.

Christiaanse y Markus (2003)

Christiaanse, E., Markus, M.L. *Participation in Collaboration Electronic Marketplaces*. Proceedings of The HICSS 2003. Hawai, USA, 2003.

CIDX (2004)

Chemical Industry Data eXchange. *Supply Chain Collaboration*. http://www.cidx.org/ChemeStandards/Download.asp.

CompTIA EIDX (2004)

CompTIA Electronics Industry Data eXchange. *Replenishment Scenario 4 - Forecast-based Supplier-Managed Inventory (SMI)*, V. 1.0. http://www.comptia.org/sections/eidx/business_process/replenishment/replmodl4.asp.

Czarnecki y Helse (2003)

Czarnecki, K., Helse, S. *Classification of Model Transformation Approaches*. Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in theContext of the Model Driven Architecture, USA, 2003

Dai y Kauffman (2001)

Dai, Q., Kauffman, R. J. *Business models for Internet-based e-procurement systems and B2B electronic markets: an exploratory assessment*. Proceedings of The 34th Hawaii International Conference on System Sciences. Hawai, USA, 2001.

Dayal y otros (2001)

Dayal, U., Hsu, M., Ladin, R. *Business Process Coordination: State of the Art, Trends, and Open Issues*. Proceedings of the 27th BLVD. Conference. Roma, Italia, 2001.

De Lara y Vangheluwe (2002)

De Lara, J., Vangheluwe, H. *AToM3: A Tool for Multi-Formalism Modelling and Meta-Modelling*. European Conferences on Theory And Practice of Software Engineering ETAPS02, Fundamental Approaches to Software Engineering (FASE). Lecture Notes in Computer Science 2306, pp 174-188. Springer-Verlag, 2002.

Dietz y otros (1998)

Dietz, J., Goldkuhl, G., Lind, M., Reijswoud, V. *The Communicative Action Paradigm for Business Modelling – A Research Agenda*. Proceedings of the Third International Workshop on The Language Action Perspective on Communication Modelling (LAP'98). Estocolmo, Suecia, 1998.

Dietz (1999)

Dietz, J. L. *Understanding and Modelling Business Processes with DEMO*. Proceedings of Conceptual Modeling - ER '99, 18th International Conference on Conceptual Modeling, Lecture Notes in Computer Science, Vol. 1728, pp. 188-202, Springer, Paris, France, 1999.

Dietz (2002)

Dietz, Jan L. G. *The Atoms, Molecules and Matter of the Organizations*. Proceedings of the 7th International Workshop on the Language Action Perspective on Communication Modeling (LAP 2002). Delft, Holanda, 2002.

Domínguez Machuca y otros (1995)

Domínguez Machuca, J., García González, S., Domínguez Machuca, M., Ruíz Jiménez, A., Álvarez Gil, M. *Dirección de Operaciones. Aspectos tácticos y operativos en la producción y los sercicios* (First Edition). McGraw-Hill, Madrid, España, 1995.

Duddy y otros (2003)

Duddy, K., Gerber, A., Lawley, M.J., Raymond, K., Steel, J. *Model Transformation: A Declarative, Reusable Patterns Approach*. Proceedings 7th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2003), pp 174-185, 2003.

Dumas y Hofstede (2000)

Dumas, M., Hofstede, A. *UML Activity Diagrams as a Workflow Specification Language*. Proceedings of the 4th International Conference UML - The Unified Modeling Language: Modeling Languages, Concepts, and Tools. Lecture Notes in Computer Science, Vol. 2185, pp. 76-90.Toronto, Canada, Springer, 2000.

Eclipse (2004a)

EMF-based UML™ 2.0 Metamodel Implementation (Eclipse UML2). http://www.eclipse.org/uml2/. 2004.

Eclipse (2004b)

Eclipse Modeling Framework. http://www.eclipse.org/uml2/. 2004.

Ehrig y otros (1999)

H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook on Graph Grammars and Computing by Graph Transformation*. volume 2 Applications, Languages and Tools. World Scientic Publishing Co., Inc., 1999.

Eriksson y Penker (2000)

Eriksson, H., Penker, M. *Business Modeling with UML: Business Patterns at Work*. John Wiley & Sons, USA, 2000.

