INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

MODELACIÓN DE INTERACCIONES MÚLTIPLES EN SISTEMAS DE INFORMACIÓN COOPERANTES.

FRANCISCO JOSÉ CAMARGO SANTACRUZ

SOMETIDO AL PROGRAMA DE GRADUADOS EN INFORMÁTICA Y COMPUTACIÓN EN CUMPLIMIENTO PARCIAL CON LOS REQUERIMIENTOS PARA OBTENER EL GRADO DE DOCTOR EN CIENCIAS COMPUTACIONALES

ASESORES:
DR. FERNANDO RAMOS QUINTANA
DR. JUAN FRAUSTO SOLÍS

CUERNAVACA, MORELOS A DICIEMBRE DE 2001.

MODELACIÓN DE INTERACCIONES MÚLTIPLES EN SISTEMAS DE INFORMACIÓN COOPERANTES.

Presentada por:
FRANCISCO JOSÉ CAMARGO SANTACRUZ

Aprobada por:

_____
Dr. Fernando Ramos Quintana
Director del Programa de Graduados en Informática y Computación, Campus Cuernavaca
Asesor de Tesis

_____
Dr. Juan Frausto Solís
Profesor, Departamento de Ciencias Computacionales, Campus Cuernavaca
Asesor de Tesis

Resumen

La naturaleza dinámica de los ambientes de agentes cooperativos dificulta la tarea de modelar interacciones permanentes entre agentes, resaltando los problemas de ambigüedad y control en la representación del mecanismo de interacción. La dificultad de modelar se incrementa si más de dos agentes están involucrados en la interacción de manera simultánea. Este problema es uno de los desafíos más importantes dentro de la investigación en Sistemas Multiagentes Cooperantes (SMAC´s).

Los Sistemas de Información Cooperantes (SIC´s o CIS por sus siglas en Inglés, Cooperative Information Systems) pueden ser vistos como SMAC´s e integran diferentes tipos de sistemas de información para que trabajen cooperativamente por un objetivo común. Estos sistemas son considerados por su naturaleza como sistemas que presentan un comportamiento dinámico y por ende, uno de sus principales problemas es el referente al como modelar y controlar múltiples interacciones simultáneas entre los agentes, de una manera sencilla para el ingeniero de software. El problema anterior se acentúa debido a que los métodos utilizados por los ingenieros de software para modelar la interacción de los SIC´s son poco expresivos y más aun, cuando se presentan situaciones de interacción entre más de dos agentes de manera simultánea. Esto complica la labor de modelación de estos sistemas y hace difícil la comunicación entre modeladores y desarrolladores lo que trae como consecuencia altos costos de desarrollo.

La contribución principal de esta tesis se centra en un modelo basado en Redes de Petri Coloreadas (RPC o CPN por sus siglas en Inglés, Coloured Petri Nets) para modelar las interacciones múltiples simultáneas en Sistemas de Información Cooperantes de una manera expresiva. Este modelo contribuye a facilitar la representación de la dinámica del sistema y en la reducción de la dificultad asociada con la modelación de la misma.

El modelo integra principalmente: a) el ciclo básico de acción llamado "*Loop*", para representar las interacciones del sistema y modelar conversaciones en las organizaciones, b) las Redes de Petri Coloreadas para la especificación de las interacciones representadas en el *loop* y para la simulación del sistema, c) los actos comunicativos de la Fundación para Agentes Inteligentes Físicos (FIPA por sus siglas en Inglés, Foundation for Intelligent Physical Agents), incluidos en la especificación del lenguaje de comunicación para agentes. El modelo brinda ventajas en la representación y el razonamiento de los mecanismos de interacción modelados en SIC´s.

Para validar el modelo propuesto, se presentan dos aplicaciones de éste en los dominios de Negocios Electrónicos (e-business por sus siglas en Inglés, Electronic Business) y Centros de Contacto respectivamente, los cuales son ambiente dinámicos que requieren de herramientas adecuadas para representar y controlar múltiples interacciones de una manera expresiva.

Palabras Claves

Ingeniería de Software Orientada a Agentes, Sistemas de Información Cooperantes, Redes de Petri Coloreadas, Sistemas Multiagentes Cooperantes.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Un Sistema de Información Cooperante (SIC) es un conjunto de agentes, datos y procedimientos que trabajan en forma cooperativa para alcanzar un objetivo común, soportando actividades diarias en la organización. El Sistema de Información Cooperante lo representamos con la siguiente expresión: *SIC={A, Cg, Gk, I}*, donde *A* es el Conjunto de Agentes, *Cg* es el Objetivo Común, *Gk* es el Conocimiento Global conformado por los datos del sistema y finalmente *I* es el Conjunto de Interacciones del Sistema, derivado de los procedimientos de la organización.

En nuestro enfoque, un SIC lo modelaremos como un Sistema Multiagente Cooperante ya que los agentes intercambian información y trabajan conjuntamente para alcanzar un objetivo común.

1.1. Planteamiento del problema

El modelar interacciones en SIC´s ha sido tratado en la literatura por varios autores entre los que destacan: [Cost 99a], [El Fallah 99b], [Demazeau 98] y [Haddadi 98]. Los modelos propuestos por los autores son aplicables, de manera sencilla a la interacción entre dos agentes. El proceso de modelar y controlar múltiples interacciones simultáneas en un SIC es una tarea compleja y las herramientas actuales que utilizan los Ingenieros de Software no son las adecuadas para hacerlo de una manera expresiva y sencilla.

Además, por un lado, los enfoques comúnmente utilizados para modelar SIC´s son insuficientes para representar su comportamiento dinámico ya que no ofrecen formas de modelar los cambios de estado en el tiempo. Por esta razón es necesario utilizar modelos que consideran tanto la estructura como el comportamiento dinámico del SIC, lo que nos permitiría de una manera adecuada representarlos, debido a que los modelos estáticos presentan limitaciones al momento de modelar y dar seguimiento a las interacciones entre los agentes. Por otro lado, el utilizar métodos formales ayuda a reducir el riesgo de caer en la ambigüedad al representar y simular la dinámica del SIC [Frausto 2001].

Un tópico relevante en el estudio de los SIC´s es el relacionado con la modelación de la cooperación entre los agentes y en particular cuando ésta se da para alcanzar un objetivo común. Modelar situaciones de cooperación en SIC´s no es una tarea sencilla, por lo que se propone representar la misma en varios niveles de cooperación para poder obtener una visión global del comportamiento de los agentes, de una manera expresiva y en particular del modelo de interacción del sistema [Ramos 2001]. Este modelo de Cooperación en SIC´s a varios niveles se presentará a detalle en el capítulo 2.

El mecanismo de interacción es una pieza central para la cooperación en SIC´s, ya que es el puente entre los protocolos de comunicación y los mecanismos de coordinación de los agentes. El problema de interacción para los SIC´s está inmerso en un mundo dinámico por naturaleza, y por consecuencia, la modelación y el seguimiento del mismo es difícil de manejar y generalmente presenta problemas de ambigüedad y control, por lo que los modelos que actualmente se obtienen no son lo suficientemente entendibles o expresivos para los Ingenieros de Software, especialmente cuando se quiere representar múltiples interacciones simultáneas.

1.2. Objetivo

Con base en lo anterior, el objetivo central, de este trabajo es desarrollar un Modelo de Interacciones Múltiples en Sistemas de Información Cooperantes que tenga las siguientes características:

- Los modelos generados deben ser expresivos para el Ingeniero de Software.
- Debe facilitar el modelar y controlar múltiples interacciones simultáneas, de una manera sencilla.
- Debe tener un enfoque formal para reducir la ambigüedad de los modelos y permitir la simulación de la dinámica del SIC.
- Debe validarse en aplicaciones de mundos dinámicos.

Para alcanzar el objetivo propuesto se desarrolló, en esta tesis, un modelo de interacciones múltiples en SIC´s, el cual se presenta en el capítulo 4. Los puntos relevantes del modelo son:

- La caracterización del problema de los sistemas de información cooperantes y la generación de un modelo de cooperación a varios niveles.
- El facilitar la generación de modelos de interacciones múltiples simultáneas en SIC´s, que consideran la expresividad como su característica central.
- La representación y el control de interacciones múltiples utilizando un método formal, como las Redes de Petri Coloreadas, relativamente fácil de usar.
- La reducción de la dificultad asociada a la modelación de interacciones múltiples.
- La visualización de la simulación dinámica del modelo en una herramienta computacional.

- La utilización del Ciclo Básico de Acción, *Loop*, como mecanismo de interacción y de los actos de habla de FIPA [FIPA 2001] y [Flores 96] como lenguaje de comunicación.
- La selección de dos aplicaciones con suficientes instancias que permitiera validar el modelo de interacciones múltiples propuesto.
- La portabilidad de los productos generados entre diferentes equipos de ingenieros de software facilitando la comunicación y el análisis de los modelos de interacciones múltiples realizados con el modelo propuesto.

1.3. Alcance

El modelo propuesto permite representar situaciones múltiples de interacción en SIC´s, considerando la fase de negociación entre agentes como parte del mecanismo, pero sin estudiar el comportamiento de los agentes durante la misma, para alcanzar el objetivo del SIC.

En el mecanismo de interacción solamente se modeló en esta tesis la dinámica de la interacción y el conjunto de actos de habla y estados de la conversación, quedando una posible extensión del modelo para incorporar la lógica de negocio asociada a las interacciones que determinan la toma de decisiones y el flujo de la interacción.

Al modelar las situaciones de interacción en los SIC´s representamos de manera simple los agentes del sistema, sin restarle generalidad al modelo. La representación e implementación de los agentes del sistema, junto con el conocimiento global asociado al SIC y los objetivos locales de cada agente, son un área de trabajo en el dominio de la Ingeniería de Conocimiento, que enriquecería el modelo desarrollado en esta tesis.

Para validar la modelación de interacciones múltiples simultáneas, se seleccionaron los siguientes dominios de aplicación: a) *Negocios Electrónicos*, en un ambiente de Negocio a Negocio donde se requiere representar y controlar múltiples interacciones simultáneas entre compradores, proveedores y la plaza de mercado. Estas interacciones se dan a través de Internet, de manera distribuida y en un ambiente altamente dinámico [Camargo 2001]; b) *Centro de Contacto*, el cual es un concepto clave en las organizaciones actuales para ofrecer un punto único de atención a los usuarios del producto o servicio. En este ambiente se requiere modelar y controlar múltiples interacciones simultáneas entre los usuarios, el centro de contacto, los diferentes centros de proceso y los técnicos asociados a estos últimos. La coordinación entre las áreas participantes y el seguimiento a las peticiones de los usuarios son factores críticos en un enfoque de calidad y retención de clientes. Los tipos de SIC´s utilizados en el trabajo son aquellos que tienen una estructura Cliente-Proveedor.

1.4. Contribuciones

La contribución central de esta tesis es un modelo expresivo de interacciones múltiples simultáneas en sistemas de información cooperantes, bajo un marco formal. Se propone realizar la representación del modelo mediante la definición de varios niveles de cooperación entre agentes. Además, al representar varias interacciones simultáneas en el modelo propuesto, se obtiene una reducción en la dificultad de la modelación, cuya validación se realizó en dos casos de aplicación.

1.5. Organización de la tesis

El trabajo está organizado de la siguiente manera, en el capítulo 2 se revisan los conceptos de los Sistemas de Información Cooperantes, incluyendo otros trabajos relacionados con la modelación de la interacción en ellos. Se realiza una revisión del Ciclo Básico de Acción, *Loop*, [Flores 96] y del proceso de construcción de diagramas de interacción, además de la comparación entre los actos de habla de Flores [Flores 86], [Flores 93a] y los de FIPA [Fipa 2001].

En el capítulo 3 se revisan los conceptos de las Redes de Petri Coloreadas, así como otros trabajos relacionados con la modelación de la interacción utilizando diferentes métodos formales y clarificando los problemas más importantes que se identificaron. Se realiza una comparación basada en la dificultad asociada a la modelación de la interacción entre agentes utilizando las RPC con un trabajo que utiliza Redes de Petri Condición Evento (C/E Petri Nets).

El capítulo 4 ilustra a detalle el Modelo de Interacciones Múltiples en Sistemas de Información Cooperantes, presentando los diferentes pasos que lo conforman así como las herramientas que se utilizan.

En el capítulo 5, se presentan los resultados de la validación del modelo en las aplicaciones de Negocios Electrónicos y Centros de Contacto. Además se presentan los detalles fundamentales de la implementación del mecanismo de interacción en la herramienta Design/CPN.

Finalmente, las conclusiones, los trabajos futuros y la bibliografía utilizada son presentados.

El trabajo cuenta con una serie de anexos donde se presenta un Manual de Diagramación de Procesos, las Reglas de Diseño con RPC, la Notación del Modelo propuesto y finalmente las Declaraciones de las RPC para la simulación en Design/CPN.

A lo largo de este trabajo se encontrarán terminos en Inglés, los cuales no serán traducidos al español con el fin de evitar una interpretación diferente a la original.

Capítulo 2

Sistemas de Información Cooperantes

Un sistema de información lo definimos como un conjunto de personas, datos y procedimientos, los cuales funcionan en conjunto; estos tienen como objetivo común el apoyar las actividades de la organización [Senn 90]. Existen varios tipos de sistemas de información: estratégico, administrativo o táctico, de conocimiento y operacional, de acuerdo al nivel organizacional que está apoyando [Laudon 2000].

En el nivel operacional se tienen sistemas de información que monitorean las actividades elementales y transaccionales de la organización. A un nivel de conocimiento, se tienen sistemas de información que ayuden a integrar nuevo conocimiento al negocio y controlar el flujo de papeles. En el nivel de administración o táctico, los sistemas de información soportan el monitoreo, control, toma de decisiones y actividades administrativas de los mandos medios. Finalmente, en el nivel estratégico los sistemas de información soportan actividades de planeación a largo plazo de la alta dirección así como, sistemas ejecutivos para facilitar la toma de decisiones.

En las organizaciones actuales, se observa la necesidad de una generación de sistemas de información que se caracterizan por la flexibilidad para integrar usuarios, sistemas y datos, de manera tal que permita preservar la inversión realizada y habilite la cooperación entre ellos y los nuevos sistemas de información. Estos son los Sistemas de Información Cooperantes (SIC´s) [Papazoglou 98]. Éste tipo de sistemas jugarán un papel fundamental en la competitividad de las empresas por la facilidad para configurar, adaptar y ampliar el sistema y sus componentes para adaptarse a los ambientes abiertos y distribuidos [Wooldridge 2001].

2.1. Sistemas de Información Cooperantes basados en Sistemas Multiagentes Cooperantes

El enfoque cooperativo para la resolución de problemas consiste en la búsqueda de soluciones a través de un grupo de entidades distribuidas llamadas agentes. Un agente lo definimos como una entidad del dominio del problema, el cual tiene capacidad de tener un comportamiento distintivo e interactuar a través de un mecanismo de comunicación dentro

de un marco de referencia bien definido [Sycara 98], [Krovi 99] [Camargo 96]. Estos pueden ir desde simples elementos de procesamiento hasta entidades de gran complejidad que exhiben conducta racional [Tagg 97].

El conjunto de agentes que interactúan entre si para lograr un objetivo en común se conoce como un Sistema Multiagente Cooperante (SMAC en español o CMAS, por sus siglas en Inglés, Cooperative Multi-agent System) [Ramos 95]. Los SMAC´s son una alternativa viable para la implementación de los sistemas de información cooperantes, ya que en éstos los agentes trabajan conjuntamente para resolver un objetivo común, e interactúan entre ellos mediante un intercambio de información, como se hace en los SMAC´s.

Los trabajos en el área de sistemas de información cooperantes basados, en SMAC´s, giran alrededor del entendimiento de los diferentes componentes, como son los agentes y su comportamiento, y los mecanismos asociados a estos, entre los que se encuentra la comunicación, coordinación e interacción, que facilitan la cooperación de los mismos [Cost 99c]. Un modelo para el desarrollo de Sistemas de Información Cooperantes, que ayude al Ingeniero de Software a integrar los componentes y mecanismos, es un área de trabajo relevante.

Uno de los aspectos más relevantes en los sistemas de información cooperantes es el mecanismo de interacción (o flujo de trabajo) entre los agentes, el cual es necesario en las aplicaciones para lograr la cooperación, ya que define las relaciones de intercambio de información entre los agentes del sistema. Esto lo realiza a través de un conjunto de reglas de comportamiento, las cuales tienen asociadas acciones encaminadas a facilitar el intercambio de información de una manera estructurada. El mecanismo de interacción utiliza los sistemas de comunicación y es utilizado para la coordinación de acciones en el sistema de información.

Es natural encontrar en los SIC´s que un agente esté participando en más de una interacción de manera simultánea, esto quiere decir que el agente tiene diferentes estados y acciones asociadas de acuerdo a cada una de las conversaciones en las que se encuentra. Una parte fundamental del proceso de desarrollo es la representación de estas situaciones de interacción, de una manera sencilla y entendible. Para asegurar que la representación de las interacciones entre los agentes se lleve a cabo de la mejor manera posible, buscando obtener un producto de software que cumpla con las especificaciones y que no tendrá errores al momento de su implementación, se recomienda la utilización de métodos formales en el proceso de desarrollo.

2.2. Ingeniería de Software Orientada a Agentes

Los mundos dinámicos en los que se encuentran inmersos los SIC´s, hacen que éstos estén en constante evolución; esto implica que requieren ser modificados, adaptados o reemplazados en periodos de corto plazo [Cheikes 93], [Morrison 95]. Ejemplo de ello son los ambientes de negocios electrónicos, donde las necesidades cambiantes del mercado y de los clientes obligan a que el sistema esté sujeto a cambios constantes en las especificaciones y en el desarrollo del mismo.

Además, los Sistemas de Información Cooperantes se caracterizan por ser sistemas que requieren estar en producción en un lapso de tiempo menor que otro tipo de sistemas, por lo que el tiempo que se invierta en su desarrollo y el proceso que se siga debe ser menor que los que poseen las metodologías actuales [Hong 2001].

Estudios en ingeniería de software proponen dos líneas de acción para tratar con los requerimientos anteriormente nombrados: 1) desarrollar nuevas metodologías para modelar SIC´s o 2) modificar, evolucionar o reinventar las existentes para responder a la naturaleza dinámica de los SIC´s e incorporar conceptos que permitan modelar agentes y las relaciones entre ellos, así como los esquemas de cooperación, coordinación, interacción y finalmente comunicación [Deloach 2001].

La Ingeniería de Software Orientada a Agentes (ISOA o AOSE de sus siglas en Inglés, Agent-Oriented Software Engineering), intenta contribuir en disminuir la dificultad relacionada con la modelación de SIC´s basados en SMAC´s. Ésta considera dos aspectos fundamentales en el desarrollo de los sistemas de información:

1. El producto o software que se produce.
2. El proceso que se sigue para producir el software.

La ISOA busca que el software desarrollado bajo su enfoque sea correcto y que el proceso que se utilizó para desarrollarlo sea llevado de una manera efectiva y eficiente [Jennings 95]. La carencia actual de métodos en la Ingeniería de Software Orientada a Agentes que permita modelar SIC´s como SMAC es un tópico relevante de estudio [Huhns 98], por la necesidad del mercado de enfoques que permitan reducir los tiempos de desarrollo y aseguren la calidad y confiabilidad de los productos obtenidos.

Utilizando ISOA podemos modelar interacciones múltiples en SIC´s, con diferentes niveles de complejidad y expresividad, de acuerdo a las herramientas que se utilicen. Con el objetivo de generar modelos expresivos para los Ingenieros de Software de una manera sencilla y efectiva identificamos requerimientos en tres áreas [Ramos 2001]:

- La Modelación de los Sistemas de Información Cooperantes.
- La Modelación del Mecanismo de Interacción.
- El Proceso de Desarrollo de los Sistemas de Información Cooperantes.

La expresividad, para el Ingeniero de Software, en la modelación de interacciones múltiples en concebida como la facilidad de entender los modelos generados y de aplicar los métodos seleccionados de una manera sencilla.

Para modelar Sistemas de Información Cooperantes, identificamos la necesidad que el ingeniero de software pueda contar con una herramienta que posea una representación gráfica y una semántica bien definida. Además, la herramienta debe permitir representar estados y acciones asociadas a múltiples interacciones, incorporar el concepto de

concurrencia en la modelación y facilitar el análisis de los modelos generados. La herramienta debe permitir modelar sistemas grandes de una manera jerárquica.

Para modelar el mecanismo de interacción identificamos la necesidad de contar con herramientas que permitan representar y controlar la dinámica de múltiples interacciones simultáneas, donde se pueda manejar diferentes mensajes de acuerdo al estado en que se encuentre la interacción y al agente que esté participando en ella. A partir de la modelación de la dinámica asociada con las interacciones del sistema, debe facilitar la observación del comportamiento del sistema a partir de las interacciones, de los cambios de los estados en el tiempo y del intercambio de información entre los agentes.

Para facilitar el proceso de desarrollo, el ingeniero de software requiere una herramienta de modelación que le permita simular el comportamiento del sistema antes de su implementación para poder detectar conflictos, tales como los abrazos mortales. Esto se realiza a partir de la observación de la dinámica de múltiples interacciones. Además, para simplificar el proceso de mantenimiento del sistema necesita reutilizar tanto software como modelos y debe poder realizar adaptaciones al sistema por cambios en los requerimientos, de una manera sencilla y efectiva.

El mecanismo de interacción ha sido estudiado, por un lado, como parte del mecanismo de comunicación [Cohen 95], [Levesque 91], [Liñan 98], y por otro, con un estudio del mismo de manera individual [Cost 99a], [El Fallah 99a]. La revisión bibliográfica realizada señala que se ha intentado desarrollar metodologías para especificar la interacción en sistemas de información cooperantes, utilizando diversos métodos, como Redes de Petri Condición / Evento (C/E) [Demazeau 98], Redes de Petri Recursivas [El Fallah 96], Lógica de Primer Orden [Haddadi 98], Contrac Nets [Lemaître 99a], [Lemaître 99b], Redes de Interacción [Cordero 99] y Diagramas de Transición de Estados [Cohen 95], entre otros. Las herramientas utilizadas en estas formalizaciones tienen varias limitaciones, sobre todo al momento de querer representar de una forma expresiva y querer controlar de una manera sencilla, la dinámica de la interacción entre más de dos agentes de manera simultánea, por lo que se ve la necesidad de explorar un método diferente, que permita realizar ésta representación.

2.3. Modelo de Cooperación en Sistemas de Información Cooperantes

El desarrollo de un sistema de información cooperante es complejo debido a que el Ingeniero de Software debe representar los aspectos específicos de la aplicación, además de modelar las interacciones entre los agentes. En [Huhns 98] se propone la programación orientada a interacciones (POI o IOP de sus siglas en Inglés, interaction-oriented programming), la cual se enfoca en representar lo que pasa entre los agentes y no solamente en los cambios internos de los mismos.

[Huhns 98] clasifica la cooperación en tres niveles, del nivel más bajo de cooperación al más alto:

- Coordinación, que permite a los agentes operar en ambientes compartidos.
- Compromiso, que adiciona coherencia a las acciones de los agentes.
- Colaboración, que incluye niveles de conocimiento sobre los compromisos y las comunicaciones.

Comparando con el enfoque propuesto en este trabajo, [Huhns 98] no considera dos niveles importantes: Comunicación e Interacción. En la Figura 2.1 presentamos la clasificación propuesta, de acuerdo a los elementos de la cooperación, en que basamos este trabajo.

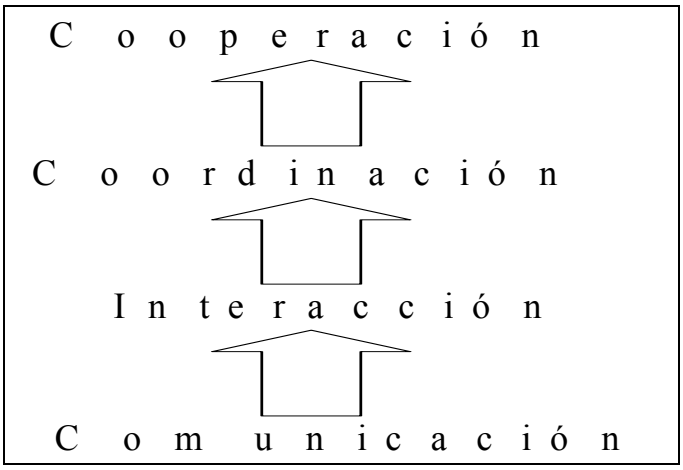


Figura 2.1: Clasificación de Acuerdo a los Elementos de la Cooperación.

En el nivel más alto de un sistema de información cooperante está la cooperación, la cual se genera en el mismo como consecuencia de los mecanismos que soportan las interacciones entre los agentes. Por cooperación entendemos la actitud de los agentes del sistema de interactuar coordinadamente intercambiando información para cumplir un objetivo común.

La coordinación es el proceso de manejar las interdependencias entre las actividades de los diferentes agentes, de tal manera que, a través del establecimiento de la secuencia correcta de acciones y la ejecución de las mismas, el sistema pueda alcanzar un objetivo determinado [Lesser 98a], [Lesser 98b], [Decker 95]. Con este proceso, los diferentes agentes del sistema interactúan de manera tal que las decisiones tomadas por uno de ellos no afecten a los otros ni al cumplimiento del objetivo común.

Analicemos cuando ocurre el problema de coordinación. Éste normalmente se da cuando un agente ha seleccionado una acción para alguna de las tareas que tiene asignada y esta selección afecta el desempeño o el orden en que las acciones son ejecutadas por otros agentes, o el momento en que las acciones son ejecutadas afectan el desempeño del sistema [Lesser 2001].

El problema es más complejo cuando el agente, al intentar realizar una selección de acciones, tiene una vista incompleta de la tarea en la cual sus acciones son sólo una parte, o la estructura de la tarea está cambiando dinámicamente o el agente tiene cierta incertidumbre acerca de la salida que puede producir la acción.

Es importante notar que cuando el agente se encuentra en cierta situación, o escenario, éste tiene potencialmente varias acciones a ejecutar, las cuales pueden estar interrelacionadas. El agente no existe aislado, éste se encuentra inmerso en un medio ambiente o conjunto de ambientes.

Una relación de coordinación existe cuando tenemos múltiples agentes en cierta situación o escenario, cada uno con su propia vista incompleta y con la posibilidad de cambiar la situación dentro del ambiente, además, con las acciones potenciales de un agente relacionadas con las de otro agente.

Por comunicación entre agentes entendemos el intercambio de información que produce un cambio de estado en el sistema. Ésta se puede dar al enviar y recibir información entre los agentes, a través del intercambio de mensajes.

La complejidad de la comunicación depende del nivel que se maneje: alto, medio o bajo, dependiendo ésta gama de los esquemas que los agentes utilicen, tales como primitivas de comunicación hasta aquellos que requieren comunicación muy sofisticada [Liñan 98].

Por comunicación de bajo nivel entendemos el uso de primitivas simples que permitan enviar y recibir mensajes, donde estas primitivas tienen interpretaciones previamente establecidas y fijas. En la comunicación de nivel medio, el lenguaje de comunicación se extiende para recibir nuevos tipos de mensajes, manteniendo la interpretación previa entre los agentes. En la comunicación de alto nivel se tiene en cuenta el entendimiento del lenguaje natural y las intenciones en comunicación de los agentes.

Un mecanismo de interacción debe proveer una descripción detallada de las posibles interacciones simultáneas entre más de dos agentes. Los estudios actuales han analizado la interacción desde un punto de vista de la teoría de los actos del habla más que en la teoría de los sistemas distribuidos, y es ahí donde se presenta un área de oportunidad para este trabajo de investigación.

Los mecanismos de interacción en el contexto de los sistemas multiagentes pueden tener dos enfoques, de acuerdo a los objetivos que los agentes tengan [Huhns 98].

En el caso de SMAC´s donde los agentes tengan objetivos en conflicto o simplemente sólo se interesan por su bienestar, el objetivo del mecanismo de interacción es el de maximizar las utilidades de los agentes.

En el caso de los SMAC´s donde los agentes tengan objetivos similares o problemas comunes, que es el tipo de sistemas que trataremos en el desarrollo de este trabajo de investigación, el objetivo del mecanismo de interacción es el de mantener un

comportamiento global coherente de los agentes sin violar la autonomía de los mismos y sin un control global explícito.

Para explicar el rol de la interacción en los SIC´s, se propone representarlo dentro de un marco general, el cual es ilustrado por el modelo presentado en la figura 2.2. En el modelo, se considera que el problema de la *Cooperación* entre agentes involucra diversos niveles: el nivel de *Comunicación*, el nivel de *Interacción* y el nivel de *Coordinación* [Ramos 2001].

El *Protocolo de Comunicación* facilita el intercambio de información entre los agentes del sistema, lo cual produce un cambio en el estado del mismo. El *Mecanismo de Interacción*, es un conjunto de reglas de comportamiento que definen el intercambio de información entre los agentes. El *Mecanismo de Coordinación*, establece una secuencia de ejecución de acciones de acuerdo a los objetivos individuales de los agentes y al objetivo común del SIC y finalmente, la *Cooperación*, la cual es el resultado de los mecanismos que soportan interacciones coordinadas entre los agentes.

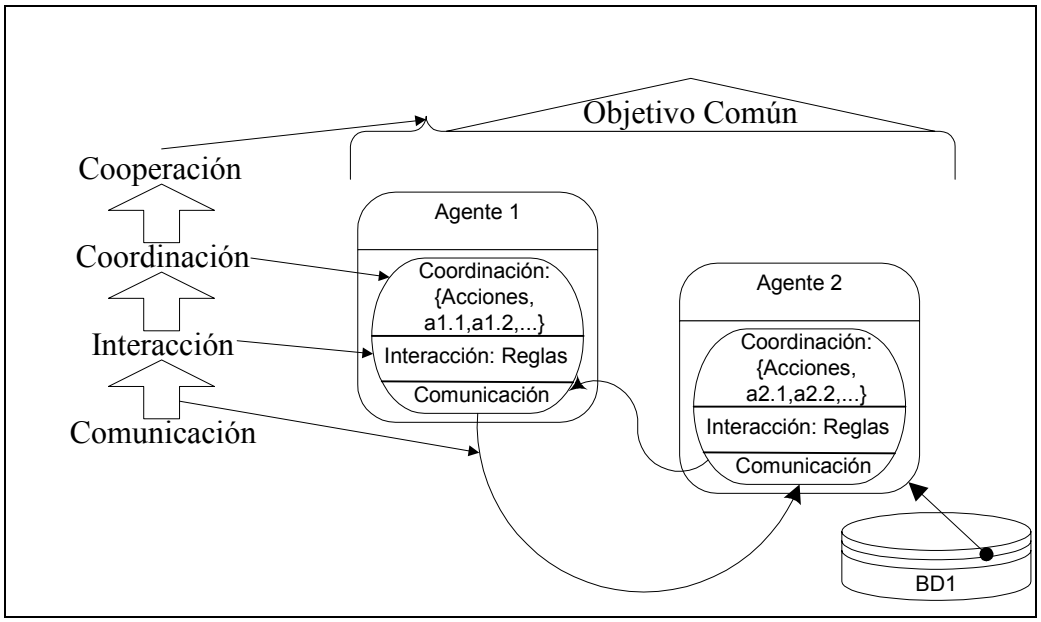


Figura 2.2: Marco de Referencia para la Cooperación en SIC.

En la figura 2.2, se presenta el modelo de una situación general de cooperación entre dos agentes de acuerdo con el marco de cooperación propuesto. El nivel de comunicación se representa con una serie de relaciones (ligas) entre agentes. La interacción se representa con un conjunto de reglas de comportamiento por cada agente. La coordinación se representa con un conjunto ordenado de acciones para cada uno de los agentes.

Es importante notar que el conjunto ordenado de acciones, en el nivel de coordinación, es un conjunto ordenado de mensajes de acuerdo a las reglas del nivel de interacción y a los mensajes del nivel de comunicación.

Por Cooperación entendemos el comportamiento de los agentes para realizar una interacción coordinada y un intercambio de información para alcanzar un objetivo común. La cooperación, coordinación y comunicación, son temas a los cuales haremos referencia dentro del tratamiento de la interacción en los SIC´s, la cual estudiaremos a mayor detalle.

2.4. El Ciclo Básico de Acción como Mecanismo de Interacción

Dentro del modelo de cooperación propuesto, el análisis del sistema está centrado en la modelación de las interacciones del SIC, donde a través del Diagrama de Interacciones determinamos quien se comunica con quien y bajo que relaciones.

Para modelar la interacción entre agentes se propone utilizar el Ciclo Básico de Acción conocido como *Loop* [Flores 96], mediante el cual se representan las conversaciones entre agentes, donde éstos juegan los roles de Cliente y Proveedor.

La acción humana tiene una estructura recurrente universal [Flores 93b]. El ciclo básico de acción capta la estructura y ofrece esta herramienta para diseñar acción efectiva con niveles de rigor, flexibilidad y eficiencia en las conversaciones entre seres humanos. Parte del enfoque del modelo propuesto en el capítulo 4, está en trasladar estos conceptos ya probados en las organizaciones para representar las interacciones entre agentes dentro de los sistemas de información cooperantes.

La acción ocurre entre los seres humanos en las conversaciones que dos personas sostienen, una llamada *Cliente* y otra llamada *Proveedor*, y que establece una relación de compromisos para cuidar en un dominio de preocupación mutuo y específico. Esta relación es llamada “Conversaciones para la Acción” [Flores 96], y son constitutivas del ser humano por su capacidad lingüística de hacer pedidos y ofertas. Es importante recalcar que se observa una acción efectiva cuando las fases del ciclo básico de acción han sido ejecutadas exitosamente.

El Ciclo Básico de Acción propone una ontología de actos comunicativos: *Request*, *Promise*, *Inform* y *Declare* (petición, promesa, afirmación y declaración). En la figura 2.3 podemos observar los Actos Comunicativos y su posición en el *Loop*, donde el Agente $_1$ inicia con una Petición (*Request*), el Agente $_2$ genera una Promesa (*Promise*), posteriormente el Agente $_2$ afirma (*Inform*) que ya terminó su trabajo y finalmente, el Agente $_1$ Declara (*Declare*) su satisfacción o no del trabajo realizado por el Agente $_2$ en la Conversación.

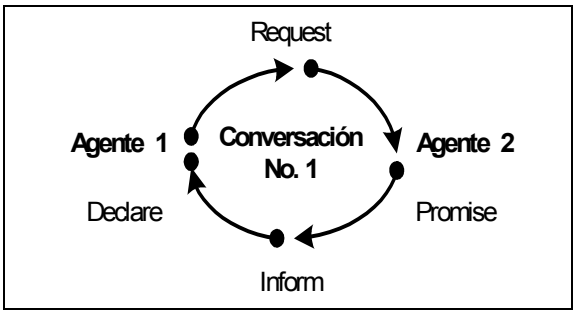


Figura 2.3: Actos Comunicativos en el *Loop.*

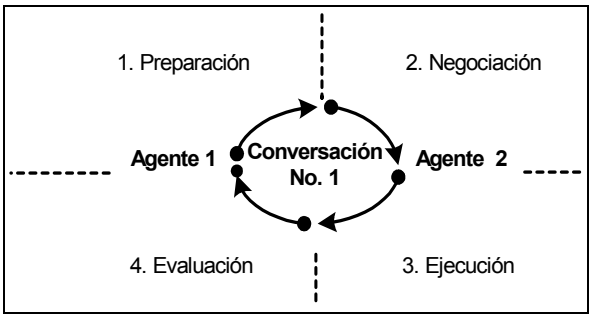


Figura 2.4: El Ciclo Básico de Acción, *Loop*.

El *Request* es el acto de habla utilizado para iniciar una conversación, e implica una petición del agente $_1$ al agente $_2$ cuando se utiliza dentro del *Loop*.

El *Promise* es el acto de habla utilizado por el agente que juega el rol de proveedor para confirmar que acepta la petición y que inicia los trabajos relacionados para satisfacer la misma.

El *Inform* es el acto de habla utilizado por el agente que juega el rol de proveedor para declarar que terminó los trabajos relacionados con la petición recibida y que hace entrega formal de los mismos.

Finalmente, el *Declare* es el acto de habla utilizado por el agente que juega el rol de cliente para declarar satisfacción o no del trabajo recibido por el agente proveedor, de acuerdo a la petición realizada y a la promesa recibida.

La figura 2.4 presenta el proceso completo del *Loop*, el cual está conformado por cuatro etapas o fases: Preparación, Negociación, Ejecución y Evaluación (*Preparation*, *Negotiation*, *Execution* y *Evaluation*, de sus originales en Inglés) [Flores 96], las cuales se describen a continuación.

Preparación (Preparation): El significado de esta etapa es diferente dependiendo si la conversación es un pedido o una oferta. En un pedido, antes de hacer la solicitud al proveedor, el cliente prepara el mismo de tal manera que cumpla con ciertas condiciones iniciales no negociables. En una oferta, el proveedor prepara el ofrecimiento que va a hacer al cliente, tomando en cuenta que el mismo contemple los intereses o preocupaciones del cliente que él conoce de antemano.

Negociación (Negotiation): Cuando el cliente ha realizado un pedido al proveedor, las partes deben negociar las condiciones de satisfacción con el objetivo que el compromiso formal o promesa por parte del proveedor, esté de acuerdo con lo que pide el cliente.

Ejecución (Execution): Una vez que el proveedor adquiere un compromiso con el cliente, la conversación está en ejecución, etapa en la cuál el proveedor está trabajando en cumplir la promesa y el cliente se encuentra esperando la entrega.

Aceptación, Evaluación (Evaluation): Al momento que el proveedor reporta término, o sea, declara haber cumplido sus compromisos, es conveniente que el cliente revise que el producto o servicio entregado cumpla con los requisitos o condiciones de satisfacción, que

habían acordado previamente. Aceptación es la etapa donde el cliente evalúa el cumplimiento del proveedor.

Una descripción práctica de como se interpreta el *Loop* y los actos comunicativos, de acuerdo a las figuras 2.3 y 2.4, es la siguiente: en el primer paso, el *Agente* $_1$ prepara la *petición* para el *Agente* $_2$ en una *conversación*. Posteriormente, el *Agente* $_1$ y el *Agente* $_2$ realizan una *negociación* acerca de la *petición*, y el *Agente* $_2$ da una *promesa* donde se establecen las condiciones de satisfacción. En el siguiente paso, el *Agente* $_2$ ejecuta la *promesa* y cuando termina la tarea, él le da un *informe* al *Agente* $_1$, declarando el termino de la misma. Finalmente, el *Agente* $_1$ *evalúa* el trabajo del *Agente* $_2$, de acuerdo a las condiciones de satisfacción establecidas, mediante una *declaración* de satisfacción o insatisfacción.

Muchas conversaciones son parte de una interacción, por lo que se necesita integrar las mismas mediante la utilización de un diagrama de interacción, como el que se presenta en la figura 2.5. En este diagrama se presentan dos conversaciones identificadas como *X* y *Y*.

La interpretación del diagrama es la siguiente: en la etapa de *Ejecución* de la conversación *X* entre los agentes $_1$ y $_2$, el agente $_2$ inicia la conversación *Y* con el agente $_3$, al termino de la cual el agente $_2$ concluye la etapa de *Ejecución* de la conversación *X*. La notación del diagrama y reglas para la construcción del mismo se encuentran en el Anexo A.

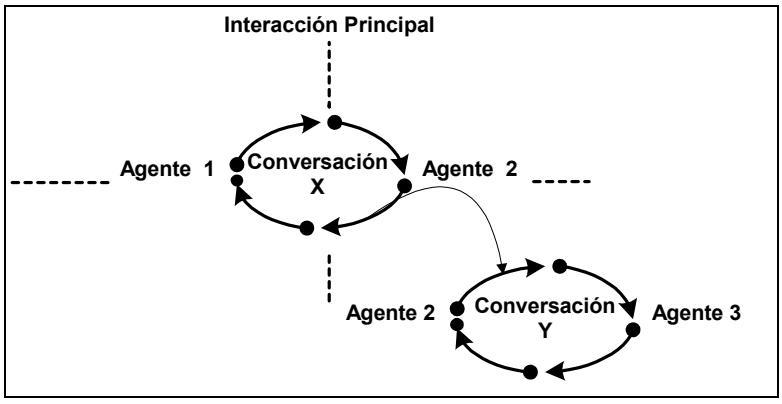


Figura 2.5: Diagrama de Interacción.

Los actos comunicativos de FIPA [Fipa 2001] son más expresivos que los actos básicos de Flores. Por ejemplo, la ontología de FIPA tiene diferentes actos de habla para expresar una Petición (*request*), mientras que en Flores solamente existe un acto. En la Tabla 2.1 se presenta una clasificación de los actos comunicativos de FIPA de acuerdo a los actos comunicativos básicos de Flores. Nuestra propuesta es utilizar los actos comunicativos de Flores para el ciclo básico de acción, pero explorar el enfoque de FIPA para mejorar la expresividad del modelo, ya que si solamente utilizamos los actos básicos de Flores, el lenguaje de comunicación pierde expresividad.

Tabla 2.1: Actos Comunicativos de FIPA y Flores.

FIPA	Comunicación para la Acción (Flores)			
Actos Comunicativos	**Request**	**Promise**	**Inform**	**Declare**
accept-proposal				X
Agree				X
Cancel			X	
Cfp (call for proposals)	X			
Confirm			X	
Disconfirm			X	
Failure			X	
Inform			X	
Inform-if (macro act)			X	
Inform-ref (macro act)			X	
not-understood			X	
propose		X		
query-if	X			
query-ref	X			
Refuse			X	
reject-proposal				X
request	X			
request-when	X			
Request-whenever	X			
subscribe	X			

El ciclo básico de acción facilita el representar una conversación coordinada entre los agentes, y permite integrar más de un agente simultáneamente, por lo que podemos realizar la modelación de interacciones múltiples, de una manera sencilla. Los trabajos de El Fallah et al. [El Fallah 98] y Cost et al. [Cost 99a], utilizan diferentes protocolos de interacción, lo cual incrementa la dificultad al momento de modelar e implementar los SIC´s. En estos trabajos una parte importante del esfuerzo se enfoca en que los agentes identifiquen qué protocolo de interacción se está utilizando, cuales son los diferentes estados asociados al mismo y cuales los actos de habla que se deben utilizar, todo esto con el objetivo de tener un conjunto de reglas de comportamiento que definan el intercambio de información en el SIC.

Una diferencia fundamental de la investigación con trabajos como los de [Cost 99b] y [El Fallah 99b] está en el tipo de sistemas de información que se modelan. Mientras el análisis en el trabajo propuesto se centra en la relación Cliente – Proveedor de los agentes del sistema, los trabajos de Cost y El Fallah no consideran estas relaciones y los sistemas modelados no necesariamente entran en esta caracterización.

En el modelo propuesto en esta tesis, se utiliza el Ciclo Básico de Acción como mecanismo de interacción. En este modelo el origen de toda conversación es siempre un interés, una *preocupación* o una necesidad que algún agente percibe al participar en una interacción dentro de la organización, y que para ser resuelto, requiere el apoyo de alguien más. [Malpica 2000].

El Ciclo Básico de Acción ha sido ampliamente aplicado como herramienta de modelación de procesos en las organizaciones pero no hay referencias previas, en la literatura

consultada, de trabajos que lo utilicen en sistemas de información cooperantes o en sistemas multiagentes cooperantes.

La conversación para la acción se da cuando ocurren los siguientes hechos entre dos personas [Flores 93a]:

- Una persona *declara* que falta algo, *declara* las condiciones bajo las cuales ella quedaría satisfecha y *pide* que la otra persona cumpla sus condiciones de satisfacción en un plazo determinado.
- La otra *promete* cumplir la petición (o no) en un cierto lapso de tiempo.
- En su momento, quien prometió *declara* haber cumplido el pedido.
- Quien hizo el pedido *declara* que sus condiciones de satisfacción han sido cumplidas y que la conversación está completa.

Durante la modelación de la interacción en SIC´s, se propone utilizar la técnica de conversación para la acción, donde los agentes juegan los roles de cliente y proveedor, de acuerdo a la interacción en que se encuentren. El uso de esta técnica es una aportación a la modelación de interacciones múltiples en SIC´s.

2.4.1. Los Componentes de una Conversación

Los elementos más importantes que componen una conversación son los Actores, las Condiciones de Satisfacción y el Lenguaje de Coordinación de Acciones. A continuación explicamos cada uno de éstos.

Actores

Son los agentes que forman parte de una conversación. Pueden ser clientes o proveedores. Cliente es el agente que espera que el proveedor realice alguna actividad, o algún trabajo que éste requiere, es quien tiene la preocupación. Proveedor es el agente del que el cliente espera algo, en cierto momento puede adquirir el compromiso de realizar un trabajo o ciertas actividades para el cliente.

En una conversación debe haber exactamente un cliente y un proveedor**.** La filosofía de conversaciones para la acción no permite que dos o más proveedores compartan un mismo compromiso simultáneamente con el mismo cliente, pues es fácil que el compromiso "pierda" su dueño. Es posible que haya varios proveedores asignados siempre y cuando cada uno de ellos tenga asignada la responsabilidad del compromiso en un momento dado. Siguiendo esta misma línea, no se puede que haya dos o más clientes compartiendo el mismo compromiso. Un proveedor y/o un cliente pueden participar en más de una conversación simultáneamente, a través de diferentes compromisos.

Condiciones de Satisfacción

Son los requisitos o condiciones que el proveedor debe cubrir para realizar un trabajo o actividad de forma satisfactoria para el cliente. Las condiciones de satisfacción pueden ser siempre las mismas en una conversación dentro de cierta interacción, o pueden variar dependiendo de las necesidades del cliente. Cuando esto ocurre, el cliente y el proveedor pueden negociar las condiciones.

Las condiciones de satisfacción, una vez que han sido aceptadas por ambas partes, se convierten en el objeto de una *promesa* que hace el proveedor al cliente. La *promesa* es sinónimo de compromiso con el cliente y cierra la etapa dos del ciclo básico de acción, donde se llega a un acuerdo.

Lenguaje de Coordinación de Acciones

Es un conjunto de declaraciones o aseveraciones que permiten al cliente y al proveedor comunicarse eficientemente en una conversación, de tal forma que no se pierda el enfoque en los compromisos. Esto nos permite tener un lenguaje uniforme y conocido por todos los agentes, además que se eliminan elementos del lenguaje que no producen acción.

Los actos de habla son la base de los compromisos lingüísticos, que forman el lenguaje de comunicación [Cohen 95]. Dentro de las conversaciones para la acción los agentes adquieren compromisos utilizando cuatro actos básicos: Pedir, Prometer, Afirmar y Declarar [Flores 93a].

Los actores (cliente o proveedor), mediante los actos de habla, son quienes propician que avance una conversación. El lenguaje, en su conjunto, está diseñado para permitir a los actores comunicarse en cualquier situación de negocios, por más excepcional que ésta parezca. Por situaciones excepcionales nos referimos a cancelaciones, renegociaciones, solicitudes de prórroga en el tiempo de entrega, solicitudes de estatus, entre otras.

Existen dos tipos de conversaciones: los pedidos y las ofertas.

Pedido: Es una conversación donde el cliente hace una solicitud o encargo al proveedor con la finalidad de cubrir una necesidad propia de su rol dentro de la interacción.

Oferta: Es aquélla conversación en la que el proveedor hace un ofrecimiento al cliente pensando en las necesidades del mismo. En éste tipo de conversación, es probable que el cliente no requiera el servicio y por tanto no lo tome.

En esta tesis, nos enfocamos a utilizar el *Pedido*, como mecanismo de interacción. El modelar la *Oferta* le permitiría al agente proveedor la capacidad de ser más pro-activo que re-activo, ya que no necesitaría recibir una petición para poder iniciar una conversación.

En una conversación se modela cómo los agentes dan seguimiento a los pedidos y a las ofertas en una relación eficiente y orientada al cumplimiento de los compromisos. Toda conversación sigue las cuatro etapas explicadas anteriormente: *Preparación*, *Negociación*, *Ejecución* y *Aceptación*.

2.4.2. El Diagrama de Interacción

Una interacción se representa en el modelo propuesto, como un conjunto de conversaciones relacionadas a través de un diagrama de interacción. Con esta herramienta representamos gráficamente las interacciones en los SIC´s.

Un diagrama de interacción no es un modelo de datos ni un diagrama de flujo de datos, lo que representa es la red de conversaciones orientada a compromisos que se crea entre los distintos agentes que participan en la interacción. Además, incorpora la forma de articular los compromisos, teniendo en mente que cada uno de ellos se dirige a atender intereses o a resolver necesidades y preocupaciones de algún agente participante.

Un diagrama de interacción es una guía o plantilla que nos describe cómo es más conveniente organizarse para resolver un proceso en particular y alcanzar un objetivo común [Flores 96]. Hay que recordar que todo proceso tiene una preocupación principal.

En el diagrama de interacción representamos la estructura de las diferentes interacciones del SIC, incorporando los agentes y sus conversaciones, además de la fase en la que participa. En la figura 2.6 observamos un ejemplo de un diagrama de interacción para una transacción en un ambiente de negocios electrónicos. En ella observamos que la interacción principal, *Transacción B2B No.1*, consta de cuatro conversaciones adicionales, *Establece Acuerdos*, *Pedir Propuestas*, *Ordenar*, *Esperar por la Entrega del Producto*. Además, podemos observar que en esta interacción participan tres tipos de agentes diferentes: *Comprador*, *Plaza de Mercado* y *Proveedor*.

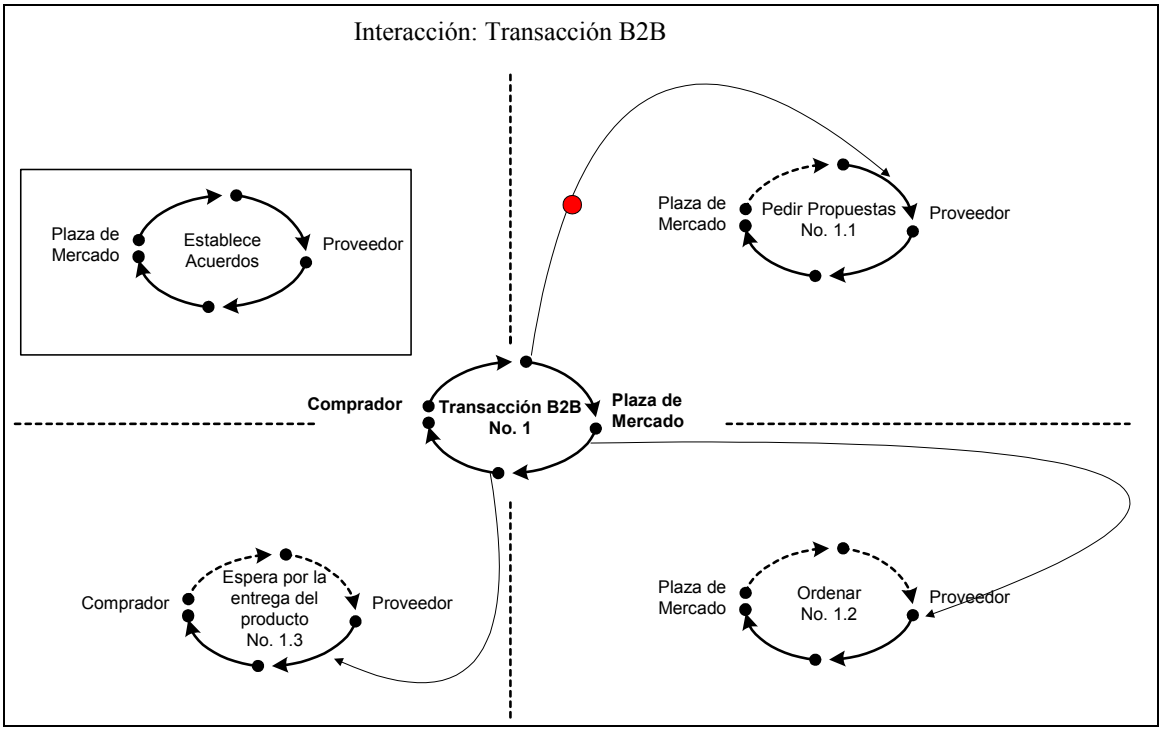


Figura 2.6: Ejemplo de un Diagrama de Interacción.