Ferrara (2004)

Ferrara, A. *Web Services: a Process Algebra Approach*. The Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004). New York City, New York, USA. November 15-18, 2004.

FIPA (2002)

FIPA (Foundation for Intelligent Physical Agents). *FIPA Communicative Act Library Specification*. http://www.fipa.org/specs/fipa00037/. 2002.

Flores y Ludlow (1980)

Flores, F., Ludlow, J. *Doing and Speaking in the Office*. Decision Support Systems: Issues and Challenges. G. Fick, H. Spraque Jr. (Eds.). Pergamon Press, pp. 95-118. New York, 1980.

Fu y Piplani (2004)

Fu, Y., Piplani, R. *Supply-side collaboration and its value in supply chains*. European Journal of Operational Research, 152, pp 281-288, 2004.

Fu y otros (2004)

Fu, X., Bultan, T., Su, J. *Analysis of Interacting BPEL Web Services*. 13th International World Wide Web Conference (WWW), New York, USA, 2004.

Fuentes y Vallecillo (2004)

Fuentes, L., Vallecillo, A. *An Introduction to UML Profiles, UPGRADE*. The European Journal for the Informatics Professional, 5 (2), pp 6-13, 2004.

Gardner (2003)

Gardner, T. *UML Modelling of Automated Business Processes with a Mapping to BPEL4WS*. First European Workshop on Object Orientation and Web Services (EOOWS), 2003.

Giménez y Lourenço (2004)

Giménez, C., Lourenço, H. R. *E-Supply Chain Management: Review, Implications and Directions for Future Research*. Proceedings of the 11th International Anual eurOMA Conference. France, 2004.

Goethals y otros (2005)

Goethals, F., Vandenbulcke, J., Lemahieu, W., Snoeck, M., Cumps, B. *Two Basic Types of Business-to-Business Integration*. International Journal of E-Business Research, 1, pp 1-15, 2005.

Gogolin (2003)

Gogolin, M. *Success and Failure of Collaboration Platforms*. Proceedings of The Tenth Research Symposium on Emerging Electronic Markets 2003. University of Bremen, Germany, 2003.

Goldkuhl (1996)

Goldkuhl, G. *Generic Business Frameworks and Action Modelling*. Proceedings of the International Workshop on the Language/Action Perspective on Communication Modeling. Verharen E., Rijst N. van, Dietz J. (Eds.). Oisterwijk, The Netherlands, 1996.

Goldkuhl y Lind (2004)

Goldkuhl, G., Lind, M. *Developing E-Interactions – a framework for business capabilities and exchanges*. Proceedings of the 12th European Conference on Information Systems (ECIS-2004). Turku, Finlandia, 2004.

Grefen y otros (2000)

Grefen, P., Aberer, K., Hoffner, Y., Ludwig, H. *CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises*. Int. Journal of Computer Systems Science & Engineering, 15 (5), pp 277-290, 2000.

Grieger (2003a)

Grieger, M. *Information integration in supply chain management: Utilizing Internet business models*. Proceedings of the 15th Annual Conference for Nordic Researchers in Logistics (Nofoma), pp. 26-41. Oulu, Finland, 2003.

Grieger (2003b)

Grieger, M. *Electronic marketplaces: A literature review and a call for supply chain management research*. European Journal of Operational Research, Elseiver, 144, pp 280-294, 2003.

Grieger y Kotzab (2002)

Grieger, M., Kotzab, H. *Supply chain Management beyond electronic marketplaces – insights from the chemical industry*. Proceedings of the 14th Annual Conference for Nordic Researchers in Logistics (Nofoma 2002). Trondheim, Norway, 2002.

Grønmo y Solheim (2004)

Grønmo, R., Solheim, I. *Towards Modeling Web Service Composition in UML*. The 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI-2004), 2004.

Haller y Schuldt (2003)

Haller, K., Schuldt, H. *Consistent Process Execution in Peer-to-Peer Information Systems*. Proceedings of the 15th Conference On Advanced Information Systems Engineering (CAiSE'2003). Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. 2681, pp. 289-307, 2003.