El diagrama de interacción inicia con la conversación principal *Transacción B2B*, entre los agentes *Comprador* y *Plaza de Mercado* y a partir de la misma y de acuerdo al proceso o interacción que se está modelando, se incluyen tantas conversaciones entre agentes como se necesiten. Cada conversación se debe situar en la etapa del ciclo básico de acción que corresponda.

En la figura 2.6 podemos observar la conversación principal marcada con el número 1, al centro de la cruz punteada que marca la separación de cada una de las etapas del *Loop*. En esta misma figura, podemos observar que la interacción total posee tres conversaciones, numeradas de acuerdo a la secuencia en que se interconectan entre ellas. La conversación encerrada en un rectángulo es llamada "conversación de contexto" y significa que debe darse antes de llegar a esta interacción, podemos decir que es un pre-requisito de la interacción. En el diagrama de interacción podemos observar como diferentes agentes se integran en un proceso del sistema de información cooperante para interactuar simultáneamente. La notación y guías de construcción del diagrama de interacción se presentan a detalle en el Anexo A.

Se exploró el como realizar la modelación del mecanismo de interacción en SIC´s por medio del Ciclo Básico de Acción, a traves de la utilización de los diagramas de interacción, para representar y controlar múltiples interacciones. De acuerdo al estado del arte realizado relacionado con la utilización de herramientas formales para modelar interacciones múltiples, a continuación se presenta la propuesta buscando una herramienta que permita representar de una manera expresiva y sencilla interacciones simultáneas.

Capítulo 3

Redes de Petri Coloreadas

El uso de diferentes métodos formales con el objetivo de modelar los mecanismos de interacción están presentes en la literatura. Estos métodos tienen algunas limitaciones al momento de modelar múltiples interacciones de una manera sencilla y expresiva, por lo que se observa la necesidad de explorar un método formal, como las Redes de Petri Coloreadas, y un esquema de representación como el Ciclo Básico de Acción para modelar el mecanismo de interacción.

3.1. Métodos Formales para Modelar Interacción

Entre los métodos formales encontrados en la literatura están: La Lógica de Primer Orden [Haddadi 98], los Diagramas de Transición de Estados [Cohen 95], [Flores 86], las Redes de Petri Condición / Evento (C/E) [Demazeau 98]. En las diferentes aplicaciones de estos métodos, la interacción usualmente aparece aislada, es decir, solamente se realiza entre dos agentes. Sin embargo, es común que en aplicaciones de SIC´s, los agentes estén involucrados en varias interacciones simultáneas por lo que el manejar interacciones múltiples resulta un problema complejo.

Entre las limitaciones de los métodos anteriores para representar múltiples interacciones se encuentran:

1. Son prácticos para especificar la estructura de la interacción cuando ella aparece en situaciones aisladas de comunicación, pero no se adaptan a modelar protocolos complejos con múltiples interacciones simultáneas en sistemas de información cooperantes, de manera sencilla.
2. Son complejos de manipular y modificar para los ingenieros de software cuando necesitan responder efectivamente a cambios en las especificaciones del sistema.
3. Se observan limitaciones para representar situaciones de interacción en SIC´s dada la naturaleza dinámica de los mismos, aunque son adecuados para representar sistemas estáticos.
4. La complejidad en la modelación de interacciones múltiples simultáneas se incrementa de manera exponencial cuando se realiza con estos métodos.

Al modelar interacciones múltiples en SIC´s, aparecen algunos problemas relevantes para los ingenieros de software que deben ser tratados con el objetivo de mejorar la efectividad en el proceso de desarrollo y la expresividad en la modelación. Algunos de estos problemas son los siguientes:

- La representación del estado de las interacciones simultáneas entre más de dos agentes.
- La representación de varios tipos de mensajes para diferentes agentes en diversos momentos del tiempo.
- La representación del comportamiento de los agentes en las interacciones de acuerdo al estado en que se encuentren, tanto ellos como la interacción.

Para tratar con los dos primeros puntos de los problemas anteriormente presentados, se exploró con Redes de Petri Coloreadas, buscando facilitar y optimizar el desarrollo de Sistemas de Información Cooperantes, de manera especial cuando éstas se encuentran incorporadas en un modelo de interacciones múltiples. Se comprobó, como presentaremos en el capítulo 4, que las RPC son una herramienta que agrega valor al Ingeniero de Software [Frausto 2001].

3.2. Redes de Petri Coloreadas: Definición y Notación

Una Red de Petri es un lenguaje formal y gráfico apropiado para modelar sistemas, complejos, que incorporen conceptos de concurrencia [Reisig 92]. Las redes de Petri han estado en desarrollo desde principios de los años sesenta, cuando Carl Adam Petri definió el lenguaje. Ésta fue la primera vez que se propuso una teoría general para sistemas discretos paralelos. El lenguaje es una generalización de la teoría de autómatas que permite expresar el concepto de concurrencia.

La Red de Petri es un grafo dirigido en el que existen dos tipos de nodos que corresponden a las plazas y transiciones, y además arcos que conectan a los nodos, con la restricción que no pueden ser del mismo tipo, es decir, los arcos no pueden conectar a dos plazas o a dos transiciones [Reisig 92].

La Red de Petri tienen los siguientes elementos básicos:

- **Plaza:** La cual es equivalente a un estado del sistema.
- **Transición:** Representa una acción en la Red de Petri que nos lleva de una plaza (o estado) a otro.
- **Nodos**: Los cuales son las plazas y las transiciones.
- **Arcos:** Es un conjunto de flechas dirigidas, las cuales conectan plazas con transiciones y viceversa pero no dos nodos del mismo tipo.
- **Nodo de entrada:** Un nodo *x* es llamado un nodo de entrada de otro nodo *y*, si y sólo si, existe un arco dirigido de *x* a *y*.

- **Nodo de salida:** Un nodo *x* es llamado un nodo de salida de otro nodo *y*, si y solo si, existe un arco dirigido de *y* a *x*. También se puede hablar de plazas de entrada y salida, transiciones de entrada y salida y arcos de entrada y salida.
- **Token:** Representa un valor en un momento dado del tiempo presente en una plaza o utilizado por una transición. Un token solamente puede estar en una plaza.
- **Marca:** Representa la distribución de tokens en la red en un momento del tiempo. Este conjunto representa el estado de la Red de Petri y se denota por M_x, donde x representa un número entero. Se representa por medio de un punto en la plaza o por medio de un número entero que indica el total de tokens en la plaza.
- **Marca Inicial:** Está dada por la distribución de los tokens en las plazas al inicio, y se denota *Mo*.
- **Habilitar la Transición:** Esto ocurre cuando en cada plaza de entrada de la transición se tienen los tokens necesarios y señalados en la expresión del arco de entrada correspondiente.
- **Disparo de la Transición:** Una transición ocurre, cuando ésta es habilitada; y su efecto se ve reflejado porque se quitan los tokens especificados de la plaza, de entrada y se adicionan los mismos a las plazas de salida.

La ejecución en una Red de Petri es controlada por el número y distribución de tokens que tiene. Los tokens presentes en las plazas controlan la ejecución de las transiciones de la red. Una Red de Petri se ejecuta por el disparo de transiciones y ésta a su vez puede dispararse si está habilitada. Para que una transición esté habilitada es necesario que cada una de sus plazas de entrada contenga al menos un token [Reisig 92].

Cuando una transición habilitada se dispara, se remueve un token de cada una de las plazas de entrada y luego se deposita un token dentro de cada una de las plazas de salida. El disparo de una transición cambiará el marcado M_0 de la red de Petri a un nuevo marcado M_1. El número de tokens en cada plaza nunca podrá ser negativo y el disparo de la transición no puede remover tokens inexistentes [Reisig 92].

Las redes de Petri las podemos clasificar en tres niveles [Christensen 2001]:

- Sistemas de Redes de Petri de nivel 1, los cuales son caracterizados por tokens booleanos, esto es, las plazas son marcadas por a lo más un token no estructurado. Las redes condición evento (C/E) son un ejemplo.
- Sistemas de Redes de Petri de nivel 2, los cuales son caracterizados por tokens que representan números enteros, esto es, las plazas son marcadas por muchos tokens no estructurados, que representan contadores. Las redes plaza/transición (P/T) son un ejemplo.
- Sistemas de Redes de Petri de nivel 3, los cuales son caracterizados por tokens de alto nivel, esto es, las plazas son marcadas por tokens estructurados que contienen información adicional en éste. Las redes de Petri coloreadas son un ejemplo.

Las Redes de Petri Coloreadas son Redes de Petri de alto nivel donde cada token tiene un color diferente y representa un tipo de datos arbitrario [Jensen 97a]. Por ejemplo, un token puede ser un registro donde el primer elemento es de tipo real, el segundo una cadena de

caracteres, el tercero una lista de enteros y así sucesivamente. Esta extensión incrementa el poder descriptivo para la modelación del sistema [Moldt 97].

El disparo de una transición se realiza dependiendo de la disponibilidad de un token del color apropiado en la plaza adecuada. Las Redes de Petri Coloreadas son un buen formalismo para describir concurrencia, sincronización y casualidad [Cordero 99], y son utilizadas para modelar, analizar y realizar prototipos de sistemas dinámicos con actividades paralelas [El Fallah 96], [El Fallah 99a] como los Sistemas de Información Cooperantes en nuestro enfoque.

Las Redes de Petri Coloreadas tienen los elementos básicos de las Redes de Petri, y adicionan algunos con el objetivo de ofrecer mayor expresividad y poder de representación. Los elementos se presentan a continuación y su representación gráfica está en la figura 3.1:

- **Plaza:** La cual es equivalente a un estado del sistema.
- **Transición:** Representa una acción en la Red de Petri que nos lleva de una plaza (o estado) a otro. Representa movimientos potenciales sobre la RPC.
- **Nodos**: Los cuales son las plazas y las transiciones.
- **Arcos:** Es un conjunto de flechas dirigidas, las cuales conectan plazas con transiciones y viceversa pero no dos nodos del mismo tipo.
- **Expresión del arco:** Es una expresión del lenguaje ML-CPN (extensión del lenguaje funcional SML). Las expresiones permiten representar situaciones complejas.
- **Nodo de entrada:** Un nodo *x* es llamado un nodo de entrada de otro nodo *y*, si y sólo si, existe un arco dirigido de *x* a *y*.
- **Nodo de salida:** Un nodo *x* es llamado un nodo de salida de otro nodo *y*, si y solo si, existe un arco dirigido de *y* a *x*. También se puede hablar de plazas de entrada y salida, transiciones de entrada y salida y arcos de entrada y salida.
- **Token:** Representa un valor en un momento dado del tiempo presente en una plaza o utilizado por una transición o una inscripción de la red. Un token solamente puede estar en una plaza.
- **Conjunto de color:** Es el valor de los datos asociados al token. Éste puede ser tan complejo como se defina. Para una plaza dada, todos los tokens deben tener un color asociado, que pertenece a un conjunto de colores de la plaza, el cual es el tipo de la misma. Por convención, los conjuntos de colores los ponemos en "itálicas". El uso de estos conjuntos de colores es análogo al uso de los tipos de datos en los lenguajes de programación. Los conjuntos de colores determinan los posibles valores de los tokens.
- **Marca:** Representa la distribución de tokens en la red en un momento del tiempo. Este conjunto representa el estado de la Red de Petri Coloreada y se denota por M_x, donde x representa un número entero. Se indica por medio de un círculo pequeño, con un entero indicando cuantos tokens hay y una expresión cerca del círculo, indicando por medio de un multiconjunto que representa los colores individuales de

los tokens y los coeficientes que ellos tienen. Por convención se omiten cuando una plaza no tiene tokens.

- **Marca Inicial:** Está dada por la evaluación de las expresiones de inicio que se encuentran distribuidas al principio en los tokens de las plazas, y se denota *Mo*. Por convención, se omiten las expresiones que resulten en el multiconjunto vacío.
- **Habilitar la Transición:** Esto ocurre cuando en cada plaza de entrada de la transición se tienen los tokens necesarios y señalados en la expresión del arco de entrada correspondiente.
- **Disparo de la Transición:** Una transición ocurre, cuando ésta es habilitada; y su efecto se ve reflejado porque se quitan los tokens especificados de la plaza, de entrada y se adicionan los mismos a las plazas de salida. El color de los tokens de entrada determina el color de los tokens de salida, lo cual es el efecto de la transición.
- **Declaraciones:** realizadas en el lenguaje CPN ML, que es una extensión del lenguaje funcional SML, para Redes de Petri Coloreadas. Especifican funciones, variables, conjuntos de colores, operaciones, constantes, expresiones, entre otras.

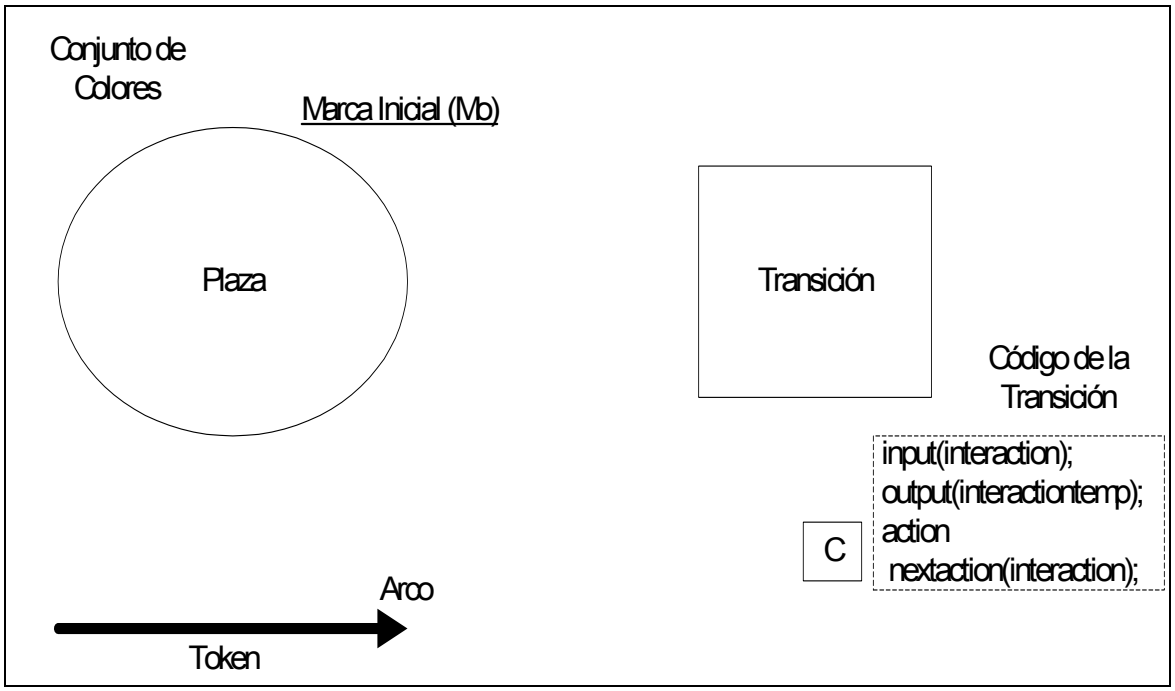


Figura 3.1: Notación Básica de las Redes de Petri Coloreadas.

Formalmente, en [Jensen 97b] se define una Red de Petri Coloreada como una tupla: *RPC= (Σ, P, T, A, N, C, G, E, I)* que satisface:

(i) Σ es un conjunto finito no vació de tipos, llamado conjunto de colores.
(ii) P es un conjunto finito de plazas.
(iii) T es un conjunto finito de transiciones.
(iv) A es un conjunto finito de arcos, tal que: $P \cap T = P \cap A = T \cap A = \emptyset$.
(v) N es una función del nodo. Ésta es definida de A hacía $PxT \cup TxP$.

(vi) C es una función de color. Ésta es definida de P hacía Σ.

(vii) G es una función de guarda. Ésta es definida de T hacía expresiones tales que: $\forall$ t$\in$ T: [Tipo(G(t)) = Booleano $\wedge$ Tipo(Variable(G(t))) $\subseteq \Sigma$].

(viii) E es una función de expresión de arco. Ésta es definida de A hacía expresiones tales que: $\forall$ a$\in$ A: [Tipo(E(a)) = $C(p(a))_{MS} \wedge$ Tipo(Variable(E(a))) $\subseteq \Sigma$], donde p(a) es una plaza de N(a). MS es un multiconjunto sobre C(p(a)).

(ix) I es la función de inicialización. Ésta es definida de P hacía expresiones cerradas, tales que: $\forall$ p$\in$ P: [Tipo(I(p)) = $C(p)_{MS}$].

Para entender el funcionamiento de las RPC a lo largo de este trabajo, se presentará a continuación un ejemplo basado en la aplicación de Negocios Electrónicos que se expondrá a detalle en el capítulo 5. En la figura 3.2 observamos una RPC con los siguientes elementos:

- **Plazas:** Se tienen tres plazas: *Preparation*, *Closed request* y *Negotiation*. Los colores asignados a cada plaza están indicados a un lado de la misma. En la figura 3.2 podemos observar que el color asignado a las tres plazas es *Tinteraction*.
- **Transiciones:** Se tienen dos transiciones: *c.cancel(preparation)* y *c.request*.
- **Guardas:** Indicadas antes de las transiciones. En la figura 3.2 hay dos, *[#nextaction interaction = C_CancelPreparation]* y *[#nextaction interaction = C_Request]*, las cuales solo permiten la activación de la transición si la condición que representan se cumple.
- **Expresión del arco:** Indicadas sobre los arcos que van de las plazas a las transiciones y de éstas a las plazas. En la figura 3.2 hay 4, *interaction*, *interaction*, *interaction* y *nextaction(interaction, anidex(4))*. La expresión *interaction* representa un token del color *Tinteraction* que fluye en el arco. La expresión n*extaction(interaction, anidex(4))* es un llamado a una función que recibe como parámetros una variable del tipo *Tinteraction* y el resultado del llamado de la función *anidex*.
- **Marca Inicial:** Indicada en la plaza *Preparation* por la expresión: *1'{buyer=[1], seller=[id=2,,n="marketplace"} ,supplier=[3,4], theproduct=[1,3], nextaction=C_Request}*, que corresponde a un token del color *Tinteraction*.
- **Declaraciones:** La descripción de las funciones y los colores asociados a la RPC se presentan en el anexo E.
- **Plaza de fusión:** Cada una de las plazas de fusión pertenecen a un **conjunto de fusión**, el cual está constituido por un conjunto de plazas que se comportan como una sola para efectos de la modelación de las RPC y la simulación de las mismas. Este mecanismo es útil para modelar sistemas jerárquicos de gran tamaño. En la figura 3.2 se tienen dos conjuntos de fusión indicados por las expresiones *FSClosed* y *FSNegoti* respectivamente. FSClosed agrupa todas las plazas que representen el estado *Closed request* mientras que *FSNegoti* agrupa todas las plazas que representen el estado *Negotiation*. Para indicar que es una plaza de fusión global se indica por

medio de la expresión *FG*. En [Jensen 97a] se encuentra una explicación amplia acerca de las plazas de fusión y el alcance de las mismas en las RPC.

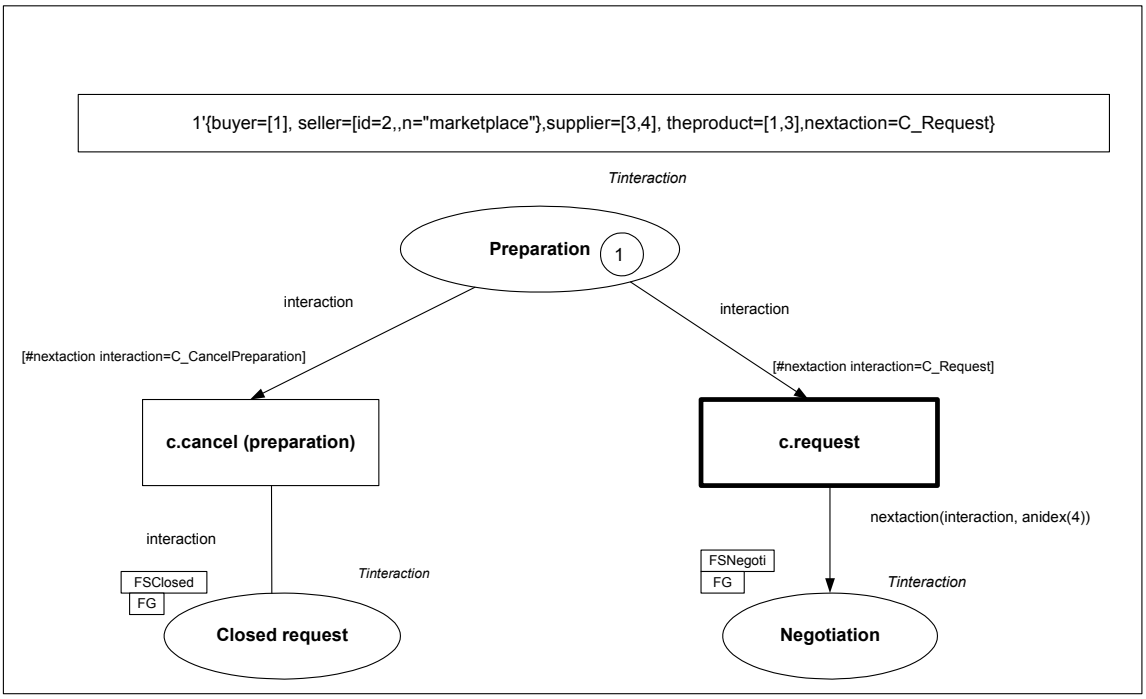


Figura 3.2: Ejemplo RPC, situación inicial.

En la figura 3.2 se represento la situación inicial de la RPC. Cuando se disparan las transiciones llegamos a la RPC presentada en la figura 3.3. A continuación explicaremos los elementos adicionales que permiten entender el funcionamiento de las RPC.

- **Habilitar la transición:** En los arcos que salen de la plaza *Preparation* se indica el número y tipo de tokens requeridos. En la figura 3.2 se observa la necesidad de un solo token del color *Tinteraction*, por lo que tanto las transiciones *c.cancel(preparation)* como *c.request* pueden ser habilitadas. Un elemento adicional que permite controlar que las transiciones se habiliten son las guardas. En la figura 3.2 se observa que la guarda *[#nextaction interaction = C_Request]* permite que sólo se habilite la transición *c.request*, ya que la guarda de la transición *c.cancel(preparation)* da como resultado de su evaluación el valor de falso. Es importante notar que para efectos de la evaluación la variable relacionada con el arco toma el valor del token que se encuentra en la plaza origen del mismo, en este caso de la plaza *Preparation*.
- **Disparo de la transición:** Ocurre cuando el token de la plaza *Preparation* es removido y utilizado por la transición habilitada, en este caso *c.request*, la cual a través de la expresión de arco relacionada n*extaction(interaction, anidex(4))*, produce un nuevo token en la plaza de salida correspondiente *Negotiation*. En la figura 3.3. observamos el resultado del disparo de la transición y su efecto en el token resultante, donde se observa un cambio en el campo *nextaction*, el cual del valor *C_Request* obtiene un nuevo valor *C_Agree*.

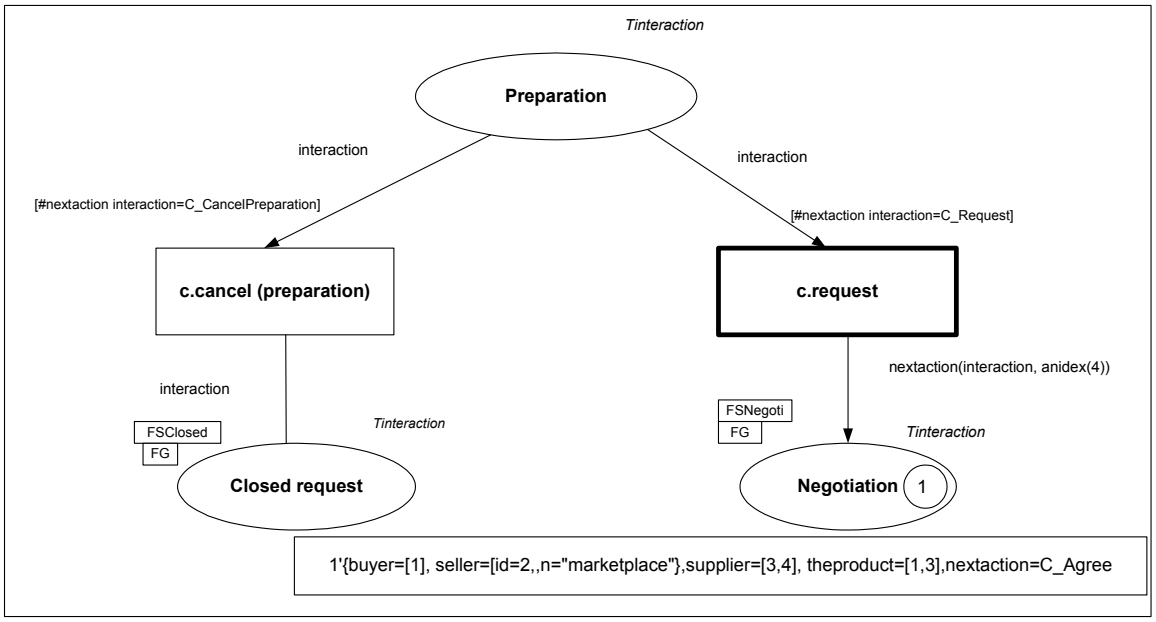


Figura 3.3: Ejemplo RPC, situación final.