Hardaker y Graham (2000)

Hardaker, G., Graham, G. *Energizing your Supply Chain for e-Commerce*. Proceedings of the IMP Conference. Bath, UK, 2000.

Hofreiter y Huemer (2004a)

Hofreiter, B., Huemer, C. *ebXML Business Processes - Defined both in UMM and*

BPSS. Proceedings of the 1st GI-Workshop XML Interchange Formats for Business Process Management, Modellierung 2004, pp 81-102. Germany, 2004.

Hofreiter y Huemer (2004b)

Hofreiter, B., Huemer, C. *Transforming UMM Business Collaboration Models to BPEL*. International Workshop on Modeling Inter-Organizational Systems (MIOS), 2004.

Holmström y otros (2002)

Holmström, J., Främling, K., Kaipia, R., Saranen, U. *Collaborative planning forecasting and replenishment: new solutions needed for mass collaboration*. Supply Chain Management: An International Journal, 7, pp 136-145, 2002.

Huget y Odell (2004)

Huget, M.P., Odell, J. *Representing Agent Interaction Protocols with Agent UML*. 5th International Workshop of Agent-Oriented Software Engineering. LNCS 3382, pp 16-30. New York, USA, 2004.

IDEF (1995)

IDEF Family of Products. *IDEF 3 - Process Description Capture Method*. http://www.idef.com/. 1995.

Jablonsky y Bussler (1996)

Jablonski, S., Bussler, C. *Workflow Management: Modeling Concepts, Architecture and Implementation*. International Thompson Computer Press. London, UK, 1996.

Jennings (2001)

Jennings, N.R. *An Agent-based Approach for Building Complex Software Systems*. Communications of the ACM, 44 (4), pp 35-41, 2001.

Jung y otros (2004)

Jung, J., Hur, W., Kang, S., Kim, H. *Business Process Choreography for B2B Collaboration*. IEEE Internet Computing, Issue January/February 2004.

Kaipia y Saarinen (2001)

Kaipia, Saarinen, N. *Measuring the value of collaboration*. 2001.

Kalakota y Robinson (1999)

Kalakota, R., Robinson, M. *E-Business: Roadmap for Success* (First Edition). Addison-Wesley, 1999.

Kauremaa y otros (2004)

Kauremaa, J., Auramo, J., Tanskanen, K., Kärkkäinen, M. *The use of information technology in supply chains: transactions and information sharing perspectiva*. Proceedings of Logistics Research Network Annual Conference. Dublin, Ireland, 2004.

Kethers y Schoop (2000)

Kethers, S., Schoop, M. *Reassessment of the ActionWorkflow Approach: Empirical Results*. Proceedings of the Fifth International Workshop on the Language Action Perspective (LAP 2000). Aachen, Alemania, 2000.

Kleppe y otros (2003)

Kleppe, A., Warmer, J., Bast, W. *MDA Explained. The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.

Koehler y otros (2003)

Koehler, J., Hauser, R., Kapoor, S., Wu, F., Kumaran, S. *A Model-Driven Transformation Method*. Seventh International Enterprise Distributed Object Computing (EDOC 2003), 2003.

Kramler (2004)

Kramler, G. *Model Driven Development of Inter-organizationalWorkflows*. Tesis de Doctorado. Institute of Software Technology and Interactive Systems, Faculty of Informatics, Vienna University of Technology. Mayo, 2004.

Küster y otros (2004)

Küster, J.M., Sendall, S., Wahler, M. *Comparing Two Model Transformation Approaches*. Proceedings UML 2004 Workshop OCL and Model Driven Engineering. Lisbon, Portugal, October 12, 2004.

Lambert y Cooper (1998)

Lambert, D.M., Cooper, M.C. *Supply Chain Management: Implementation issues and research opportunities*. The International Journal of Logistics Management, 9, pp 1-19, 1998.

Lazcano y otros (2000)

Lazcano, A., Alonso, G., Schuldt, H., Schuler, C. *The WISE approach to Electronic Comerse*. International Journal of Computer Systems Science and Engineering, Special issue on Flexible Workflow Technology Driving the Networked Economy, 15 (5), pp 343-355, 2000.