Los modelos basados en Redes de Petri Coloreadas presentan ventajas con respecto a otros tipos de modelos de comportamiento, entre las que se encuentran [Jensen 97a]:

- El modelo creado utilizando una Red de Petri Coloreada, es una descripción del sistema que se está representando y puede utilizarse como una especificación formal de éste o como una presentación conceptual del mismo.
- Con base en la especificación del sistema, es posible analizar la dinámica del modelo realizado con la Red de Petri Coloreada, mediante una simulación de la misma.
- Los analistas de sistemas pueden mejorar su capacidad de entendimiento del sistema modelado, al observar el comportamiento del mismo.

Durante esta tesis, observamos que las Redes de Petri Coloreadas poseen características tales que permiten modelar aspectos dinámicos del sistema y en especial representar múltiples interacciones entre agentes de una manera simultánea. Entre las características se encuentran: el tener una representación gráfica y una semántica bien definida. Además el permitir modelar un sistema jerárquicamente, lo cual es apropiado para sistemas de información de gran tamaño. Ejemplos de la aplicación de las RPC se presentan en el capítulo 5.

Encontramos en este trabajo que al momento de modelar el mecanismo de interacción del SIC con Redes de Petri Coloreadas, nos permitió de manera sencilla y expresiva, representar la dinámica y el estado de la interacción o interacciones de más de dos agentes simultáneamente. Con esto, el ingeniero de software puede modelar sistemas de información cooperantes con interacciones más complejas que las actuales y con los beneficios que trae en el desarrollo de los mismos.

Como parte del modelo descrito en el capítulo 4, las Redes de Petri Coloreadas le facilitan al ingeniero de software simular los modelos generados en el análisis y diseño, con el objetivo de observar la dinámica de las interacciones múltiples y detectar posibles errores, como abrazos mortales (donde dos o más agentes no continúan interactuando debido a este problema), entre otros. Además que permite visualizar el estado del sistema en un momento dado, aportando al desarrollador del SIC una herramienta valiosa.

Con base en las razones anteriormente expuestas y en las características de las Redes de Petri Coloreadas, presentamos a continuación una propuesta para que los Ingenieros de Software utilicen la misma como herramienta de modelación, que puede ser adecuada, para realizar el análisis y el diseño de interacciones múltiples en sistemas de información cooperantes.

3.3. Redes de Petri Coloreadas para la Modelación de Interacciones Múltiples

Las propiedades de las Redes de Petri Coloreadas facilitan el modelar conversaciones e interacciones múltiples, así como observar el estado de las mismas cuando participen más de dos agentes. Además permite representar el comportamiento en la interacción de acuerdo al estado en que se encuentren los agentes y la interacción.

Con las RPC podemos modelar diferentes agentes que puedan intercambiar mensajes de diversos tipos de acuerdo al estado en que se encuentren. Una ventaja adicional es la herramienta computacional Design/CPN [Jensen 2001], que ayuda a simular la dinámica de las interacciones del sistema y permite encontrar problemas, tanto sintácticos como de diseño antes de implementar el sistema de información cooperante.

Consideramos que con el uso de las Redes de Petri Coloreadas representamos de manera sencilla y expresiva la dinámica y el estado de las interacciones simultáneas entre más de dos agentes, por lo que podemos representar interacciones más complejas en los sistemas de información cooperantes.

Para los ingenieros de software, éste es un método formal utilizable y que les permite observar la dinámica del sistema, lo cuál es un factor importante por el origen mismo de las interacciones.

Otros trabajos que modelan interacción utilizando Redes de Petri Coloreadas son El Fallah et al. [El Fallah 99a] y Cost et al. [Cost 99b]. El Fallah et al., se enfoca al estudio del diseño de sistemas Multiagentes combinando dos aspectos:

1. La Observación Distribuida para capturar las interacciones entre agentes, de par en par.
2. Las Redes de Petri Coloreadas, como el formalismo para identificar diseños orientados a la interacción.

Cost et al., se enfocan en la construcción de un lenguaje para especificar conversaciones, llamado *Protolingua* dentro del marco de trabajo del ambiente de desarrollo de agentes, *Jackal*, y propone utilizar Redes de Petri Coloreadas como el lenguaje para especificar conversaciones. Tanto El Fallah et al., como Cost et al., no tratan con el problema de la dificultad en la modelación de la interacción y la expresividad asociada a la misma, además que representan solamente las interacciones entre dos agentes.

Los trabajos de Cost y El Fallah se enfocan a sistemas donde los mecanismos de interacción son de diferentes tipos, por lo que los dominios de aplicación son diversos. El trabajo presentado solamente se validó en ambientes de aplicaciones cliente – proveedor utilizando como mecanismo de interacción el ciclo básico de acción [Flores 96] pero la contribución principal de modelar interacciones múltiples simultáneas en SIC´s utilizando RPC es valido en cualquier tipo de aplicaciones, como se presentó en [Camargo 2001], [Ramos 2001] y [Frausto 2001].

3.4. Análisis de Complejidad de los Modelos de Interacción

Un punto central en este trabajo está en como reducir la complejidad asociada en la modelación de múltiples interacciones entre agentes en los SIC´s. Por complejidad entendemos el número de recursos necesarios para realizar una actividad. Para fines de este trabajo la complejidad asociada al modelaje está relacionada con el número de ligas visibles al ingeniero de software que representen relaciones de interacción entre los agentes del SIC.

En la figura 3.4 se presenta el modelo utilizado por Demazeau et al. [Demazeau 98] con Redes de Petri Condición / Evento (C/E), donde tres agentes o entidades son representadas: *Entidad i-1*, *Entidad i*, *Entidad i+1*, y *m* mensajes, donde *i=1..n* representa el total de agentes en el sistema, y *j=1..m* representa el número total de mensajes en el sistema.

Este enfoque utiliza el concepto de "línea de mensaje", el cual es modelado por medio de una Red de Petri C/E, donde el *medio virtual* relaciona dos agentes capaces de intercambiar *m* diferentes clases de mensajes y consiste de *k* líneas como las que se presentan en la figura 3.4 y donde el *medio virtual* es representado por medio de la línea punteada.

En este modelo los agentes se representan por medio de Redes de Petri C/E. El nivel de detalle de las Redes de Petri C/E, tales como las plazas, las transiciones y los arcos o ligas que las unen son evidentes a los Ingenieros de Software en el *medio virtual*.

Vamos a realizar el análisis asociando la complejidad del modelo al número de arcos o ligas entre los agentes que representan interacciones a partir de las líneas de mensajes.

Por cada agente que se incorpore en el modelo de interacciones múltiples se requieren *2m* ligas para integrar el mismo con cada una de las líneas de mensaje mas *2m* ligas por *n-1* agentes. Si queremos incorporar más de un agente, el número de ligas necesario se

incrementa en *2m*((n+s)*((n+s)-1)/2)* mas *2m* ligas por *(n+s)-1* agentes, donde *s* es el número de agentes adicionales que participan en la interacción y *m* es el número total de mensajes. Este número de ligas es por cada uno de los agentes existentes en el sistema.

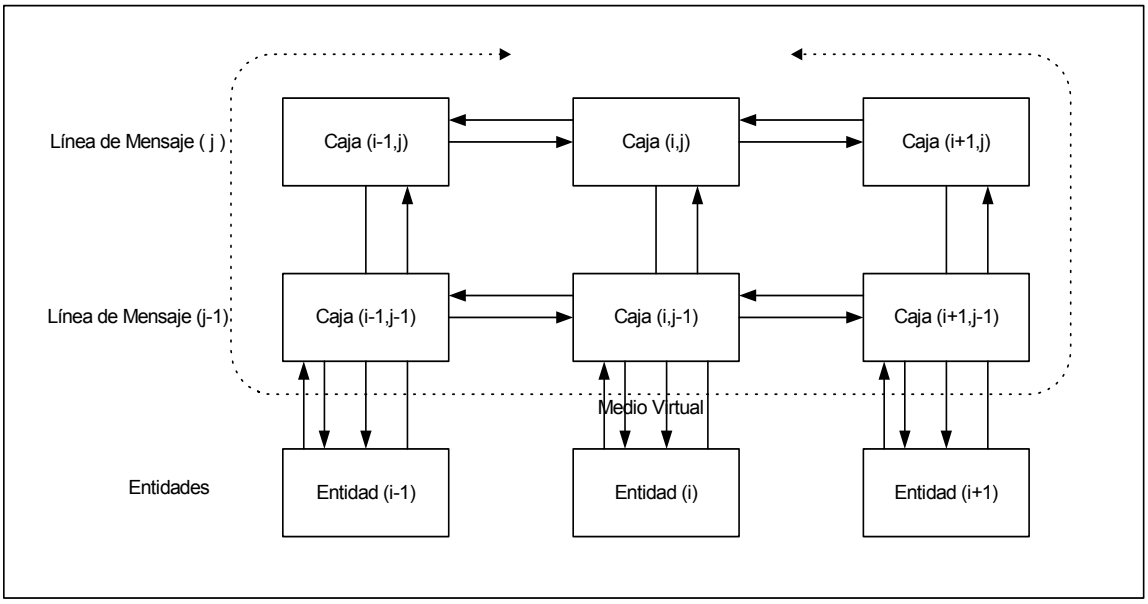


Figura 3.4: Modelo de Interacción de Demazeau.

Vamos a comparar el modelo anterior con una representación del mismo problema en el enfoque que se propone en esta tesis, el cual presentamos en la figura 3.5. La representación se realiza con Redes de Petri Coloreadas, donde observamos que cada mensaje se modela por medio de RPC que se encuentran inmersas en los agentes. La interacción entre los agentes se representa por medio de una relación o liga entre ellos. No existe un medio virtual análogo al presentado por Demazeau. Esto simplifica el modelo al ocultar los detalles de las RPC´s al Ingeniero de Software.

El incorporar un nuevo agente al SIC para que interactúe de manera simultánea implica modificar la representación del agente y sus mensajes asociados en las declaraciones de los colores de las RPC así como, incorporarlo conjuntamente con sus relaciones de interacción en el diagrama de agentes de la Figura 3.5.

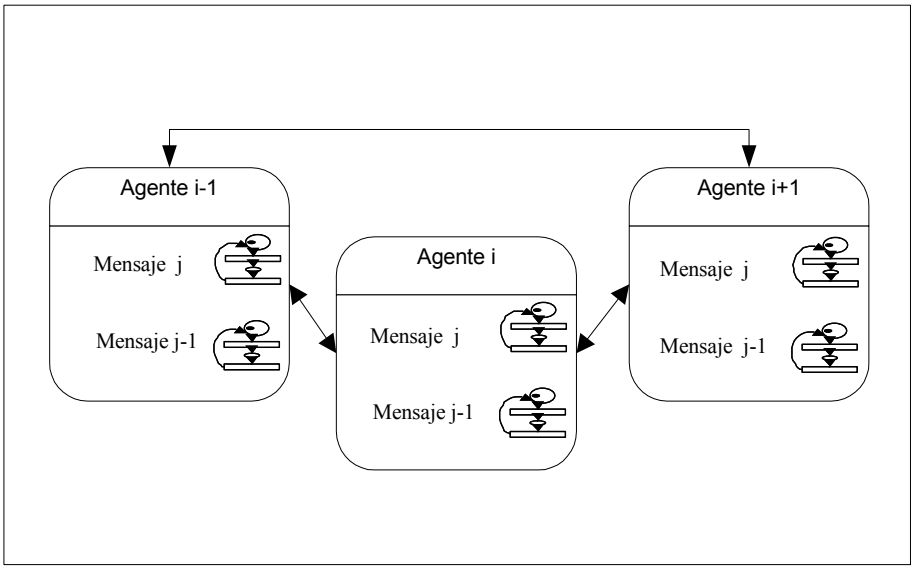


Figura 3.5: Modelo de Interacción Propuesto.

Si más de dos agentes están interactuando, el uso del medio virtual para modelar esta situación dificulta la representación. En cambio, al utilizar el enfoque basado en RPC, la dificultad se reduce considerablemente.

En Demazeau et al. se presenta un análisis de la complejidad del modelo asociada al número de ligas de las interacciones entre agentes, considerando:

- *m*: número de mensajes del sistema.
- *n*: número de agentes del sistema.

El número de ligas asociado al modelo presentado en la Figura 3.4 es *(2m*(2n-1))* [Demazeau 98].

En [Ramos 2001] se presenta un análisis de la complejidad del modelo propuesto utilizando RPC´s bajo los mismos parámetros de Demazeau et al., obteniendo un número de ligas asociado al modelo presentado en la figura 3.5 de *(n*(n-1))/2*, con un factor adicional, el número de ligas es independiente del número de mensajes. En la tabla 3.1 resumimos las diferencias fundamentales entre el enfoque propuesto y el enfoque de Demazeau et al.

Tabla 3.1: Comparación entre Demazeau y el Enfoque Propuesto.

Tópico	Modelo Propuesto	Modelo de Demazeau
Medio Virtual	No se modela explícitamente gracias al uso de técnicas de sistemas distribuidos, como los puertos, para representar la estructura del sistema.	Se modela con Redes de Petri C/E, dejando los detalles visibles al Ingeniero de Software.
Dependencia de Mensajes	El modelo es independiente	Es altamente dependiente el

	del número de mensajes intercambiados entre los agentes.	modelo del número de mensajes, lo cual incrementa la complejidad asociada.
Complejidad de Modelar Nuevas Interacciones Múltiples Simultáneas, relacionada al número de ligas visibles al ingeniero de software	*((n+s)*((n+s)-1))/2*, donde: *s* es el número de agentes adicionales que están participando en la interacción. *n* es el total de agentes del sistema	*(((n+s)*((n+s)-1))/2)* (2m*((2(n+s))-1)*, donde: *s* es el número de agentes adicionales que están participando en la interacción. *m* es el número total de mensajes. *n* es el total de agentes del sistema

Claramente, el modelo propuesto es adecuado para representar aplicaciones grandes, como los sistemas de información cooperantes, en contraste con la técnica de Demazeau et al., debido a la explosión combinatoria que presenta su método cuando se modelan interacciones múltiples simultáneas. Además, la expresividad del modelo está directamente relacionada con la dificultad del entendimiento del mismo, por lo que un modelo donde el número de ligas visibles al ingeniero de software es grande, no es lo suficientemente expresivo. De acuerdo a lo anteriormente presentado, los modelos generados al utilizar RPC poseen mayor facilidad de entendimiento que los modelados con Redes de Petri C/E.

Capítulo 4

Modelo de Interacciones Múltiples en Sistemas de Información Cooperantes

El problema de integrar herramientas de software para especificar sistemas complejos dinámicos, como los SIC´s, no ha sido tratada a profundidad, además los trabajos actuales realizados por diferentes autores trabajan sobre un modelo estático [Odell 2001].

Los SIC´s son complejos debido a su naturaleza dinámica y al manejo de múltiples interacciones simultáneas, y por ello la especificación de estos sistemas para el ingeniero de software es difícil de realizar. Construir software que satisfaga las necesidades antes mencionadas de una manera estructurada facilita la especificación de sistemas complejos.

4.1. Arquitectura

En la Figura 4.1 se muestra la arquitectura que integra los diferentes modelos propuestos para representar la interacción en SIC´s, la cual se basa en el análisis y diseño de interacciones, tomando en cuenta dos niveles de especificación del sistema:

- Especificación explícita, bajo un enfoque estático.
- Especificación implícita, bajo un enfoque dinámico.

La especificación explícita se realiza a partir de dos modelos: el *modelo individual* en el cual describimos cada uno de los agentes y el *modelo estructural*, en el cual describimos las interacciones entre los agentes.

La especificación implícita se representa dentro del *modelo dinámico*. En ésta se observa y controla la simulación de las interacciones del sistema.

Se construye el modelo de comportamiento del SIC a partir de los modelos *individual* y *estructural*. El modelo *individual* se construye a partir del conjunto de agentes del dominio del problema. El modelo *estructural* se construye a partir de las tareas, distribuidas en el tiempo y el espacio, que un SIC ejecuta y las cuales normalmente involucran flujos de

información entre agentes. El modelo *estructural* representa las relaciones entre agentes, llamadas en nuestro modelo *conversaciones*.

Finalmente, el modelo *dinámico* representa la dinámica de las interacciones y permite observar la evolución de las tareas del SIC.

La especificación explícita se simula en la herramienta Design/CPN. Las Redes de Petri Coloreadas se utilizan para representar tanto el modelo estático como el comportamiento dinámico de las interacciones del sistema.

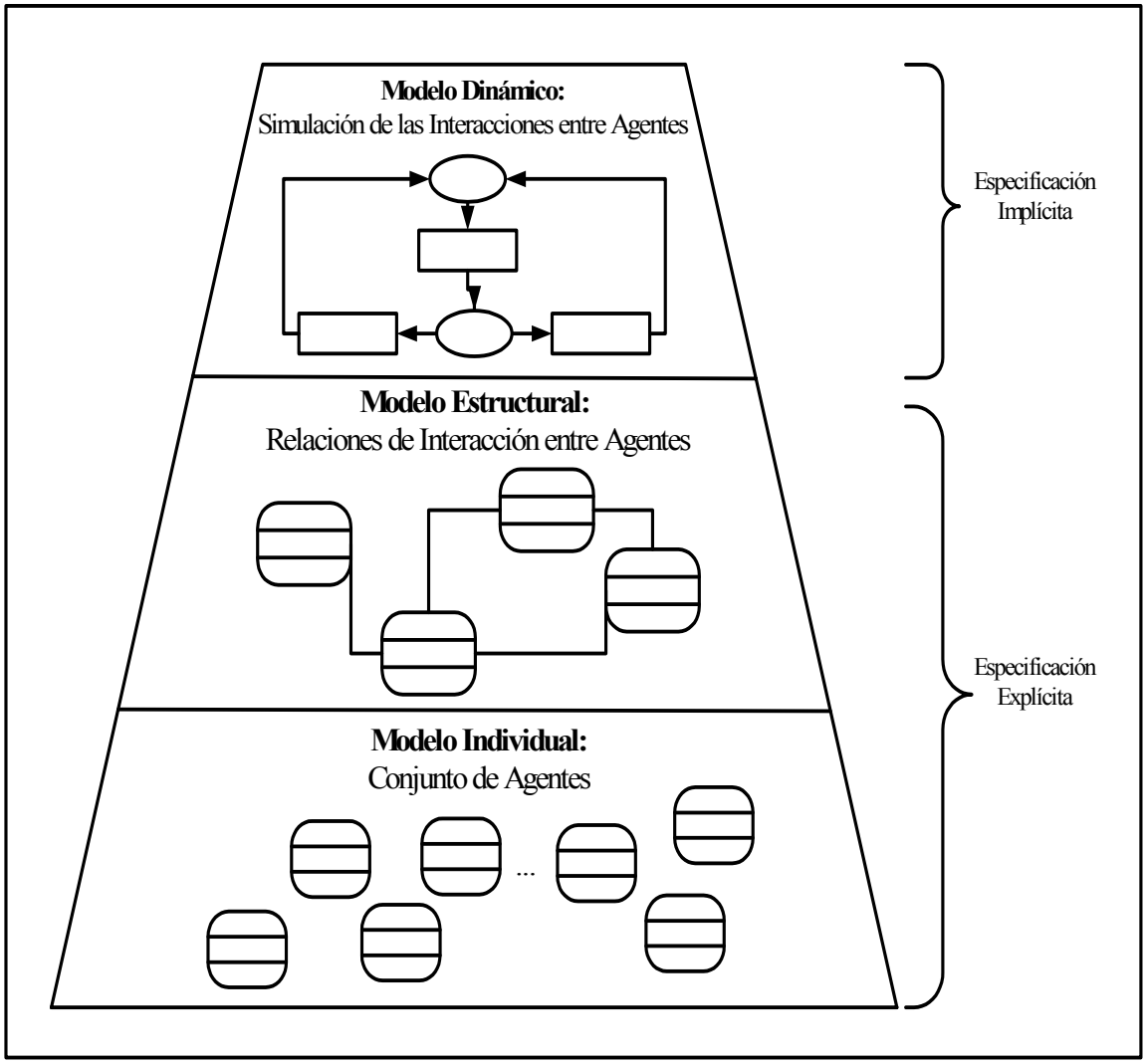


Figura 4.1: Arquitectura del Modelo.

En la figura 4.1 se observa que para pasar del *modelo individual* al *estructural* se utilizan los agentes identificados, se representan en el *loop* las interacciones en las que están participando y los mensajes que están intercambiando. Los diferentes *loop´s* construidos son mapeados a Redes de Petri Coloreadas.

Para pasar del *modelo estructural* al *dinámico*, se realiza una simulación de las RPC con el objetivo de observar la dinámica de las interacciones del sistema. La arquitectura está diseñada para que el ingeniero de software pueda pasar de manera recursiva e iterativa entre los diferentes modelos, con el objetivo de refinar cada uno de ellos.

El modelo se centra en el análisis y diseño de los sistemas de información cooperantes como un conjunto de "conversaciones" entre los diferentes agentes del sistema. Una conversación es una secuencia de mensajes entre dos agentes, que toma lugar en un período de tiempo y que tiene ciertas condiciones de terminación [Bradshaw 97]. Una conversación puede dar lugar a más conversaciones con diferentes agentes. Nótese que la conversación es más poderosa, desde el punto de vista de expresividad, en la modelación de sistemas de información cooperantes que lo que serían solamente los actos del habla tomados de una manera aislada.

En una conversación intervienen dos agentes, pero un agente puede estar en más de una conversación de manera simultánea, es decir, el agente debe saber en que etapa de cada una de las conversaciones en las que está participando se encuentra y cuál es la siguiente acción que debe ejecutar en la misma.

El conjunto de conversaciones modela un proceso dentro de un sistema de información cooperante. Los procesos están relacionados con actividades que se realizan en los sistemas, como por ejemplo, en un ambiente de negocios electrónicos, la venta de productos, la entrega de productos; En un ambiente de centro de contacto, la recepción de una petición, la asignación de un técnico a una orden de trabajo, entre otros. En este trabajo consideramos a la interacción como un proceso.

Los mensajes solamente ocurren dentro del contexto de una conversación, estos son parte del agente y su intercambio está regido por las reglas de la interacción. Una regla de interacción define, de acuerdo al estado de la conversación y al mensaje recibido, cuál es el acto de habla que debe utilizar. Por ejemplo, si el agente se encuentra en preparación y recibe un *request*, el puede seleccionar entre los siguientes actos de habla de acuerdo a las reglas de interacción: *Agree*, para aceptar la petición; *Decline*, para rechazar la petición; *Report completion with no agreement*, para terminar la petición; *Counteroffer*, para ofrecer otra opción.

Con el propósito de modelar interacciones múltiples en SIC´s, el modelo desarrollado en esta tesis integra principalmente tres elementos:

- En el ciclo Básico de Acción llamado "*Loop*", se representan las interacciones del sistema y se modelan las conversaciones en las organizaciones de una manera coordinada.
- En las Redes de Petri Coloreadas se mapean las interacciones representadas en el loop para observar la dinámica del comportamiento del sistema.
- En el modelo de las RPC se utilizan los actos comunicativos del FIPA para trabajar con un lenguaje estándar de comunicación para agentes.

El modelo de Interacciones Múltiples en Sistemas de Información Cooperantes [Ramos 2001], [Frausto 2001] está compuesto por los siguientes pasos:

Especificación Explícita:

1. Modelo Individual:
   1.1. Identificar agentes y sus intenciones.
2. Modelo Estructural:
   2.1. Construir el diagrama de agentes.
   2.2. Construir los diagramas de interacción.
   2.3. Diseñar los mensajes y puertos de los agentes.
   2.4. Representar el mecanismo de interacción utilizando Redes de Petri Coloreadas.

Especificación Implícita:

3. Modelo Dinámico:
   3.1. Simular la evolución de las interacciones del sistema.

4.2. Modelo Individual

El primer paso del modelo consiste en identificar los agentes del sistema y sus intenciones asociadas. Aquí se identifican las abstracciones clave en el espacio del problema, las cuales son candidatas a ser agentes, además de acuerdo al rol que juegan estas abstracciones en el dominio de la aplicación podemos identificar las intenciones asociadas a cada una de ellas.

Una abstracción clave es un agente que forma parte del vocabulario del dominio del problema. Por ejemplo, en el dominio de negocios electrónicos, los clientes y proveedores son candidatos a ser agentes. Técnicas utilizadas en la Ingeniería de Software Orientada a Objetos para identificar objetos pueden ser utilizadas en este paso, tales como las presentadas en [Booch 91].

El resultado a obtener es un conjunto de agentes *A*, de la forma:

- $A=\{(a_1,C_1), ..., (a_m,C_m)\}$, donde la dupla (a_j,C_j), representa al agente a_j con su conjunto de intenciones C_j, con $j=1..m$, donde m es el número total de agentes en el sistema.
- El conjunto de intenciones: $C_j = \bigcup_{i=1}^{r} cj_i$ donde cj_i es la intención i del conjunto C_j, con $i=1..r$, donde r es el número total de intenciones del agente a_j, y $j=1..m$, donde m es el número total de agentes en el sistema.

Se debe identificar el objetivo común del sistema, denotado como *Cg*, que es la razón por la cual los agentes trabajan cooperativamente en el sistema.

4.3. Modelo Estructural

El segundo paso del modelo consiste en construir el diagrama de agentes que represente al sistema de información. La estructura del modelo es el diagrama que muestra los diferentes agentes del sistema y las interacciones en las que estos participan. La figura 4.2 presenta los elementos del diagrama, como son los agentes en los nodos y las interacciones entre ellos en los arcos.