Lee y otros (1997)

Lee, H., Padmanabhan, P., y Whang, S. *Information distortion in a supply chain: the bullwhip effect*. Management Science, 43, pp 546-558, 1997.

Lee y Wang (2000)

Lee, H., Wang, S. *Information sharing in the supply Caín*. International Journal of Manufacturing and Technology Management, 1, pp 79-93, 2000.

Lee y Wang (2001)

Lee, H., Wang, S. *E-Business and Supply Chain Integration*. Proceedings of Stanford Global Supply Chain Management Forum. Standford University, USA,

2001.

Leymann y Roller (2004)

Leymann, F., Roller, D. *Modeling Business Processes with BPEL4WS*. Workshop XML4BPM 2004 / XML Interchange Formats for Business Process Management. Marbug, Alemania, 2004.

Lind y Goldkuhl (2001)

Lind, M., Goldkuhl, G. *Generic Layered Patterns for Business Modelling*. Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling (LAP 2001), pp 21-22. Montreal, Canada, July, 2001.

Liu y Kumar (2003)

Liu, E., Kumar, A. *Leveraging Information Sharing to Increase Supply Chain Configurability*. Proceedings of The Twenty-Fourth International Conference on Information Systems, 2003.

Markus y otros (2003)

Markus, M.L., Axline, S., Edberg, D., Petrie, D. *The Future of Enterprise Integration: Strategic and Technical Issues in Systems Integration*. Competing in the Information Age: Align in the Sand (Second Edition). Ed. J. N. Luftman, Oxford: Oxford University Press, pp. 252-287, 2003.

McLaren y otros (2002)

McLaren, T.S., Head, M.M., Yuan, Y. *Supply Chain Collaboration Alternatives: Understanding the Expected Costs and Benefits*. Internet Research: Electronic Networking, Applications and Policy, 12, pp 348-364, 2002.

Medina-Mora y otros (1992)

Medina-Mora, R., Winograd, T., Flores, R., Flores, F. *The Action Workflow Approach to Workflow Management Technology*. Proceedings of the 4th Conference on Computer Supported Cooperative Work, ACM. New York, 1992.

Medjahed y otros (2003)

Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A. H. H.; y Ahmed, K.E. *Business-to-Business Interactions: Issues and Enabling Technologies*. The VLDB Journal, 12(1), Springer, pp 2041-2046, 2003.

Mellor y otros (2004)

Mellor, S.J., Scott, K., UHL, A., Weise, D. *MDA Distilled – Principles of Model-Driven Architecture*. Addison-Wesley, 2004.

Mens y otros (2004)

Mens, K., Czarnecki, Van Gorp, P. *A Taxonomy of Model Transformations*. Online Proc. of the Dagstuhl Seminar on Language Engineering for Model-Driven Software Development. J. Bézivin, R. Heckel (Eds.). Schloss Dagstuhl (Waden, D),

2004.

Mentzer y otros (2000)

Mentzer, J.T., Foggin, J.H., Golicic, S.L. *Collaboration: The Enablers, Impediments, and Benefits*. Supply Chain Management Review, 2000.

Mulani y Lee (2002)

Mulani, N.; y Lee, H. *New Business Models for Supply Chain Excellence*. Achievin Supply Chain Excellence through Technology, 4, pp 14-18, 2002.

OASIS (1999)

OASIS. *Electronic Business using eXchange Markup Language (ebXML)*. http://www.ebxml.org. 1999.

OASIS (2002)

OASIS ebXML CPP/A Technical Commitee. *Collaboration-Protocol Profile and Agreement Specification*, Versión 2.0. http://www.ebxml.org/specs/ebcpp-2.0.pdf. 2002.

OMG (2003a)

Object Management Group. *Model Driven Architecture (MDA) Guide*, Versión 1.0.1. http://www.omg.org/mda. 2003.

OMG (2003b)

Object Management Group. *UML 2.0 Superstructure Specification*. http://www.omg.org/docs/ptc/03-08-02.pdf. 2003.

OMG (2003c)

Object Management Group. *Meta Object Facility (MOF) Specification*, Versión 2.0. http://www.omg.org/docs/ptc/2003-10-04. 2003.