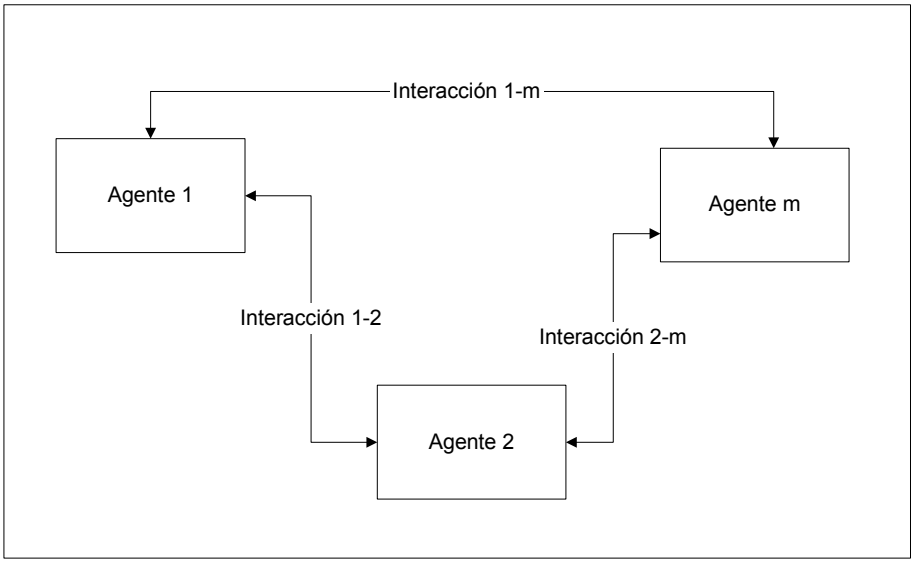


Figura 4.2: Diagrama de Agentes.

El diagrama de agentes del SIC es un grafo dirigido bi-direccional, representado por la dupla $DA=(A,R)$, donde se satisfacen los siguientes requerimientos:

(i) A es un conjunto finito de Agentes, $A=\{(a_1,C_1), ..., (a_m,C_m)\}$, donde la dupla (a_j,C_j), representa al agente a_j con su conjunto de intenciones C_j, con $j=1..m$, donde m es el número total de agentes en el sistema, como se explico anteriormente.

(ii) R es un conjunto finito de Relaciones de Interacción entre los agentes de A, representada por:

- $R=(r_1, r_2 ... r_n)$, donde r_k representa la relación de interacción k del sistema con:
  - $k=1..n$, donde n es el número total de relaciones de interacción en el SIC.
  - $r_k=(a_p, a_q, d_k)$, donde:
    - $a_p, a_q \in A, p \neq q, p=1..m, q=1..m, m=$ número de agentes del SIC.
    - d_k es una cadena de caracteres que representa el identificador asignado a la relación de interacción k.

La representación del grafo de la figura 4.2 es:

$DA=((Agente_1, Agente_2, ..., Agente_m), ((Agente_1, Agente_2, \text{“Interacción } _{1\text{-}2}\text{”}), (Agente_2, Agente_m, \text{“Interacción } _{2\text{-}m}\text{”}), ..., (Agente_1, Agente_m, \text{“Interacción } _{1\text{-}m}\text{”}))$.

Las relaciones de interacción se determinan en el sistema de acuerdo a las diferentes conversaciones en las cuales los agentes participan. A continuación se presenta la herramienta utilizada, con el objetivo de describir a detalle cada una de las interacciones y la forma como se enlazan las mismas para representar múltiples interacciones de una manera simultánea.

Construcción de los diagramas de interacción

El tercer paso del modelo consiste en describir las diferentes conversaciones del sistema, partiendo de la interacción principal. Este paso complementa y refina el diagrama de agentes, ya que con base en los resultados obtenidos aquí podemos redefinir interacciones en el diagrama.

Hay que recordar que la modelación de sistemas de información cooperantes es un proceso iterativo y recursivo, como un espiral, donde no seguimos un proceso secuencial, sino por el contrario, el finalizar cierto modelo en alguno de los pasos nos puede llevar a modificar modelos construidos anteriormente, así como nos sirven de entrada para la construcción de los modelos en los pasos subsecuentes.

Para construir los modelos de interacción, primeramente se debe identificar la interacción principal del sistema, que está alineada con el *Objetivo Común (Cg)* del sistema de información cooperante. La interacción principal es aquella que el ingeniero de software identifica y define como central para que los agentes trabajen de manera cooperativa con el fin de resolver un problema. Se debe tener en cuenta para definir la interacción principal cuál es la razón de ser del SIC.

Una interacción principal generalmente está conformada por una o más conversaciones entre agentes. En este paso debemos identificar las diferentes conversaciones que se llevan a cabo entre los agentes del sistema, que son relevantes para el dominio del problema que se está analizando, y que forman parte de la interacción central. Un sistema de información cooperante puede tener más de una interacción, pero es importante seleccionar una de ellas como la central y a partir de la misma derivar el conjunto de interacciones identificadas. Por ejemplo, en el ambiente de negocios electrónicos existen diversas interacciones como la entrega del producto, el levantar el pedido, el proceso de pago, pero la central es el realizar la transacción de compraventa, a partir de la cual se derivan las demás interacciones.

Al definir la interacción central, construimos el *loop* principal en el cual se relacionan los diferentes agentes y se derivan las conversaciones que forman parte de esta interacción, cada una de ellas representada por un *loop*. Para cada una de las conversaciones se determinan los agentes que participan en la misma y se construye el diagrama de interacción, de acuerdo a la notación que presentamos en la figura 4.3 y siguiendo la guía y las recomendaciones presentadas en el anexo A.

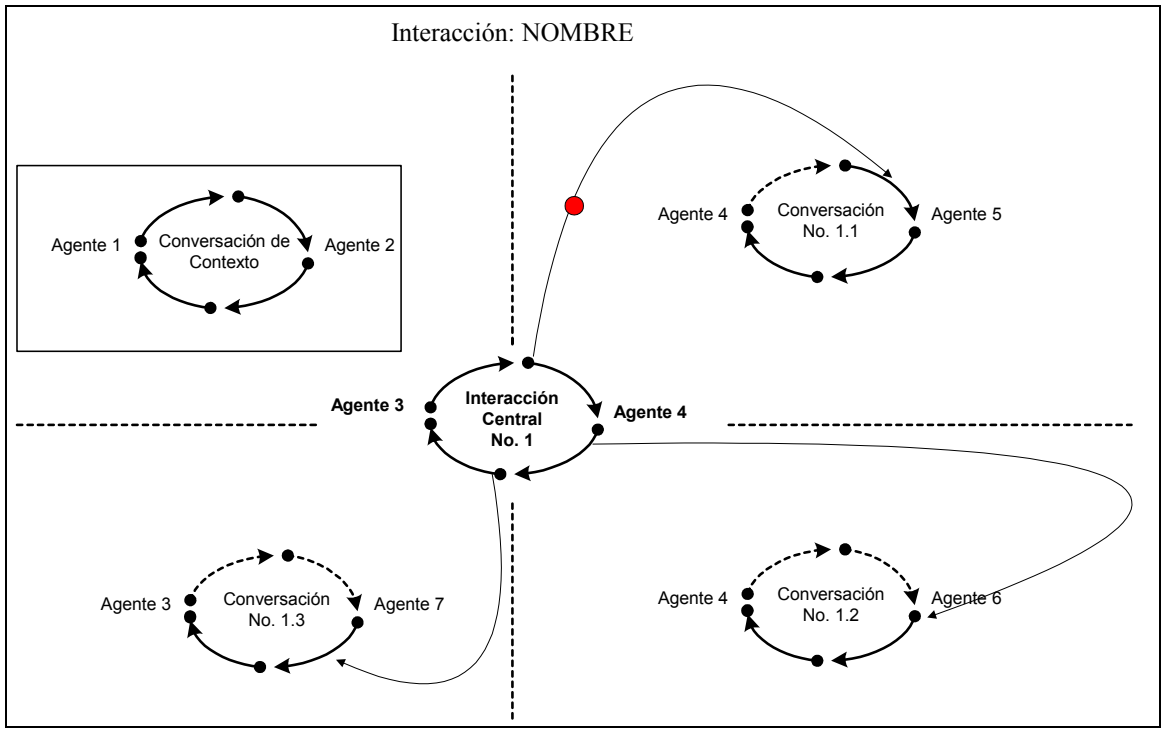


Figura 4.3: Diagrama de Interacción o Mapa de Proceso.

Para cada una de las conversaciones que forman parte de las interacciones del sistema, se debe llenar un conjunto de formas de especificación de servicio, como las que se presentan en las figuras 4.4 y 4.5.

ESPECIFICACIÓN DE SERVICIO: CONVERSACIÓN

Nombre de la Interacción:	ID de la Interacción:	Fecha:	Versió n
Nombre de la Conversación:		ID de la Conversación:	
Objetivo Principal de la Conversación:			
Cliente:	Proveedor:		

Figura 4.4: Descripción de la Conversación.

El Diagrama de Interacción *j* es un grafo D_j, representado por la dupla $D_j = (S_j, L_j)$, donde:

- $S_j = \bigcup_{k=1}^{r} sj_k$ *Sj* es el conjunto de conversaciones del diagrama de interacción *j*, con la conversación $sj_k = (a_p, a_q, dj_k)$, donde:
  - $a_p, a_q \in A, p \neq q, \ p=1..m, q=1..m, m=$ número de agentes del SIC.
  - $k=1..r$, donde *r* es el total de conversaciones en el diagrama de interacción *j*.

- dj_k es una cadena de caracteres que representa el identificador asignado a la conversación *k* en el diagrama de interacción *j*.

- $L_j = \bigcup_{k=1}^{v} lj_k$ L_j es el conjunto de ligas entre conversaciones del diagrama de interacción *j*, con la liga $lj_k = (sj_t, sj_s, e_1, e_2)$, donde:
  - $sj_t, sj_s \in Sj,\ t \neq s,\ t=1..r,\ s=1..r,\ r=$ total de conversaciones en el diagrama de interacción *j*.
  - $e_1, e_2 \in E,$ donde $E=\{$*"preparación", "negociación", "ejecución", "evaluación"*$\}$, conjunto de etapas del ciclo básico de acción.

El diagrama de interacción de la figura 4.3 queda representado con la siguiente expresión,
D={

{(Agente $_3$, Agente $_4$, "Interacción Central"), (Agente $_4$, Agente $_5$, "Conversación $_{1.1"}$), (Agente $_4$, Agente $_6$, "Conversación $_{1.2"}$), (Agente $_3$, Agente $_7$, "Conversación $_{1.3"}$), (Agente $_1$, Agente $_2$, "Conversación de Contexto")},
{((Agente $_3$, Agente $_4$, "Interacción Central"), (Agente $_4$, Agente $_5$, "Conversación $_{1.1"}$), "negociación", "negociación"), ((Agente $_3$, Agente $_4$, "Interacción Central"), (Agente $_4$, Agente $_6$, "Conversación $_{1.2"}$), "ejecución", "ejecución"), ((Agente $_3$, Agente $_4$, "Interacción Central"), (Agente $_3$, Agente $_7$, "Conversación $_{1.3"}$), "evaluación", "ejecución")}

}.

El conjunto de Interacciones del Sistema es representado por la expresión: donde: $I = \bigcup_{k=1}^{n} i_k$

- i_k, es el conjunto de diagramas de interacción para la interacción *k,* con $k=1..n$ y *n* el número total de interacciones en el SIC.
- El conjunto i_k se representa con la expresión: $i_k = \bigcup_{j=1}^{r} Dk_j$ donde:
  - Dk_j, es el diagrama de interacción *j* de i_k, con $j=1..r$ y *r* el número total de diagramas de interacción en el conjunto i_k.

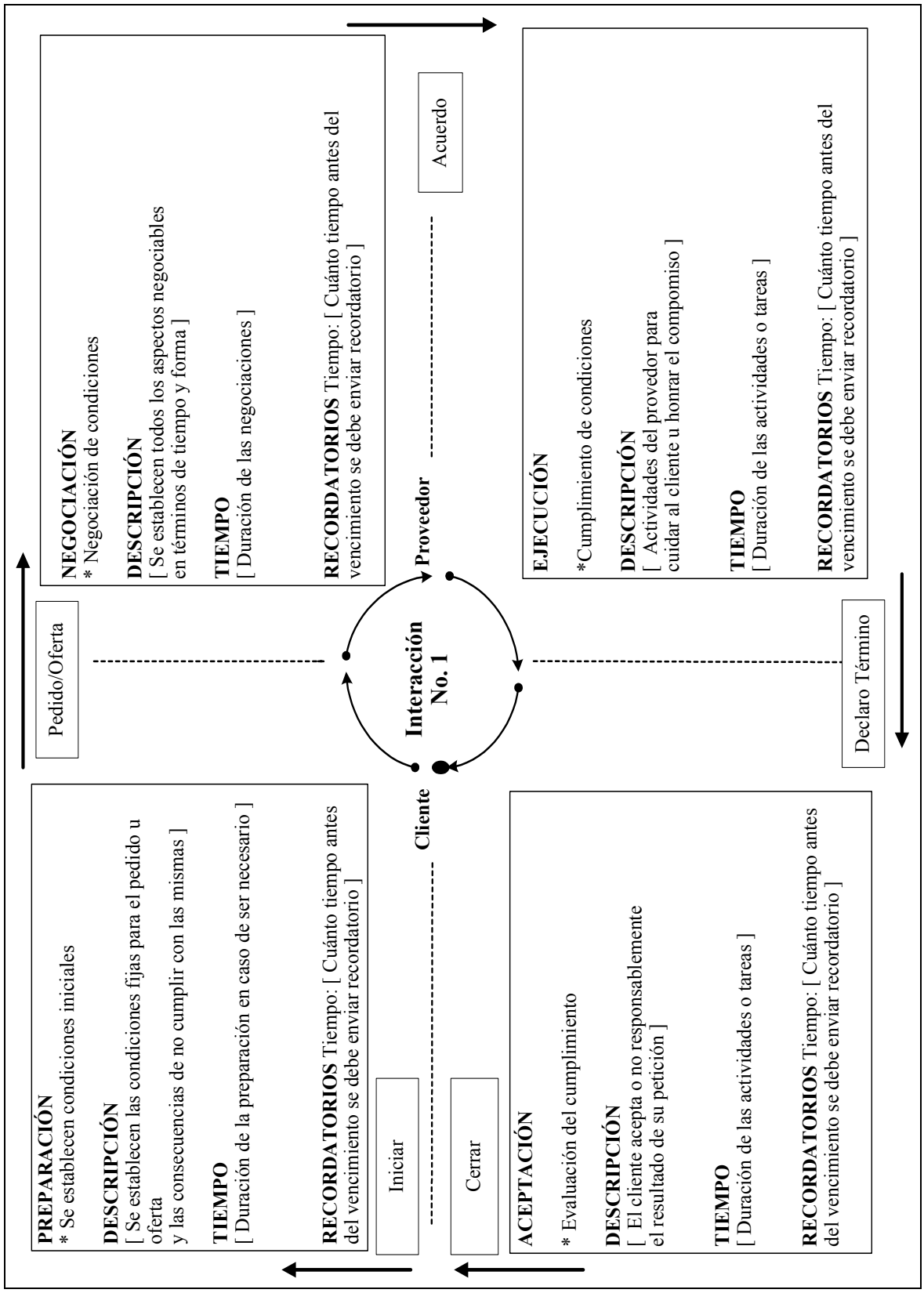


Figura 4.5: Especificación de la Conversación.

Al finalizar este paso, podemos representar el Sistema de Información Cooperante (SIC) por medio de la siguiente expresión: SIC={A, Cg, Gk, I}, donde:

- Conjunto de Agentes, $A=\{(a_1,C_1), ..., (a_m,C_m)\}$, donde la dupla (a_j,C_j), representa al agente a_j con su conjunto de intenciones C_j, con $j=1..m$, donde m es el número total de agentes en el sistema, como se explico anteriormente.
- Objetivo Común, Cg= Objetivo central del Sistema de Información Cooperante.
- Conocimiento Global, Gk={Conjunto de requerimientos de información, comúnes al SIC}.
- Conjunto de Interacciones, $I=\{i_1, i_2,...,i_t\}$, donde $i_r = \{D_{r1}, D_{r2},..., D_{rs}\}$, con D_{rk} el Diagrama de Interacción k de la interacción r, con $k=1..s$, donde s es el número total de diagramas de interacción para la interacción r y $r=1..t$, donde t es el total de interacciones del sistema, como se explico anteriormente.

Es importante resaltar que el Conocimiento Global (Gk, de sus siglas en Inglés Global Knowledge) se construye en las diferentes etapas y se utiliza el diccionario de datos como herramienta de apoyo. El formato del mismo se presenta en la figura 4.6.

ESPECIFICACIÓN DEL SERVICIO: DICCIONARIO DE DATOS

* Términos generalmente utilizados por el dueño del servicio

NOMBRE DEL SERVICIO	ID DE SERVICIO	FECHA	VERSIÓN

TERMINO	SIGNIFICADO

ANEXOS

* Sección para documentos adicionales que apoyan a explicar las definiciones.

Figura 4.6: Formato del Diccionario de Datos.

Diseño de los mensajes y puertos de los agentes

El modelo parte de un mecanismo de interacción inspirado en el Ciclo Básico de Acción, el cual tiene los siguientes componentes:

- **Cliente**, que representa el agente que solicita el servicio o es usuario del mismo. En el modelo el *Cliente* es representado por el agente que inicia la conversación.
- **Proveedor**, que representa el agente que puede brindar el servicio o es el responsable del mismo. En el modelo el *Proveedor* es representado por el agente que interactúa con el *Cliente* dentro de una cierta conversación.

- **Estados**, que representan las diferentes etapas en las que puede estar una conversación entre dos agentes. En el modelo se representa por medio de las plazas en las Redes de Petri Coloreadas.
- **Actos de Habla**, que representa el mecanismo de comunicación utilizado entre los agentes con el fin de interactuar de una manera coordinada. En el modelo se representa por medio de las transiciones en las Redes de Petri Coloreadas.

El mecanismo de interacción tiene los mensajes o actos del habla definidos, de acuerdo al estado en que se encuentre en una conversación determinada. Es una actividad para el ingeniero de software, el utilizar los diferentes actos de habla con el objetivo de transformar un diagrama de interacción, con su(s) respectiva(s) especificaciones de las diferentes conversaciones en la fase de análisis, a una especificación detallada de las conversaciones en la fase de diseño, donde la misma se encuentra en términos de actos de habla.

En la tabla 4.1 tenemos los diferentes actos de habla utilizados en [Flores 96], cuando un agente está en el rol de Cliente, de acuerdo a las diferentes etapas de la conversación, como se presenta en el capítulo 2. Cuando el ingeniero de software está diseñando los mensajes y puertos de los diferentes agentes, es importante que considere en el mismo la incorporación de los diferentes actos de habla de acuerdo al rol que juegan en la conversación los agentes.

Tabla 4.1: Actos de Habla del Agente Cliente.

	Interacción			
Actos Comunicativos Básicos	Preparación	Negociación	Ejecución	Evaluación
Cliente				
Request	Inicia			
Declare satisfaction				
No agr				Inicia
No rep				Inicia
Void				Usa
Cancel				
No agr		Usa		
Preparation	Inicia			
Void				Inicia
Cancel make new request		Inicia		
Decline to accept				
No agr		Inicia		
Void			Inicia	
Close				
Revoked				Inicia
Declined		Usa		Usa
Ask recons				
Revoked		Inicia		
Declined		Inicia		
Counter		Usa		
Decline counteroffer		Usa		
Agree to counteroffer			Inicia	

Notación:
Inicia: Dispara la conversación.
Usa: Utilizado en la conversación.

En la tabla 4.2 tenemos los diferentes actos de habla utilizados en [Flores 96], cuando un agente está en el rol de Proveedor, de acuerdo a las diferentes etapas de la conversación, como se presenta en el capítulo 2.

Tabla 4.2: Actos de Habla del Agente Proveedor.

Actos Comunicativos Básicos	Interacción			
	Preparación	Negociación	Ejecución	Evaluación
Proveedor				
Agree			Inicia	
Decline		Inicia		
Counteroffer		Inicia		
Revoke and counteroffer		Inicia		
Revoke				
No agr		Usa		
Void			Usa	
Report completion				
No agr				Inicia
Void				Inicia
Ask recons				
Cancel		Inicia		
Close				
Canceled				Usa
Satisfied				Usa

Notación:
Inicia: Dispara la conversación.
Usa: Utilizado en la conversación.

Es factible que el ingeniero de software pueda utilizar los actos comunicativos de FIPA [FIPA 2001] cuando diseñe los diferentes puertos de comunicación y mensajes. En las tablas 4.3 y 4.4 se presenta una relación entre los actos comunicativos de [Flores 96] y [FIPA 2001], de acuerdo al rol del agente en la conversación, donde se especifica que acto de habla de FIPA es factible utilizar en el diseño de los puertos. Es importante notar que los actos de Flores son más específicos en ciertos casos, por lo que un solo acto de habla de FIPA puede equivaler a varios de Flores. En caso que el ingeniero de software decida utilizar los actos de habla de FIPA, éste debe realizar modificaciones a las Redes de Petri Coloreadas que modelan el mecanismo de Flores, con el objetivo que las mismas "entiendan" el lenguaje de comunicación.

Tabla 4.3: Actos de Habla de FIPA Vs. Flores para el Cliente.

Cliente	Comunicación para la Acción (Flores)				Fipa
Actos Comunicativos	**Request**	**Promise**	**Inform**	**Declare**	**Actos Comunicativos**
Request	X				Request
Declare satisfaction				X	Inform
No agr					
No rep					
Void					
Cancel				X	Cancel
No agr					
Preparation					
Void					
Cancel make new request	X				Reject - proposal
Decline to accept				X	Refuse
No agr					
Void					
Close				X	Inform
Revoked					
Declined					
Ask recons	X				Request - when
Revoked					
Declined					
Counter	X				Propose
Decline counteroffer				X	Refuse
Agree to counteroffer				X	Agree

Tabla 4.4: Actos de Habla de FIPA Vs. Flores para el Proveedor.

Proveedor	Comunicación para la Acción (Flores)				Fipa
Actos Comunicativos	**Request**	**Promise**	**Inform**	**Declare**	**Actos Comunicativos**
Agree		X			Agree
Decline				X	Refuse
Counteroffer	X				Propose
Revoke and counteroffer	X				Reject - proposal
Revoke				X	Refuse
No agr					
Void					
Report completion				X	Inform
No agr					
Void					
Ask recons	X				Request - when
Cancel					
Close				X	Inform
Canceled					
Satisfied					

El diseño de los puertos de comunicación de los agentes nos ayuda a reducir la complejidad en la modelación del SIC, en particular en el número de ligas entre los agentes en el diagrama que los representa. El concepto de Puerto de Comunicación es ampliamente

utilizado en el diseño de sistemas distribuidos y es una de las áreas de oportunidad en el diseño de SIC´s. Existen dos tipos de puertos de comunicación en el modelo:

- Los Puertos de Entrada (*Pin*).
- Los Puertos de Salida (*Pout*).

Los Puertos de Entrada son aquellos que representan el conjunto de mensajes que un agente puede recibir de otro en una conversación dada y para una cierta interacción. Los Puertos de Salida son aquellos que representan el conjunto de mensajes que un agente puede enviar a otro en una conversación dada y para una cierta interacción. Cada uno de los puertos de comunicación tiene un tipo de datos asociado que representa el conjunto de mensajes que puede recibir o enviar a través del mismo. Es importante notar que en una conversación *C*, entre los agentes A_1 y A_2, el puerto de entrada del agente A_1 es equivalente al Puerto de Salida del agente A_2, por lo que hablamos de puertos simétricos por conversación. Esto lo denotamos por $A_1.Pin_{C,A2}=A_2.Pout_{C,A1.}$

Cada puerto de entrada o salida tiene asociado un conjunto de mensajes que intercambia con otros agentes dentro de una conversación dada. Este conjunto de mensajes debe ser construido de acuerdo a los actos de habla definidos en el lenguaje de coordinación de acciones, en nuestro caso, los actos de habla de Flores o de FIPA.

Para el diseño de los mensajes exploramos las siguientes opciones: 1) utilizar las primitivas básicas de Flores o FIPA presentadas en las tablas 4.3 y 4.4, tales como *request*, *inform*, entre otros, para representar los mensajes intercambiados entre los agentes y como parámetro el contenido del mensaje, con la información necesaria para su procesamiento. 2) utilizar un formato libre y lo más apegado posible al lenguaje de negocios o natural, para representar los diferentes mensajes, de acuerdo a los diagramas de interacción construidos y a la especificación de las conversaciones. Uno de los parámetros debe definir el tipo de mensaje que es, de acuerdo a la clasificación presentada en Flores y FIPA de las tablas 4.3 y 4.4, tales como: *request*, *inform*, entre otros. Un ejemplo de la primera opción es un mensaje de la forma: *request(parámetros)*. En la segunda opción, el mensaje seria de la forma: *pedir(parámetros, request)*.

Si la opción utilizada es la primera, la restricción mayor está en la rigidez del lenguaje de comunicación, pero la ventaja está en la representación y modelación del mecanismo de interacción, ya que éste es universal en su estructura para diferentes dominios de aplicación. Si la opción utilizada es la segunda, la ventaja está en la expresividad y flexibilidad del lenguaje de comunicación, pero su restricción mayor está en la necesidad de modelar para cada aplicación el mecanismo de interacción o de generar interpretes del lenguaje de comunicación a los actos de habla de Flores o FIPA, según sea el caso.

A partir del diagrama de agentes derivamos el diagrama de puertos, que se presenta en la figura 4.7. Este diagrama representa los diversos puertos de comunicación, tanto de entrada como de salida, para los diferentes agentes de acuerdo a las conversaciones en las que participe.