OMG (2003d)

Object Management Group. *XML Metadata Interchange (XMI) Specification*, Versión 2.0. http://www.omg.org/docs/ptc/2003-05-02. 2003.

OMG (2003e)

Object Management Group. *UML 2.0 OCL Specification*. http://www.omg.org/cgi-bin/doc?ptc/2003-10-14. 2003.

OMG (2003f)

Object Management Group. *MOF 2.0 query / views / transformations RFP*. OMG document ad/02-04-10. 2003

OMG (2004)

Object Management Group. *UML Profile for Enterprise Distributed Object Computing (EDOC)*. http://www.omg.org/technology/documents/formal/edoc.htm.

2004.

Peltz (2003)

Peltz, C. *Web services orchestration and Choreography – A look at WSCI and BPEL4WS*. Web Services Journal, Julio, 2003.

Petersen y Chapman (2000)

Petersen, M., Chapman, L. *Demand Activated Manufacturing Architecture (DAMA) - Supply Chain Collaboration Development Methodology*. Proceedings of International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2000). Temper, Arizona, USA, 2000.

QVTP (2003)

QVT-Partners. *MOF Query/Views/Transformations, Revised Submission*. OMG Document: ad/2003-08-08. 2003.

Raisinghani (2005)

Raisinghani, M. S. *E-Business Models in B2B: Process Based Categorization and Analysis of B2B Models*. International Journal of E-Business Research, 1, pp 16-36, 2005.

Rebstock y Thun (2003)

Rebstock, M., Thun, P. *Interactive Multi-Attribute Electronic Negotiations in the Supply Chain: Design Issues and an Application Prototype*. Proceedings of the 36th Hawaii International Conference on System Sciences, 2003.

RosettaNet (1999)

RosettaNet Consortium. Reference Web Site, http://www.rosettanet.org/RosettaNet/Rooms/DisplayPages/LayoutInitial, 1999.

Sahay y Gupta (2004)

Sahay, B.S., Gupta, A.K. *Supply Chain Management: The Critical Link to E-Commerce Success*. Emerging issues in Supply Chain Management (First Edition). Ed. B. Sahay, New Delhi, India: MacMillan India Limited, 2004.

Sahay y Maini (2004)

Sahay, B.S., Maini, A. *Supply Chain: A Shift form Transactional to Collaborative Relationships*. Emerging issues in Supply Chain Management (First Edition). Ed. B. Sahay, New Delhi, India: MacMillan India Limited, 2004.

Sahay y Mohan (2004)

Sahay, B., Mohan, R. *Emerging issues in Supply Chain Management*. Emerging issues in Supply Chain Management (First Edition). Ed. B. Sahay: McMillan India, 2004.

Schoop (2002)

Schoop, M. *Business Communication in Electronic Commerce*. The Language-Action Perspective on Communication Modeling (LAP 2002). 2002.

Searle (1975)

Searle, J.R. *A taxonomy of illocutionary acts*. Language, Mind and Knowledge. K. Gunderson (Eds.). University of Minnesota, USA, 1975.

Selic (2003)

Selic, Bran. *The Pragmatics of Model-Driven Development*. IEEE Software, Vol. 20, No. 5, pp 19-25, September 2003.

Sendall (2003)

Sendall, S. *Combining Generative and Graph Transformation Techniques for Model Transformation: An Effective Alliance?* OOPSLA '03 Workshop "Generative techniques in the context of MDA", 2003.

Sendall y Kozaczynski (2003)

Sendall, S., Kozaczynski, W. *Model Transformation – the Heart and Soul of Model-Driven Software Development*. IEEE Software, Vol. 20, No. 5, pp 42-45, 2003.

Simchi-Levi y otros (1999)

Simchi-Levi, D., Kaminsky, P., Simchi-Levi, E. *Designing and Managing the Supply Chain*. McGraw Hill, 1999.

Småros (2002)

Småros, J. *Collaborative forecasting in practice*. Proceedings of Logistics Research Network (LRN) 7th annual conference. Birmingham, UK, 2002.