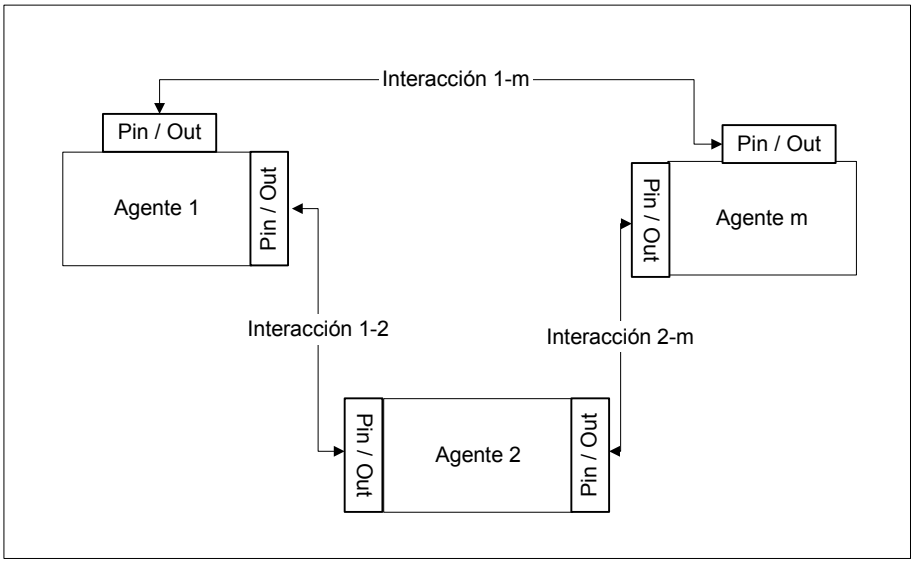


Figura 4.7: Diagrama de Puertos.

Del diagrama de puertos y con base en los diferentes mensajes del mecanismo de interacción se genera la siguiente información, que representa el diseño de los puertos:

- *Interacción(Agente $_1$, Agente $_2$, ..., Agente $_n$)*, que define el nombre de la interacción y los agentes que participan en la misma, donde $n \leq m$, con m el máximo número de agentes en el sistema.
- $\{s_1, s_2, ..., s_m\}$, es el conjunto de conversaciones que forman parte de la interacción, donde la conversación $s_k = (a_p, a_q, d_k)$, como se explico anteriormente, es representada de la siguiente manera:
  - $a_p, a_q \in A$, conjunto de agentes del sistema, $p \neq q$, $p=1..m$, $q=1..m$, m= número de agentes del SIC.
  - $k=1..r$, donde r es el total de conversaciones en la interacción.
  - d_k es una cadena de caracteres que representa el identificador asignado a la conversación k.
- Puertos: Los actos comunicativos son utilizados de acuerdo al ciclo básico de acción, *Loop*. Los agentes en una interacción pueden jugar diferentes roles, como *cliente* o *proveedor*, de acuerdo a las conversaciones en las que participen. En los puertos de entrada y de salida se representa está situación. La notación que utilizamos para representar los puertos en una conversación C, entre el *Agente $_1$* y el *Agente $_2$* es:
  - *Agente $_1$.P(in o out) $_C$,Agente $_2$ ($m_1, m_2, ..., m_x$)*, donde m_i es el mensaje i en el puerto dentro de la conversación C entre el *Agente $_1$* y el *Agente $_2$*, con $i=1..x$, donde x es el número total de mensajes en el puerto y *Agente $_1$*, *Agente $_2 \in A$*, con A= Conjunto de Agentes del SIC, como se presentó anteriormente.
  - El tipo de datos del puerto está dado por el número y tipo de mensajes que puede manejar. Los mensajes deben ser en nuestro caso los actos comunicativos de Flores o los de FIPA, pero no una mezcla de los dos. Si el ingeniero de software quiere utilizar otro tipo de mensajes, el modelo lo permite, siempre y cuando éste sea consistente a lo largo de la aplicación del mismo.
- Para el diagrama de la figura 4.7 tenemos los siguientes puertos:

- Los puertos del Agente $_1$ son:
  - Agente $_1$.Pin$_{\text{interacción1.2,}}$Agente $_2$ (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type)).
  - Agente $_1$.Pout$_{\text{interacción1-2,}}$Agente $_2$ (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer).
  - Agente $_1$.Pin$_{\text{interacción1.m,}}$Agente $_m$ (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type)).
  - Agente $_1$.Pout$_{\text{interacción1-m,}}$Agente $_m$ (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer).
- Los puertos del Agente $_2$ son:
  - Agente $_2$.Pin $_{\text{interacción1-2}}$, Agente$_1$ = Agente $_1$.Pout $_{\text{interacción1-2}}$, Agente2.
  - Agente $_2$.Pout $_{\text{interacción1-2}}$, Agente $_1$ = Agente $_1$.Pin $_{\text{interacción1-2,}}$ Agente 2.
  - Agente $_2$.Pin $_{\text{interacción2-m}}$, Agente $_m$ (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type)).
    - Agente $_2$.Pout $_{\text{interacción2-m}}$, Agente $_m$ (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer).
- Los puertos del Agente $_m$ son:
  - Agente $_m$.Pin $_{\text{interacción1-m,}}$ Agente $_1$ = Agente $_1$.Pout $_{\text{interacción1-m}}$ Agente $_m$.
  - Agente $_m$.Pout $_{\text{interacción1-m}}$, Agente $_1$ = Agente $_1$.Pin $_{\text{interacción1-m}}$, Agente $_m$.
  - Agente $_m$.Pin $_{\text{interacción2-m}}$, Agente $_2$ = Agente $_2$.Pout $_{\text{interacción2-m}}$, Agente $_m$.
  - Agente $_m$.Pout $_{\text{interacción2-m}}$, Agente $_2$ = Agente $_2$.Pin $_{\text{interacción2-m}}$, Agente $_m$.

Con el objetivo de entender cuando un agente debe utilizar cada acto de habla de acuerdo al estado de la conversación, a la etapa del *Loop* en la que se encuentre y del rol que desempeñe en ese momento, se construye la Especificación Detallada de la Conversación.

Para la construcción de la Especificación Detallada de la Conversación, retomamos las diferentes especificaciones de las conversaciones generadas en el paso anterior de este modelo y con base en los actos de habla presentados en las tablas 4.1 y 4.2, representamos las acciones de cada etapa del Loop. En la figura 4.8 se presenta el formato de la Especificación Detallada de la Conversación. Nótese que en una etapa puede darse un intercambio de actos de habla de acuerdo al mecanismo de interacción que se está utilizando.

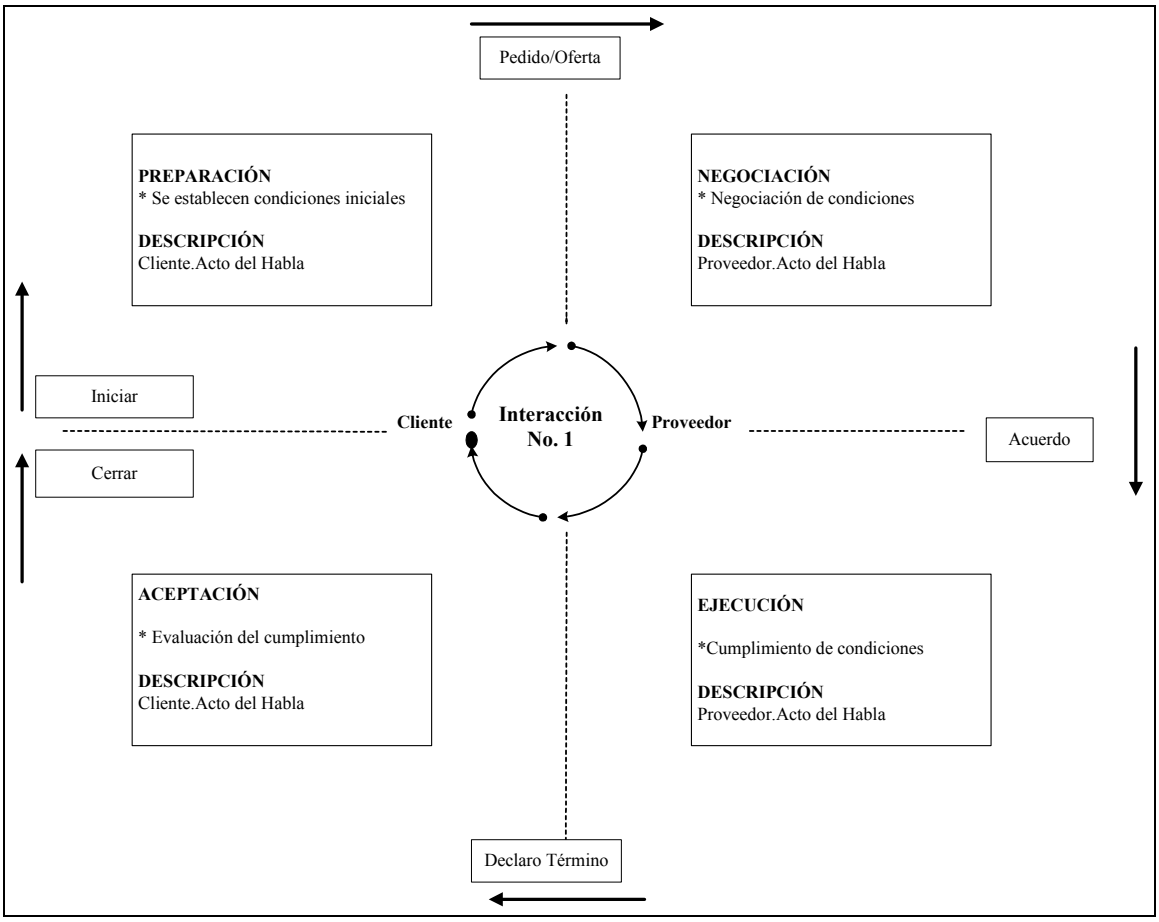


Figura 4.8: Especificación Detallada de la Conversación.

Representación del modelo de interacción con Redes de Petri Coloreadas

En este paso del modelo, representamos el mecanismo de interacción que da soporte a los diferentes tipos de mensajes que utilizan los agentes durante sus conversaciones. En este caso, representamos por medio de Redes de Petri Coloreadas el conjunto de estados, relaciones y actos del habla que representan el mecanismo de interacción.

En este modelo los estados son representados por medio de las plazas y los actos de habla por medio de transiciones. Las relaciones representan transiciones entre plazas, producto de un cambio de estado en la conversación debido a un acto de habla generado por alguno de los agentes.

El mecanismo general de interacción se presenta en la figura 4.9, donde se tiene la página jerárquica (*hierarchy page*, de su notación en Inglés en las RPC), llamada *BasicActionLoopForInteraction*, en donde los agentes Cliente y Proveedor son representados y las diferentes páginas y sus relaciones son mostradas. Cada uno de los nodos de la página jerárquica es una RPC y existe un nodo especial llamado nodo de declaraciones (*Declaration Node*).

En la página jerárquica se representa el mecanismo de interacción *loop*. En ella tenemos la RPC que representa al Cliente, llamada *MainClient*, con los diferentes estados en los que puede estar en la conversación, los cuales están representados por las RPC: *Negotiation, Performance, Acceptance, Acceptance No Agreement, Closure After Decline, Closure After Revoke, Closure After Cancel No Agreement, Countered, Closure After Early Satisfaction.*

También tenemos la RPC que representa al Proveedor, llamada *MainProvider*, con los diferentes estados en los que puede estar en la conversación, los cuales están representados por las RPC: *Performance, Acceptance, Acceptance No Agreement, Closure After Decline, Closure After Revoke, Closure After Cancel, Countered, Closure After Early Satisfaction*.

El mecanismo de plazas de fusión es utilizado para interconectar la estructura de la Red de Petri Coloreada en diferentes páginas. La plaza de fusión es una plaza que es igualada por una o más plazas tal que las plazas fusionadas actúan como una sola plaza con una sola marca [Jensen 97a]. Se definen varios conjuntos de fusión, que agrupan plazas que tienen el mismo nombre, como por ejemplo *Negotiation*, *Preparation*, entre otros, y éstas representan las plazas comunes en la interacción. A continuación se presentarán las Redes de Petri Coloreadas de los estados principales del mecanismo de interacción.

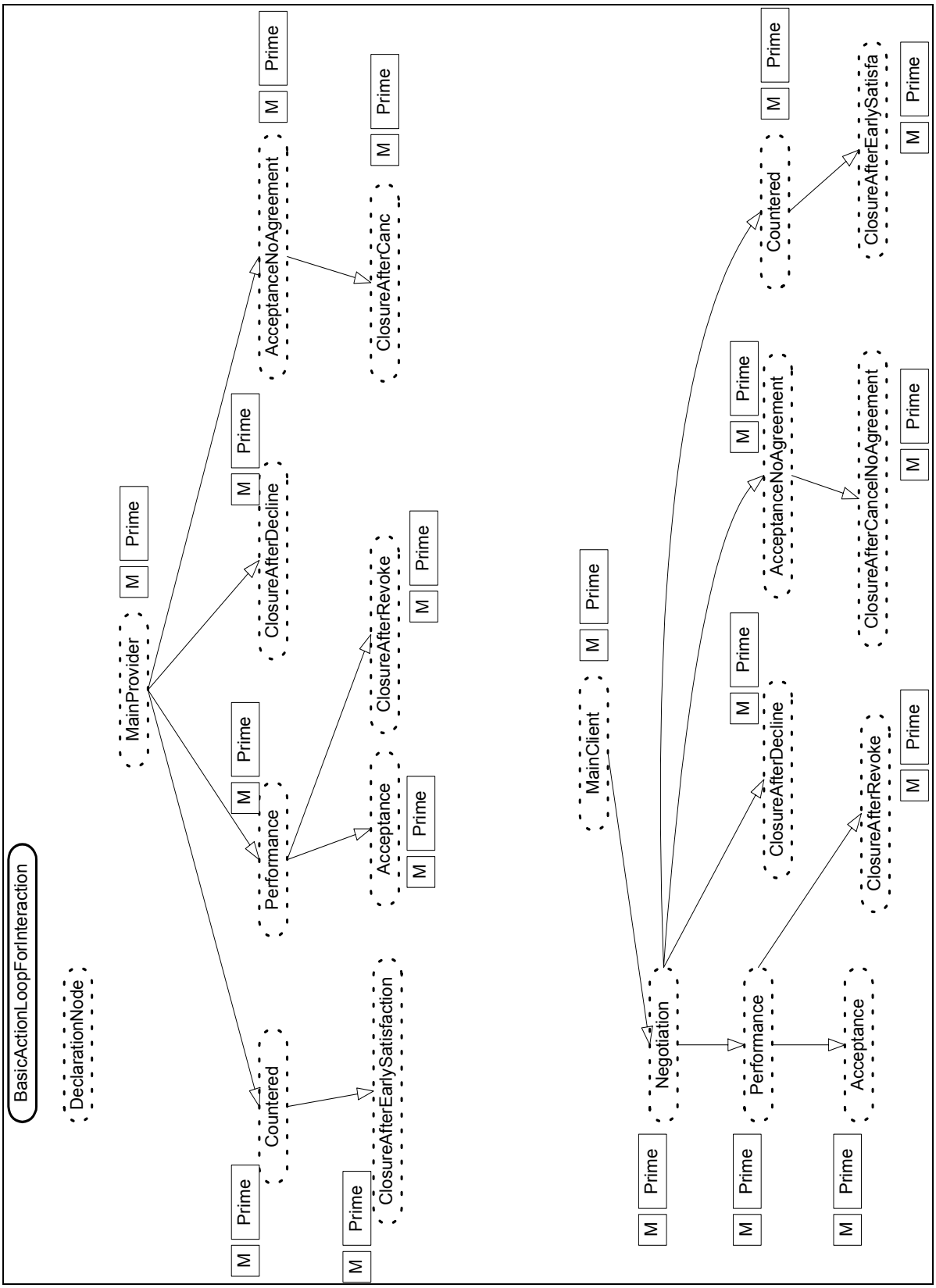


Figura 4.9: La Página Jerárquica en RPC para el Mecanismo General de Interacción.

La etapa de *Preparación (Preparation)* es la primera en una conversación general. Ésta se presenta en la figura 4.10 y es la RPC *MainClient*. La conversación inicia con un mensaje del cliente: *request*, y el proveedor puede responder con diferentes tipos de mensajes, tales como *agree*, *decline*, *report completion with no agreement*, y *counteroffer*, dentro de la etapa llamada *Negociación (Negotiation)*.

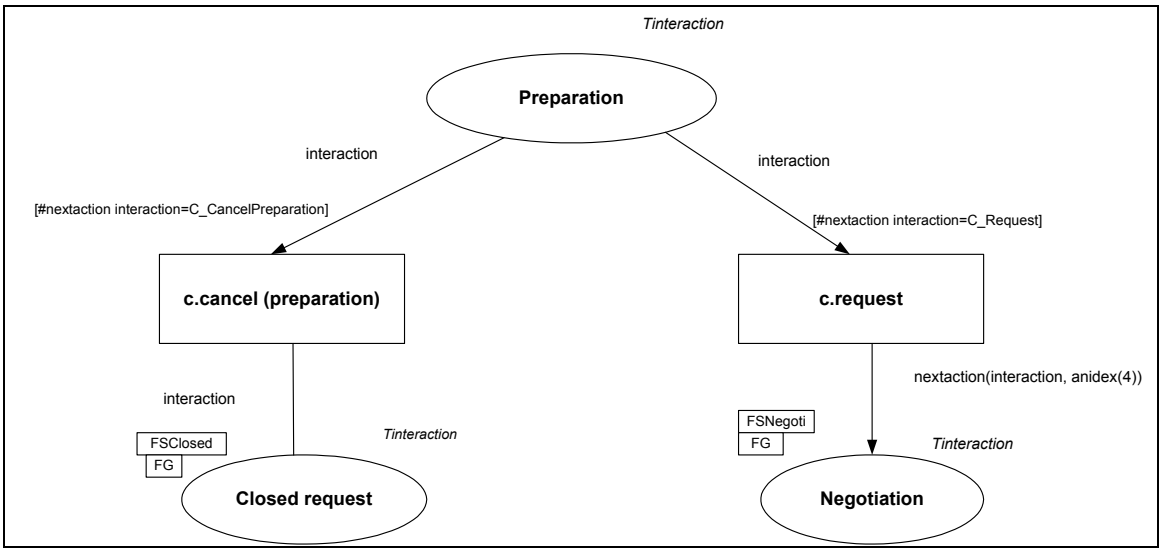


Figura 4.10: RPC de la Etapa de Preparación.

En una interacción normal, sin conflictos, el flujo comunicativo de coordinación es: cliente *request*, proveedor *agree*, proveedor *report completion*, cliente *declare satisfaction*. Pero si los agentes enfrentan una situación de interacción con conflicto, requieren de un mecanismo adicional llamado negociación [Rosenschein 94], [Rosenschein 98a], [Rosenschein 98b], [Sycara 91], donde un posible flujo comunicativo de coordinación es: cliente *request* y el proveedor tiene diferentes opciones:

- *Agree,* para aceptar la petición; *Decline*, para rechazar la petición; *Report completion with no agreement,* para terminar la petición; *Counteroffer*, para ofrecer otra opción.

Una situación de interacción de este tipo es parte de la etapa de *Negociación* en la RPC de *MainProvider*, donde el cliente y el proveedor tienen un intercambio de mensajes antes de avanzar a la siguiente etapa. Esta etapa se presenta en la figura 4.11.

La figura 4.12 presenta una parte de la etapa de *Negociación*, la situación de *Contraoferta* (*Countered)*, donde el cliente o el proveedor pueden aceptar, rechazar o contraofertar la petición. La etapa de *Contraoferta* es central en el mecanismo de interacción para resolver conflictos y llegar a acuerdos.

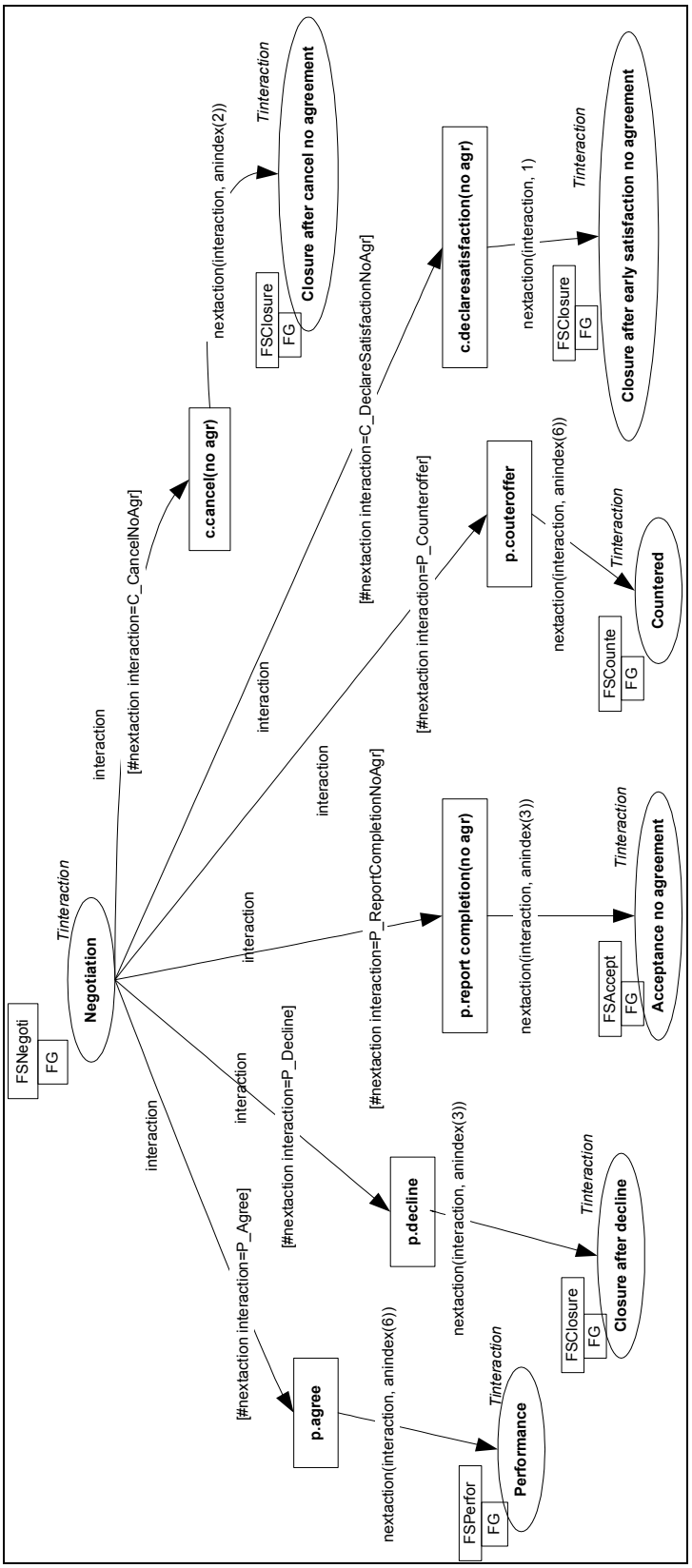


Figura 4.11: RPC de la Etapa de Negociación.

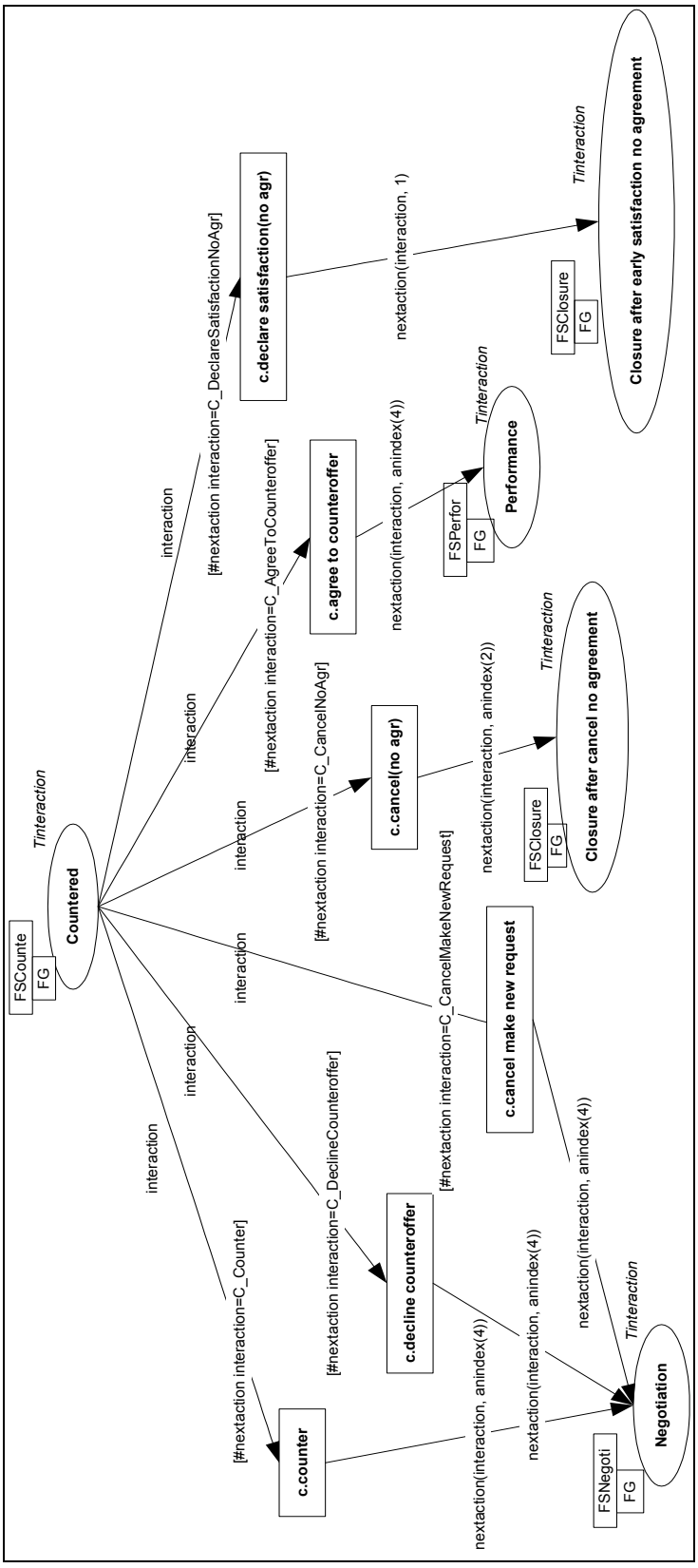


Figura 4.12: RPC de la Etapa de Contraoferta.

La figura 4.13 presenta la etapa de *Ejecución* (*Execution)*, donde el proveedor trabaja para satisfacer el requerimiento del cliente. El proveedor puede utilizar los mensajes siguientes: *report completion*, *revoke*, *revoke and counteroffer*. En el caso del mensaje de *revoke*, la conversación probablemente regresa a la etapa de *Negociación*, y con el mensaje de *report completion* la conversación se mueve a la etapa de *Evaluación (*E*valuation)*.

Cuando la conversación está en la etapa de *Evaluación*, presentada en la figura 4.14, el cliente debe evaluar el trabajo del proveedor, de acuerdo al resultado que le ofrece y con base en las condiciones de satisfacción acordadas en la etapa de *Negociación*. Los mensajes validos en esta etapa son *declare satisfaction*, *decline to accept*, *cancel* o *cancel and make new request*. Si los mensajes son *declare satisfaction* o *cancel*, la conversación termina, pero si el mensaje es *decline to accept*, la conversación regresa a la etapa de *Ejecución* o con el mensaje *cancel and make new request*, ésta regresa a la etapa de *Negociación*.

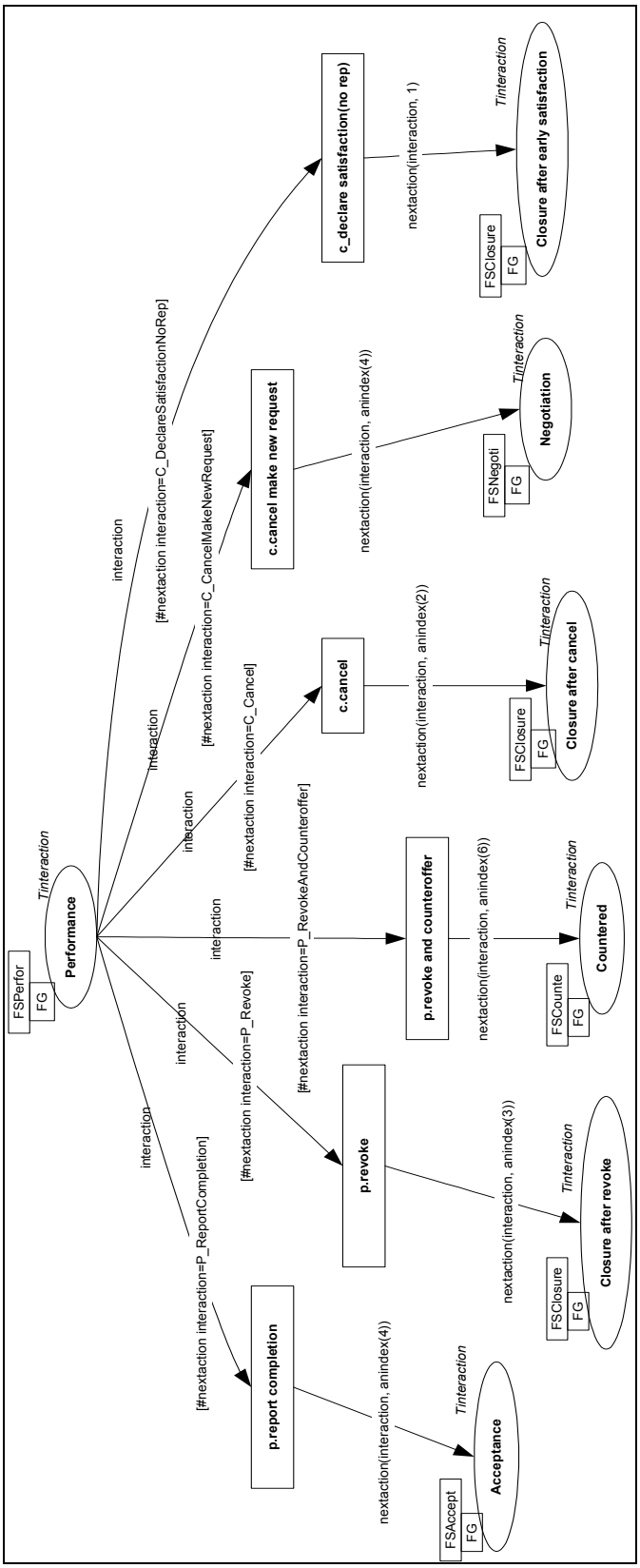


Figura 4.13: RPC de la Etapa de Ejecución.

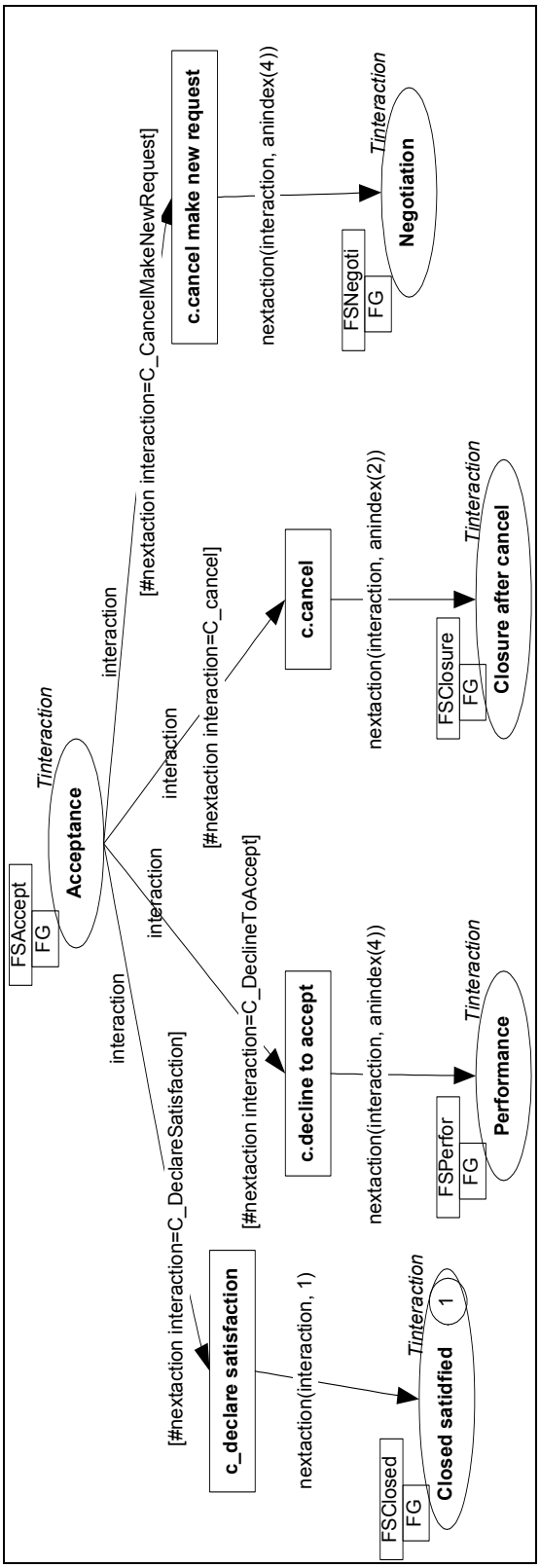


Figura 4.14: RPC de la Etapa de Evaluación.

Para modelar la interacción en la Red de Petri Coloreada, lo ilustramos con un ejemplo. Aquí tenemos dos instancias diferentes de un token de color llamado Interacción, en una conversación determinada.

(i) interacción $_1$ = ((Agente $_1$), Agente $_2$, (Agente $_3$, Agente $_4$, …, Agente $_n$), Producto $_1$, Agente $_2$.Declare Satisfaction))

(ii) interaction $_2$ = ((Agente $_7$), Agente $_2$, (Agente $_4$, Agente $_5$, … , Agente $_m$), Producto $_2$, Agente $_2$.Request))

En la instancia (i), el Agente $_2$ está en la etapa de *Aceptación* en la interacción $_1$, y en la instancia (ii), el mismo Agente $_2$ se encuentra en la etapa de *Preparación* en la interacción $_2$. Aquí podemos ver como en la misma RPC, múltiples y simultáneas conversaciones son modeladas expresivamente y controladas de acuerdo a los valores de los tokens en algún momento del tiempo. Cada instancia tiene diferentes Agentes participando en las conversaciones y los valores de cada uno de ellos forman el estado particular de una conversación en un momento dado.

La adaptación del mecanismo general de interacción para dar soporte a una aplicación en particular se realiza a través de diversas funciones desarrolladas en el área de *declaración* de las Redes de Petri Coloreadas, así como en las definiciones de los colores para los tokens. Es trabajo del ingeniero de software plasmar la lógica de la aplicación en las Redes de Petri Coloreadas que modelan el mecanismo de interacción para poder simular la aplicación en el siguiente paso.

4.4. Modelo Dinámico

Al terminar la serie de pasos anteriores tenemos un modelo estático de la interacción en el Sistema de Información Cooperante, y es una vista explícita del mismo construida por el ingeniero de software. En este paso entramos en la vista implícita del modelo y vamos a observar la parte dinámica del mismo, a través de la simulación del mecanismo de interacción en la herramienta Design/CPN.

La simulación de los modelos de interacción nos ayuda a observar posibles dinámicas de la interacción en el SIC, antes de implementar el mismo en un lenguaje de programación determinado y sobre una arquitectura dada. El realizar las simulaciones manualmente puede generar errores cuando se hace, además de ser un proceso tedioso y lento, y en algunas ocasiones impractico para sistemas de gran tamaño. El poder utilizar una herramienta automática para esta actividad agrega valor al ciclo de ingeniería de software y asegura la calidad en el desarrollo de los SIC´s.

Los modelos del paso anterior, en los cuales se desarrolla un conjunto de Redes de Petri Coloreadas deben estar incorporados a la herramienta. Para ello el ingeniero de software debe cuidar que el modelo esté correctamente representado en Design/CPN. La herramienta ofrece, entre otras características relevantes, una verificación de la sintaxis del modelo la cual es útil para realizar la simulación.

Al construir la Red de Petri Coloreada, el ingeniero de software debe cuidar los siguientes aspectos:

- Generar la Página Jerárquica, donde represente la estructura global del sistema y las relaciones entre las diferentes RPC. Esta página se genera mediante la técnica de refinamiento paso a paso, de tal manera que puede incorporar o retirar RPC de acuerdo al modelo del SIC.
- Generar el Nodo de Declaraciones o Página de Declaraciones Globales, donde se representan los tipos de datos del SIC, las variables y funciones, entre otros elementos. Esta área es muy importante al momento de especializar el mecanismo de interacción a un dominio de aplicación dado.
- Generar las diferentes páginas que contienen las Redes de Petri que forman el SIC. Es importante notar que el conjunto de RPC forman la estructura de la red y ésta es la parte estática de la misma.

En el momento de construir la Red de Petri Coloreada recomendamos ver el Anexo B de este trabajo. Los componentes básicos a cuidar de cada elemento del diagrama, en sus diferentes regiones, son:

- Plazas:
  - Nombre.
  - Conjunto de Colores (Tipo).
  - Marca Inicial.
- Transiciones:
  - Nombre.
  - Guardas.
  - Código.
  - Tiempo.
- Arcos:
  - Inscripciones.

De acuerdo al mecanismo seleccionado, se requiere definir los Conjuntos de Fusión, las Plazas de Fusión o la Sustitución de Transiciones, con el objetivo de modelar grandes sistemas de información cooperantes, de una manera jerárquica. Para revisar a detalle la sintaxis de la herramienta recomendamos consultar los manuales en línea que se encuentran en [Jensen 2001] o en [Christensen 2001].

Encontramos que con el uso de las Redes de Petri Coloreadas, el modelar interacciones múltiples en SIC´s resulta más expresivo para el ingeniero de software, además la complejidad asociada a la modelación y control de las mismas se disminuyo. Elementos relevantes de las RPC que utilizamos para lograr éstos resultados son:

- El uso de las Redes de Petri Coloreadas, para reducir la complejidad asociada en la modelación de interacciones múltiples y mejorar la expresividad de los modelos resultantes.

- Los Tokens de Colores, donde se representan y modelan tipos de datos abstractos, permitiendo que las interacciones múltiples entre agentes puedan ser representadas de una manera expresiva y puedan ser controlada de una manera relativamente sencilla.
- El mecanismo de las Plazas de Fusión , que ayuda a modelar grandes sistemas de información cooperantes jerárquicamente, con una estructura modular, compartiendo estados entre los diferentes módulos del sistema.
- La representación en el diagrama de la Red de Petri Coloreada de los estados de las interacciones como plazas y los actos comunicativos como transiciones.
- Finalmente, en la herramienta Design/CPN, la ejecución de la interacción donde se puede observar y controlar antes de la implementación del sistema, mediante una simulación de la misma.

La integración de diversos métodos, técnicas y herramientas en un modelo que facilite la representación de múltiples interacciones en sistemas de información cooperantes es una aportación al dominio de ingeniería de software orientada a agentes.

Capítulo 5

Aplicaciones y Análisis de Resultados

El validar el modelo de interacciones múltiples en sistemas de información cooperantes, es una parte central en este trabajo, así como el especificar los mismos utilizando RPC. En nuestro enfoque iniciamos modelando el mecanismo de interacción utilizado el Ciclo Básico de Acción y sobre el mismo incorporamos los modelos que representen diversos dominios de aplicación. A continuación se presentaran los escenarios de simulación y la especificación de los mismos.

5.1. Implementación del Mecanismo de Interacción Utilizando Redes de Petri Coloreadas

Una parte central en la modelación de la interacción en sistemas de información cooperantes es el mecanismo de interacción utilizado. A diferencia de [El Fallah 99b] y [Cost 99b] que utilizan diversos mecanismos de interacción en la implementación de los sistemas multiagentes cooperantes, nosotros utilizamos el mecanismo propuesto por [Flores 96]. Para ellos partimos del diagrama de estados y actos de habla que representa el flujo de las conversaciones, los diferentes estados en que se puede encontrar un cliente y un proveedor, así como las transiciones entre ellos cuando un acto de habla se intercambia.

Para modelar el ambiente con Redes de Petri Coloreadas, y reducir la complejidad asociada con múltiples interacciones simultáneas, se determinó que las *plazas* representarían los diferentes estados de las conversaciones y que las *transiciones* representarían los actos de habla que se intercambian entre el cliente y el proveedor, que en el contexto de este trabajo representan agentes. Las funciones y declaraciones presentadas en este capítulo están escritas en CPN-ML.

El concepto de *Token de Color* es fundamental en el modelo, ya que es la base para representar y controlar interacciones múltiples de manera simultánea y expresiva. La integración de los diferentes estados de las conversaciones entre los agentes se realiza a través del concepto de *Plazas de Fusión* o *Fusion Place* en Inglés, las cuales nos ayudan a modelar sistemas complejos de una manera jerárquica, siguiendo una estructura modular y sobre todo, compartiendo estados entre diferentes módulos del sistema.

Encontramos que la mejora en la expresividad en la modelación de las interacciones múltiples utilizando RPC se tiene en el poder representar y observar el estado de las interacciones entre más de dos agentes, a través de los valores de un token de un color determinado, en un momento dado del tiempo. El representar diferentes mensajes para varios agentes en diversos estados se realiza utilizando las *funciones* que proveen las RPC y los tokens de color.

Si se quisiera representar el comportamiento de los agentes en la interacción de acuerdo al estado en que se encuentren, esto sería factible de realizar mediante el uso de *funciones* combinadas con el estado de la interacción para un agente representado por los valores de un token de un color determinado.

Una parte fundamental es el uso de la herramienta Design/CPN, la cual integra los conceptos mencionados anteriormente y permite simular las interacciones en el sistema, observar y controlar el estado y la dinámica de las mismas antes de la implementación del Sistema de Información Cooperante.

La simulación del modelo se basa en la generación aleatoria de actos de habla por parte de los agentes participantes en la interacción, donde se analiza lo siguiente:

- Flujo Básico: Marca inicial que permita seguir la conversación básica entre los agentes.
- Todos los Flujos: Marcas y Tokens que lleven tanto al cliente como al proveedor a sus diferentes estados, para probar el modelo de la RPC.
- Flujos con Conflicto: Subconjunto del anterior, donde se simula un conflicto que lleve la conversación a la etapa de negociación.

Posteriormente se revisarán las situaciones deseadas para las aplicaciones modeladas. Parte de la función que genera la selección aleatoria del acto de habla se presenta en la Tabla 5.1; en el Anexo E se presenta completa.

La función *anindex* recibe como parámetro el número máximo de opciones que puede tener un agente de acuerdo a los actos de habla que puede utilizar en un estado dado, y regresa un número entero que se utiliza para seleccionar el siguiente acto de habla. Para ello genera un número aleatorio con la función *randomindex* y si el número es mayor que el máximo de opciones, lo normaliza con la función *fixtheindex* de tal manera que siempre se cuenta con una opción valida al momento de seleccionar un acto de habla. La función *fixtheindex* se presenta en el Anexo E.

Tabla 5.1: Función para Calcular un Acto de Habla en la Simulación.

```
fun randomindex(theindex:Toptions) =
   ran'TOptions ();
```

```
fun anindex(thelimit : TOptions) =
```

```
Let
 val tempind = randomindex(thelimit)
 val arigthindex = tempind <= thelimit
in
 case arigthindex of
  true  => avalue(tempind) |
  False => fixtheindex(tempind,thelimit)
end;
```

Los actos de habla considerados, se presentan en la Tabla 5.2 en el lenguaje CPN-ML, y se modelan por medio de un token de color enumerado llamado *Tports*, el cual contiene tanto el puerto de entrada (*Pin*) como el de salida (*Pout*) de los agentes. Estos actos de habla pueden ser los de Flores o los de FIPA.

Tabla 5.2: Actos de Habla del Mecanismo de Interacción.

```
color TPorts = with  P_Agree |
               P_Decline |
               P_Counteroffer |
               P_Comment |
               P_ReportCompletion | P_ReportCompletionNoAgr |
               P_Revoke | P_RevokeNoAgr |
               P_RevokeAndCounteroffer |
               P_Close | P_CloseSatisfied | P_CloseCanceled |
               P_AskRecons | P_AskReconsCancel | (* TPortIn *)
               C_Request |
               C_Comment |
               C_DeclareSatisfaction | C_DeclareSatisfactionNoAgr |
               C_DeclareSatisfactionNoRep |
               C_Cancel | C_CancelPreparation | C_CancelNoAgr |
               C_CancelMakeNewRequest |
               C_DeclineToAccept | C_DeclineToAcceptNoAgr |
               C_Close | C_CloseRevoked | C_CloseDeclined |
               C_AskRecons | C_AskReconsRevoke | C_AskReconsDecline |
               C_Counter |
               C_DeclineCounteroffer |
               C_AgreeToCounteroffer;   (* TPortOut *)
```

El token de color que representa la interacción es un elemento central, ya que debe permitir de una manera expresiva la modelación y el control de más de dos interacciones simultáneas. Uno de los objetivos de diseño del modelo presentado en esta tesis es la facilidad de la adaptación del mismo a cambios en las especificaciones del sistema así como su aplicabilidad a diversos dominios de aplicación. Para ello la especificación está dividida en diversas *funciones* y *tokens* de colores que permiten, de una manera sencilla,

realizar esta labor de mantenimiento. En la Tabla 5.3 presentamos la definición del token de color que representa la interacción, donde se modelan los agentes participantes (*Agent1*, *Agent2*, ... , *AgentN*), la siguiente acción (acto de habla, *nextaction*) e información adicional que se requiera en la interacción (*addinfo*).

Tabla 5.3: Definición del Token de Color que Representa la Interacción.

```
color Tinteraction = record
        Agent1:Tagent1 *
        Agent2: Tagent2*
        ... *
        agentN: TagentN *
        addinfo: Tadinfo *
        nextaction: Tports;
```

En la Tabla 5.4 podemos observar un ejemplo particular de un token de color que representa una interacción aplicado al ambiente de Negocio a Negocio que presentaremos en la siguiente sesión.

Tabla 5.4: Ejemplo del Color *Tinteraction* para el Ambiente de Negocio a Negocio.

```
color Tinteraction = record
        buyer:Tlistbuyer *
        seller: TMarketplace *
        supplier:Tlistsupplier *
        theproduct: Tproduct *
        nextaction: TPorts;
```

El mecanismo de interacción se modela por un conjunto de Redes de Petri Coloreadas, donde en cada una de ellas se representan los estados de las conversaciones entre un cliente y un proveedor. Los *tokens* que circulan sobre las RPC representan las interacciones entre los diferentes agentes que se encuentran jugando los roles de cliente o proveedor. La Figura 5.1 presenta una RPC donde podemos observar:

- Plazas, que representan estados de la conversación.
- Transiciones, que representan actos de habla.
- Inscripciones en los arcos, que representan las diferentes interacciones del sistema y la forma como se calcula el siguiente acto de habla.
- Guardas en las transiciones, que determinan cual transición debe activarse.

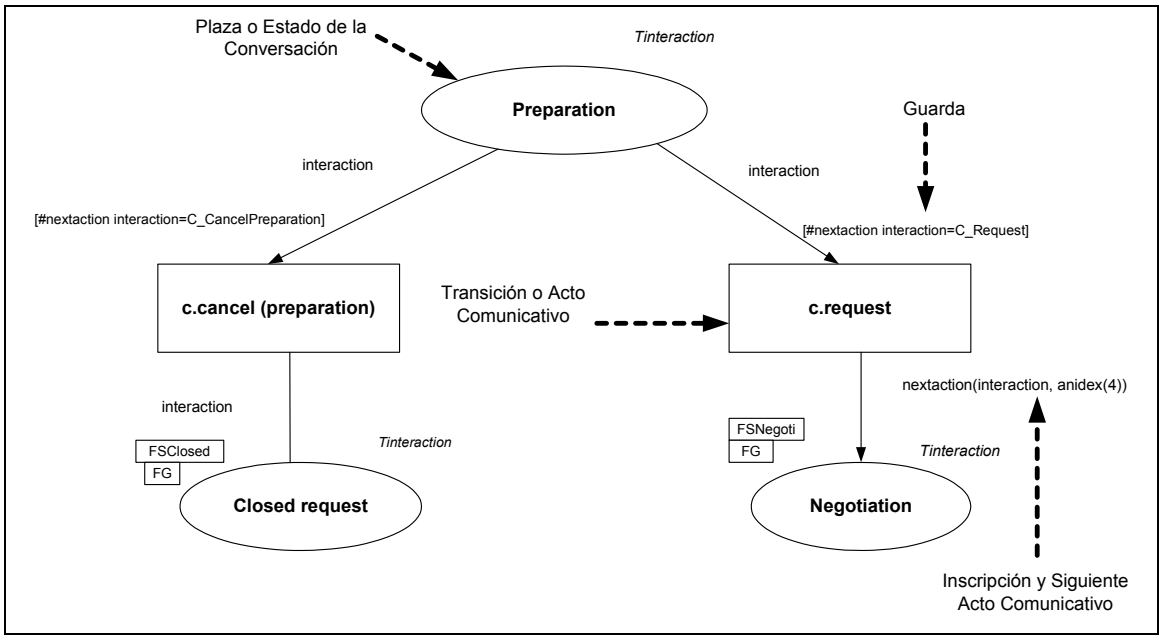


Figura 5.1: Representación de un Estado con una Red de Petri Coloreada.

La función que calcula el siguiente acto comunicativo o de habla que un agente debe emitir en una interacción dada, de acuerdo al estado en que se encuentra, se presenta en la Tabla 5.5. En ella podemos observar que la función *nextaction* recibe una interacción dada y un índice (número entero) que representa el siguiente acto de habla calculado por la función *anindex* descrita en la Tabla 5.1.

Tabla 5.5: Función *nextaction*.

```
fun nextaction(intein: Tinteraction, tindex: TOptions) =
   {buyer = #buyer intein
   seller =#seller intein
   supplier = #supplier intein
   theproduct =#theproduct intein
   nextaction =selectaction(#nextaction intein, tindex)};
```

La función *nextaction* asigna los valores de la interacción seleccionando el siguiente acto comunicativo de acuerdo al índice que se le entregó como parámetro en *tindex*. Esto se realiza con la función *selectaction* que se presenta en la Tabla 5.6, la cual recibe el acto de habla anterior y el índice para seleccionar el nuevo. Es importante notar que las reglas que rigen la interacción se representan en estas funciones y están acordes al mecanismo de interacción del Ciclo Básico de Acción.