Småros y Främling (2001)

Småros, J., Främling, K. *Peer-to-peer information systems - an enabler of collaborative planning, forecasting and replenishment*. Proceedings of Logistics Research Network (LRN) 6th Annual Conference. Edinburgh, UK, 2001.

Spekman y otros (1998)

Spekman, R.E., Kamauff, J.W., Myhr, N. *An empirical investigation into supply chain management: a perspective on partnerships*. Supply Chain Management: An International Journal, 3, pp 53-67, 1998.

Thöne y otros (2002)

Thöne, S., Depke, R., Engels, G. *Process-Oriented, Flexible Composition of Web Services with UML*. Proccedings of the International Workshop on Conceptual Modeling Approaches for e-Business: A Web Service Perspective (eCOMO 2002), Tampere, Finland, Springer LNCS 2784, 2002.

Tsalgatidou y Pilioura (2002)

Tsalgatidou, A., Pilioura, T. *An Overview of Standards and Related Technology in Web Services*. International Journal of Distributed and Parallel Databases, Special Issue on E-Services, 12(2), pp 135-162, September 2002.

Turban y otros (2003)

Turban, E., McLean, E., Wetherbe, J. *Information Technology for Management - Transforming Organizations in the Digital Economy* (4th Edition). Wiley and Sons, 2003.

UN/CEFACT (2001)

UN/CEFACT's Techniques and Methodologies Group (TMG). *UN/CEFACT Modelling Methodology* (UMM-N090 Revision 10). http://www.untmg.org/doc_bpwg.html. 2001.

UN/CEFACT y OASIS (2003)

UN/CEFACT y OASIS. *ebXML Business Specification Schema*, Version 1.10. http://www.untmg.org/downloads/General/approved/ebBPSS-v1pt10.zip. 2003.

van der Aalst (2000)

van der Aalst, W. M. P. *Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries*. Information and Management, 37(2), pp 67-75, 2000.

van der Aalst y otros (2003)

van der Aalst, W. M. P.; ter Hofstede, A. H. M.; Kiepuszewski, B.; Barro, A. P. *Workflow Patterns*. Distributed and Parallel Databases, 14(3), pp 5-51. July 2003.

Varro y otros (2002)

Varro, D., Varro, G., Pataricza, A. *Designing the automatic transformation of visual languages*. Science of Computer Programming, vol. 44(2): pp 205-227, 2002.

Verdicchio y Colombetti (2002)

Verdicchio, M., Colombetti, M. *Commitments for agent-based supply chain management*. ACM SIGecom Exchanges 3(1):13-23, 2002.

VICS (1998)

Voluntary Interindustry Commerce Standard (VICS). *Collaborative Planning, Forecasting and Replenishment*. www.cpfr.org, 1998.

VICS (2002)

Voluntary Interindustry Commerce Standard (VICS). *Collaborative Planning, Forecasting and Replenishment - Voluntary Guidelines*, V 2.0. http://www.vics.org/committees/cpfr/voluntary_v2/, 2002.

VICS (2004)

Voluntary Interindustry Commerce Standard (VICS). *CPFR - An Overview*. http://www.vics.org/committees/cpfr/CPFR_Overview_US-A4.pdf, 2004

Villarreal y otros (2001)

Villarreal, P., Caliusco, M.L., Salomone, E., Galli, M.R., Chiotti, O. *Support for Collaboration and Negotiation in SC Management*. IV Simposio de Administración de la Producción, Logística y Operaciones Internacionales. Conferencia de la Sociedad de Administración de Producción y Operaciones (SIMPOI 2001/POMS). Guarujá, Brasil, 2001.

Villarreal y otros (2003a)

Villarreal, P.D., Caliusco, M.L., Zucchini, D., Arredondo, F., Zanel, C., Galli, M. R., Chiotti, O. *Integrated Production Planning and Control in a Collaborative Partner-to-Partner Relationship*. Managing e-Business in the 21st Century. Ed. S. Sharma & J. Gupta. Heidelberg Press, pp. 91-110. Australia, 2003.

Villarreal y otros (2003b)

Villarreal, P., Salomone, E., Chiotti, O. *Managing Public Business Processes in B2B Relationships using B2B Interaction Protocols*. XXIX Conferencia Latinoamérica de Informática (CLEI 2003). Bolivia, 2003.

Villarreal y otros (2003c)

Villarreal, P., Caliusco, M.L., Galli, M.R., Salomone, E., Chiotti, O. *Decentralized Process Management for Inter-Enterprise Collaboration*. Vision, Special Issue on Supply Chain Management, Vol 7, pp. 69-79. Editor: B. S. Sahay, Management Development Institute, Gurgaon, India, 2003.

Villarreal y otros (2003d)

Villarreal, P., Salomone, E., Chiotti, O. *B2B Relationships: Defining Public Business Processes using Interaction Protocols*. Journal of the Chilean Society of Computer Science, Special issue on the Best Papers of the JCC 2003, Vol. 4(1), 2003.

Villarreal y otros (2004a)

Villarreal, P., Caliusco, M.L., Galli, M.R., Salomone, E., Chiotti, O. *Decentralized Process Management for Inter-Enterprise Collaboration*. Emerging issues on Supply Chain Management. Ed. B. Sahay. New Delhi, India: MacMillan India Limited, 2004.

Villarreal y otros (2004b)

Villarreal, P., Salomone, E., Chiotti, O. *A UML Profile for Modeling Collaborative Business Processes based on Interaction Protocols*. Argentine Symposium on Information Systems (ASIS 2004), 33 JAIIO. Argentina, 2004.

Villarreal y otros (2005)

Villarreal, P., Salomone, E, Chiotti, O. *Applying Model-Driven Development to Collaborative Business Processes*. Proceedings 8º Workshop Iberoamericano de

Ingeniería de Requisitos y Ambientes de Software (IDEAS´05). Valparaiso, Chile, 2005.

W3C (1997)

Word Wide Web Consortium. *Extensible Markup Language (XML)*. http://www.w3c.org/XML/Core/#Publications. 1997.

W3C (1999)

Word Wide Web Consortium. *XSL Transformations (XSLT)*, Versión 1.0. http://www.w3.org/TR/xslt, 1999.

W3C (1999b)

Word Wide Web Consortium. *XML Path Language (XPath), Version 1.0*. http://www.w3.org/TR/xpath, 1999.

W3C (2001)

Word Wide Web Consortium. *Web Services Description Language (WSDL)*, 1.1., W3C Note. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.

W3C (2002)

Word Wide Web Consortium. *Web Service Choreography Interface (WSCI)*, 1.0. http://www.w3.org/TR/wsci/, 2002.

Weigand y Heuvel (1998)

Weigand, H., y Heuvel, W. J. *Meta-Patterns for Electronic Commerce Transactions based on FLBC*. Hawaii International Conference on System Sciences (HICSS 98). IEEE Press, 1998.

Weigand y otros (1998)

Weigand, H., Heuvel, W., Dignum, F. *Modelling Electronic Commerce Transactions - A layered approach*. Proceedings of the Third International Workshop on the Language Action Perspective (LAP'98). Estocolmo, Suecia, 1998.

WfMC (1995)

Workflow Management Coalition. *The Workflow Reference Model, Technical report*. WfMC TC-1003. January 1995.

Willmott y Otros (2002)

Willmott, S., Calisti, M., Rollon, E. *Challenges in Large-Scale Open Agent Mediated Economies*. In proceedings of the 4th workshop on Agent Mediated Electronic Commerce (AMEC IV): Designing Mechanisms and Systems. pp. 325 – 340, Bologna, Italia, 2002.

Wooldridge y Otros (2000)

Wooldridge, M., Jennings, N., Kinny, D. *The Gaia Methodology for Agent-Oriented Analysis and Design*. Journal of Autonomous Agents and Multi-Agent Systems". Vol. 3(3), 2000.

Xu y otros (2001)

Xu, K.; Dong, Y.; y Evers, P. *Towards Better Coordination of the Supply Chain*. Transportation Research. Part E: Logistics and Transportation Review, 37, pp 35-54, 2001.

Yang y Papazoglou (2000)

Yang, J.; Papazoglou, M.P. *Communications of the ACM*. Vol. 43, No. 6, pp 39-47, 2000.