Tabla 5.6: Función *selectaction*.

```
fun selectaction(anaction:TPorts, tindex:TOptions) =
   case anaction of
```

```
    C_Request => pcanegotiation(anaction, tindex) |
    C_DeclineToAccept => pcaperformance(anaction,tindex) |
    C_Counter => pcanegotiation(anaction, tindex) |
    C_DeclineCounteroffer => pcanegotiation(anaction, tindex) |
    C_DeclineToAcceptNoAgr => pcanegotiation(anaction, tindex) |
    C_CancelMakeNewRequest => pcanegotiation(anaction, tindex) |
    C_CancelNoAgr => pcaclosureaftercancelnoagreement(anaction, tindex) |
    C_AgreeToCounteroffer => pcaperformance(anaction, tindex) |
    C_DeclareSatisfactionNoAgr => pcaclosureafterearlysatisfactionnoagreement(anaction,
tindex) |
    C_Cancel => pcaclosureaftercancel(anaction, tindex) |
    C_DeclareSatisfactionNoRep => pcaclosureafterearlysatisfaction(anaction, tindex) |
    C_DeclareSatisfaction => pcaclosedsatisfied(anaction, tindex) |
    C_AskReconsRevoke => pcanegotiation(anaction, tindex) |
    C_AskReconsDecline => pcanegotiation(anaction, tindex) |
    C_Close => pcacomment() |
    C_CloseRevoked => pcacomment() |
    C_CloseDeclined => pcacomment() |
    C_AskRecons => pcanegotiation(anaction, tindex) |
    P_Agree   => ccaperformance(anaction, tindex) |
    P_Decline => ccaclosureafterdecline(anaction, tindex) |
    P_ReportCompletionNoAgr => ccaacceptancenoagreement(anaction, tindex) |
    P_Counteroffer => ccacountered(anaction, tindex) |
    P_ReportCompletion => ccaaceptance(anaction, tindex) |
    P_Revoke => ccaclosureafterrevoke(anaction, tindex) |
    P_RevokeAndCounteroffer => ccacountered(anaction, tindex) |
    P_RevokeNoAgr => ccaclosureafterrevokenoagreement(anaction, tindex) |
    P_CloseCanceled => ccacomment() |
    P_CloseSatisfied => ccacomment() |
    P_Close => ccacomment() |
    P_AskRecons => ccacountered(anaction, tindex) |
    P_AskReconsCancel => ccacountered(anaction, tindex) ;
```

La función *selectaction* llama a la correspondiente función de acuerdo al acto de habla que recibe como parámetro. Por ejemplo, en la Tabla 5.6, en la última línea, si el acto de habla es *P_AskReconsCancel*, donde el proveedor pide reconsiderar la cancelación por parte del cliente, la función *selectaction* llama a la función *ccacountered* que implica que el cliente realizará una *contraoferta* de acuerdo a la petición del proveedor. La función *ccacountered* se presenta en la Tabla 5.7 donde se observan los diversos actos de habla que el cliente puede utilizar en la etapa de *negociación* en el estado de una *contraoferta*. En el Anexo E se presenta la página de declaraciones del modelo en Design/CPN, donde se encuentra la definición de todas las funciones.

Tabla 5.7: Función *ccacountered*.

```
fun ccacountered(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_Counter |
    2 => C_DeclineCounteroffer |
    3 => C_CancelMakeNewRequest |
    4 => C_CancelNoAgr |
    5 => C_AgreeToCounteroffer |
    6 => C_DeclareSatisfactionNoAgr ;
```

A continuación presentaremos los productos resultantes de la aplicación del modelo y la simulación del mismo en dos aplicaciones, los Ambientes de Negocio a Negocio y los Centros de Contacto.

5.2. Modelo de Interacciones Múltiples en un Ambiente de Negocio a Negocio

En un ambiente de Negocio a Negocio (Business to Business, B2B de sus siglas en Inglés) se dan diversas interacciones simultáneas entre los participantes como: compradores, proveedores y la plaza de mercado (o marketplace en Inglés), que necesitan ser representadas y controladas. En la plaza de mercado, los diferentes participantes intercambian información acerca de precios y ofertas de productos, con el objetivo que se generen transacciones comerciales entre ellos, buscando un beneficio común. Estas transacciones generalmente ocurren en un ambiente totalmente distribuido y abierto, utilizando Internet como el medio de comunicación. La figura 5.2 presenta de manera conceptual una plaza de mercado con varios clientes y proveedores interactuando.

El representar las relaciones entre los diversos participantes es un problema complejo para los ingenieros de software, y el controlar el ambiente a medida que el número de clientes y proveedores aumenta lo complica más. A continuación presentaremos paso a paso la modelación de esta aplicación en el modelo propuesto, con el objetivo de observar como los modelos generados brindan un mejor entendimiento del problema al ingeniero de software y permiten observar la dinámica de las interacciones en este ambiente.

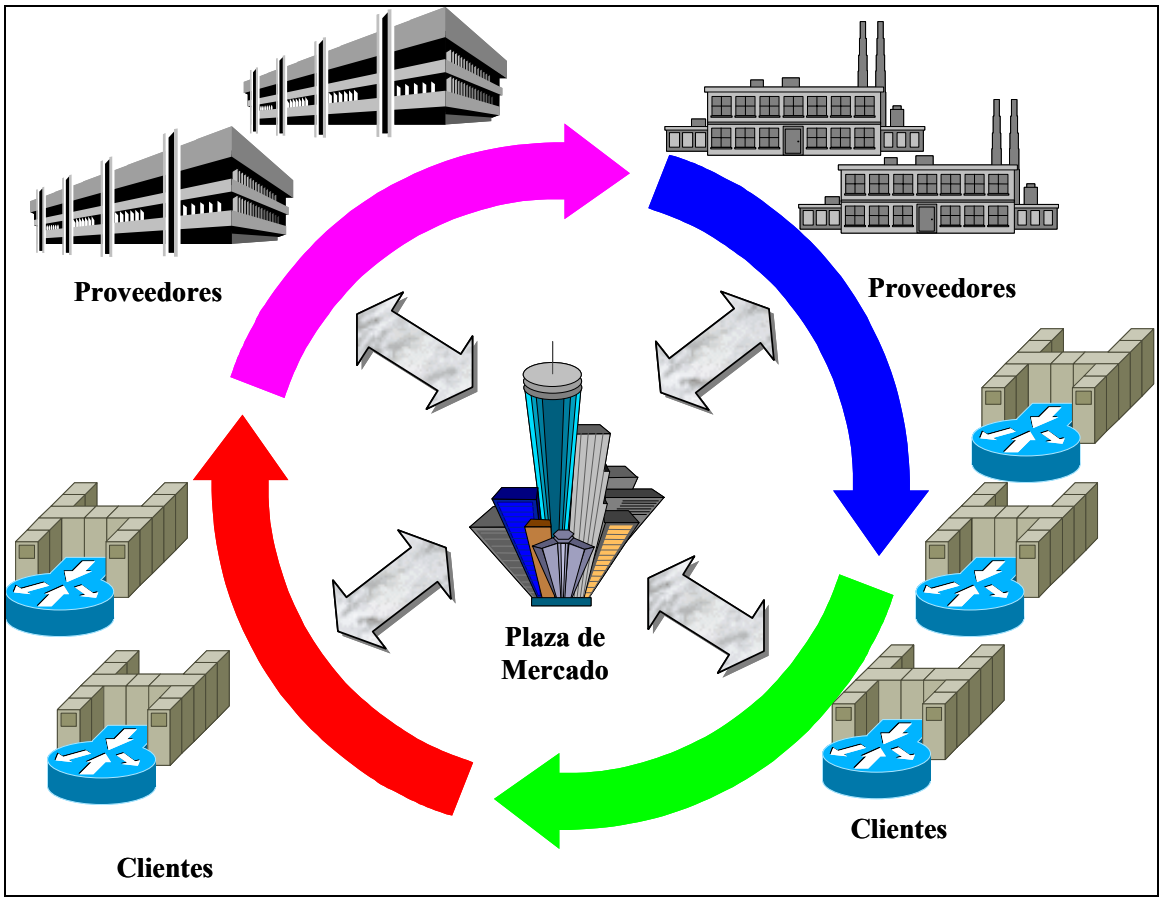


Figura 5.2: Esquema Conceptual de una Plaza de Mercado.

Modelo Individual

El primer paso es identificar los agentes y sus intenciones. Aquí tenemos tres agentes:

- Comprador.
- Plaza de Mercado.
- Proveedor.

Construimos el conjunto de agentes A, con los agentes y sus intenciones:

A = {(Comprador, Comprar el producto que necesita con la mejor opción en el mercado), (Plaza de Mercado, Facilitar relaciones efectivas entre compradores y proveedores), (Proveedor, Realizar la mejor oferta en calidad, precio y volumen de productos)}.

Modelo Estructural

La Figura 5.3 presenta el diagrama de agentes derivado del segundo paso, donde se identifican las diferentes conversaciones entre los agentes, como son: *Transacción B2B*, *Ordenar*, *Pedir Propuestas*, *Establecer Acuerdos* y *Esperar por la entrega del producto*.

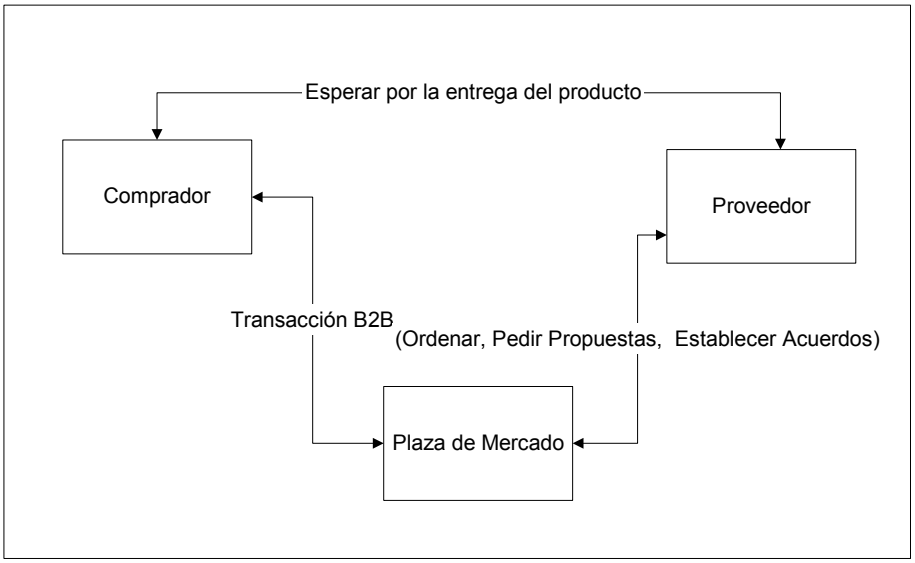


Figura 5.3: Diagrama de Agentes Negocio a Negocio.

Para construir el diagrama de interacciones, la interacción principal identificada del sistema es: *Transacción B2B*, la cual está compuesta por cuatro conversaciones:

- Comprador *Transacción B2B* Plaza de Mercado.
- Plaza de Mercado *Pedir Propuestas* Proveedor.
- Plaza de Mercado *Ordenar* Proveedor.
- Comprador *Espera por la entrega del producto* Proveedor.

La interacción *Establecer Acuerdos* entre la Plaza de Mercado y el Proveedor es una interacción de contexto y solamente da información de referencia o que es prerrequisito para la interacción principal. La Figura 5.4 presenta el diagrama de interacción.

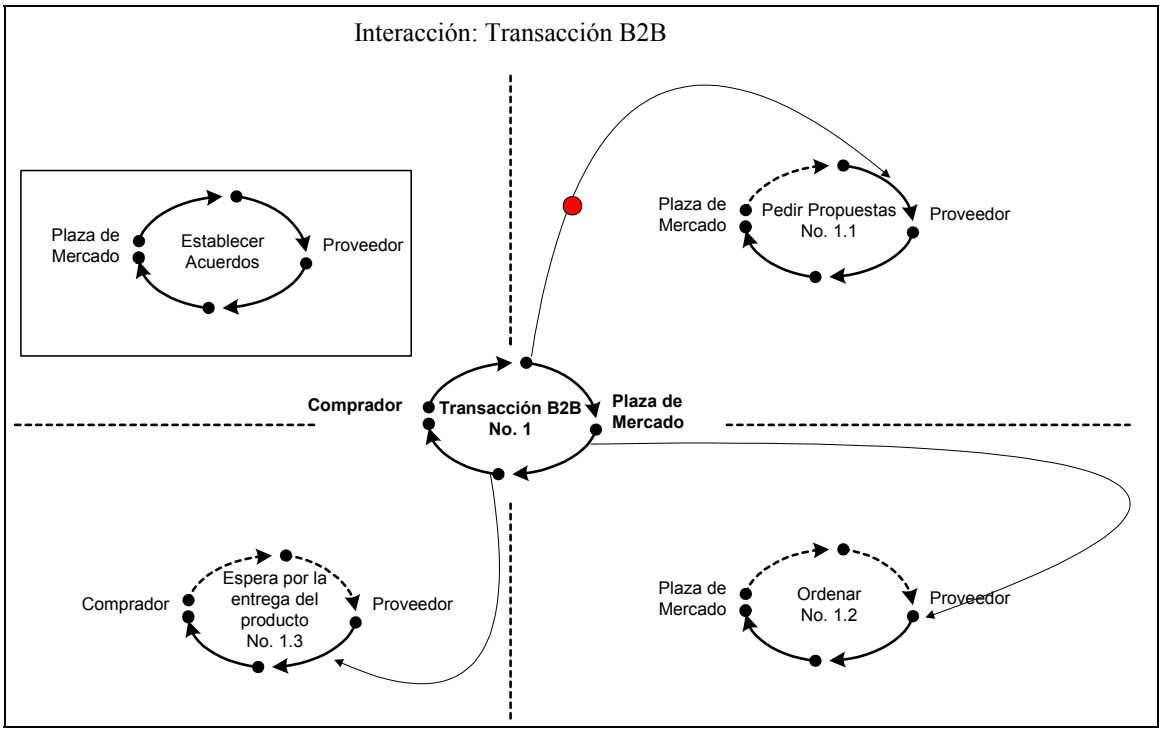


Figura 5.4: Diagrama de Interacción para la Transacción B2B.

La especificación de la conversación *Pedir Propuestas* entre la Plaza de Mercado y el Proveedor se presenta en las Figuras 5.5 y 5.6. Aquí describimos la información acerca de la interacción y sus conversaciones en las dos partes que componen la forma:

- La Descripción de la Conversación, presentada en la Figura 5.5.
- La Especificación de la Interacción, presentada en la Figura 5.6.

Descripción de la Conversación:

Nombre de la Interacción: Transacción B2B	ID de la Interacción: *1*	Fecha: *Octubre 2001*	Versión *1.0*
Nombre de la Conversación: Pedir Propuestas		ID de la Conversación: *1.1*	
Objetivo Principal de la Conversación: Recibir las mejores ofertas de los proveedores de acuerdo a las necesidades de los clientes.			
Cliente: *Plaza de Mercado*	Proveedor: *Proveedor*		

Figura 5.5: Descripción de la Conversación *Pedir Propuestas*.

En la especificación de la conversación, se describe el intercambio de actos comunicativos entre los agentes *Plaza de Mercado* (con el rol cliente) y *Proveedor* (con el rol proveedor) en cada paso del Ciclo Básico de Acción, *Loop*.

Todas las conversaciones en el diagrama de interacción deben tener una descripción y especificación de la conversación como las presentadas en las Figuras 5.5 y 5.6. Los mensajes son representados por los actos comunicativos, en este caso en partícular los de Flores. Como referencia estos actos se presentaron en las Tablas 4.1 y 4.2.

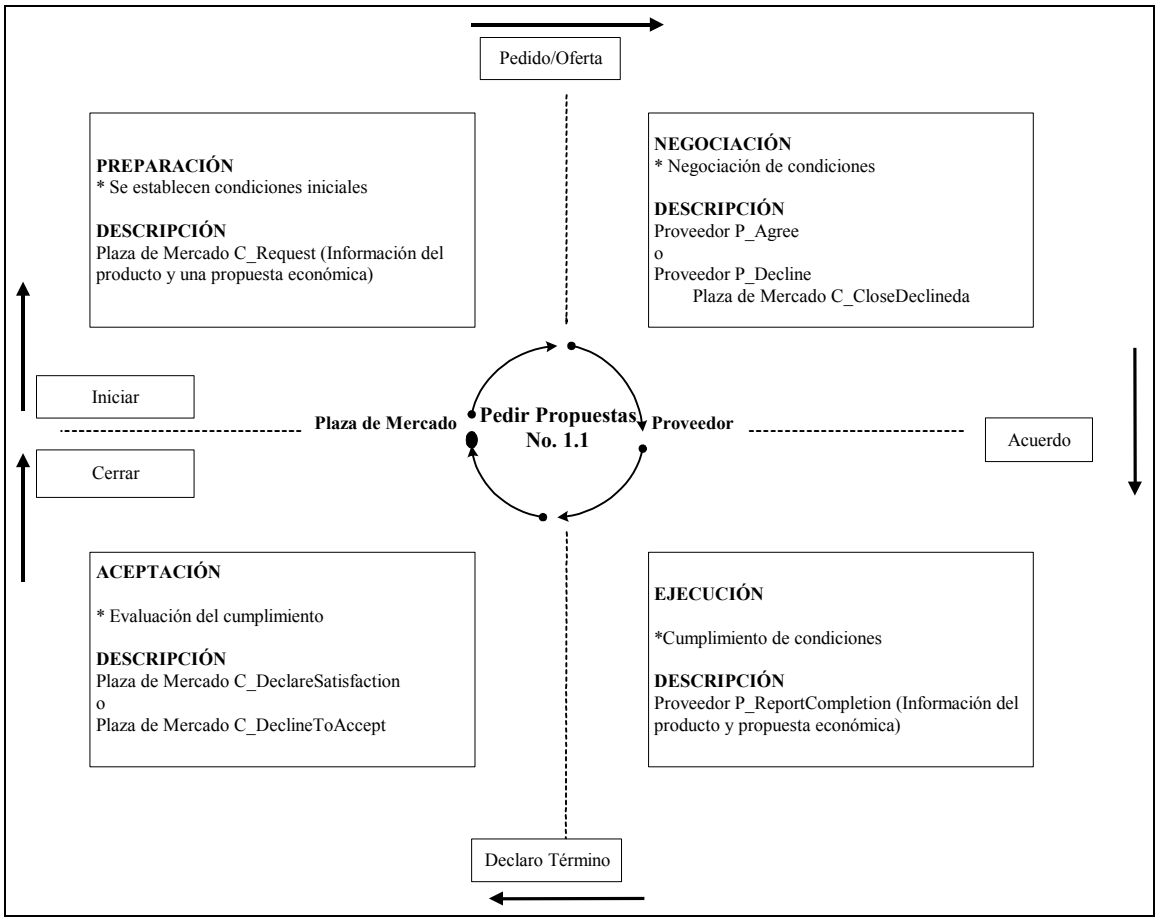


Figura 5.6: Especificación de la Conversación *Pedir Propuestas*.

En el modelo de la conversación *Pedir Propuestas*, la *Plaza de Mercado* pide una propuesta económica de cierto producto, dentro de un acuerdo previo al que llegó con el *Proveedor* en la interacción *Establece Acuerdos*. Los *Proveedores* pueden o no enviar sus propuestas, pero la *Plaza de Mercado* toma la decisión de aceptar o rechazar la misma, con base en el acuerdo previo. Cuando un agente está participando en una interacción, el flujo de la conversación es controlado por el Ciclo Básico de Acción, *Loop*.

Por ejemplo, en la Figura 5.6, en el paso de *Aceptación*, si la *Plaza de Mercado* regresa el mensaje *C_DeclareSatisfaction*, la conversación termina, pero si el mensaje es *C_DeclineToAccept*, la conversación se mueve al paso de *Ejecución*. La Tabla 5.8 presenta la función *ccaacceptance* que modela en Design/CPN esta situación.

Tabla 5.8: Función *ccaacceptance*.

```
fun ccaaceptance(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_DeclareSatisfaction |
    2 => C_DeclineToAccept |
    3 => C_Cancel |
    4 => C_CancelMakeNewRequest ;
```

Al finalizar este paso, podemos representar el Sistema de Información Cooperante (SIC) por medio de la siguiente expresión: SIC={A, Cg, Gk, I}, donde:

- Conjunto de Agentes, *A = {(Comprador, Comprar el producto que necesita con la mejor opción en el mercado), (Plaza de Mercado, Facilitar relaciones efectivas entre compradores y proveedores), (Proveedor, Realizar la mejor oferta en calidad, precio y volumen de productos)}*.
- Objetivo Común, Cg= Satisfacer las necesidades de los Compradores con las mejores ofertas y los mejores acuerdos con los Proveedores.
- Conocimiento global, Gk={Necesidad del Comprador, Productos, Acuerdos}.
- Conjunto de Interacciones, *I={Transacción B2B}*.

Ahora, diseñaremos a detalle los diferentes puertos del sistema, de acuerdo a la interacción principal:

Interacción:
Transacción B2B (Comprador, Plaza de Mercado, Proveedor)

Conversaciones:
- {Comprador *Transacción B2B* Plaza de Mercado,
- Plaza de Mercado *Pedir Propuestas* Proveedor,
- Plaza de Mercado *Ordenar* Proveedor,
- Comprador *Espera por la entrega del producto* Proveedor}.

Puertos:
Conformados por los actos comunicativos que se utilizan de acuerdo al Ciclo Básico de Acción, *Loop*. Los agentes *Comprador*, *Plaza de Mercado* y *Proveedor*, juegan dos roles diferentes en una conversación, el de Cliente y el de Proveedor.

Un agente en cada conversación en la que participa, tiene dos puertos, el puerto de entrada (Pin) donde recibe los mensajes y el puerto de salida (Pout) a través del cual envía los mensajes. En la Figura 5.7 se presenta el diagrama de puertos.

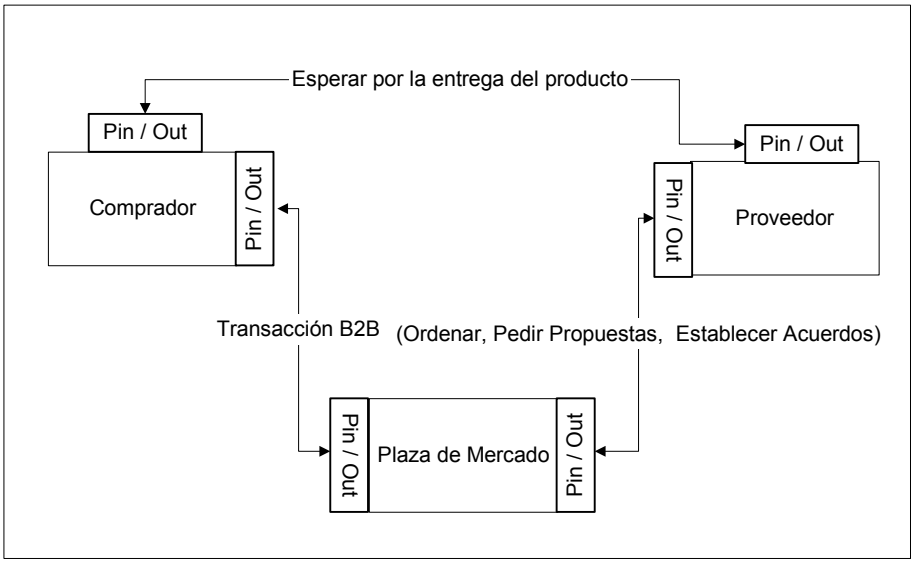


Figura 5.7: Diagrama de Puertos del Ambiente de B2B.

Los puertos del *Comprador* son:

- Comprador.Pin $_{\text{Transacción B2B}}$, Plaza de Mercado (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type), Comment).
- Comprador.Pout $_{\text{Transacción B2B}}$, Plaza de Mercado (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer, Comment).
- Comprador.Pin $_{\text{Esperar por la entrega del producto}}$, Proveedor (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type), Comment).
- Comprador.Pout $_{\text{Esperar por la entrega del producto}}$, Proveedor (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer, Comment).

Los puertos de la *Plaza de Mercado* son:

- Plaza de Mercado.Pin $_{\text{Transacción B2B}}$, Comprador = Comprador.Pout $_{\text{Transacción B2B}}$, Plaza de Mercado.
- Plaza de Mercado.Pout $_{\text{Transacción B2B}}$, Comprador = Comprador.Pin $_{\text{Transacción B2B}}$, Plaza de Mercado.
- Plaza de Mercado.Pin $_{\text{Pedir Propuestas, Ordenar, Establecer Acuerdos}}$, Proveedor (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type), Comment).
- Plaza de Mercado.Pout $_{\text{Pedir Propuestas, Ordenar, Establecer Acuerdos}}$, Proveedor (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer, Comment).

Los puertos del *Proveedor* son:

- Proveedor.Pin $_{\text{Pedir Propuestas, Ordenar, Establecer Acuerdos}}$, Plaza de Mercado = Plaza de Mercado.Pout $_{\text{Pedir Propuestas, Ordenar, Establecer Acuerdos}}$, Proveedor.
- Proveedor.Pout $_{\text{Pedir Propuestas, Ordenar, Establecer Acuerdos}}$, Plaza de Mercado = Plaza de Mercado.Pin $_{\text{Pedir Propuestas, Ordenar, Establecer Acuerdos}}$, Proveedor.
- Proveedor.Pin $_{\text{Esperar por la entrega del producto}}$, Comprador = Comprador.Pout $_{\text{Esperar por la entrega del producto}}$, Proveedor.
- Proveedor.Pout $_{\text{Esperar por la entrega del producto}}$, Comprador = Comprador.Pin $_{\text{Esperar por la entrega del producto}}$, Proveedor.

En la tabla 5.9 se tiene la representación en Design/CPN del token de color que modela en forma genérica los puertos y sus mensajes.

Tabla 5.9: Representación de los Puertos en Design/CPN para el Ambiente de B2B.

```
color TPorts = with  P_Agree |
               P_Decline |
               P_Counteroffer |
               P_Comment |
               P_ReportCompletion | P_ReportCompletionNoAgr |
               P_Revoke | P_RevokeNoAgr |
               P_RevokeAndCounteroffer |
               P_Close | P_CloseSatisfied | P_CloseCanceled |
               P_AskRecons | P_AskReconsCancel | (* TPortIn *)
               C_Request |
               C_Comment |
               C_DeclareSatisfaction | C_DeclareSatisfactionNoAgr |
               C_DeclareSatisfactionNoRep |
               C_Cancel | C_CancelPreparation | C_CancelNoAgr |
               C_CancelMakeNewRequest |
               C_DeclineToAccept | C_DeclineToAcceptNoAgr |
               C_Close | C_CloseRevoked | C_CloseDeclined |
               C_AskRecons | C_AskReconsRevoke | C_AskReconsDecline |
               C_Counter |
               C_DeclineCounteroffer |
               C_AgreeToCounteroffer;   (* TPortOut *)
```

El token de color que representa la interacción es un registro con una lista de Compradores (*buyer*), una Plaza de Mercado (*seller*), una lista de Proveedores (*supplier*), un Producto (*theproduct*) y un acto comunicativo que pertenece a los puertos (*nextaction*). Este token se presenta en la tabla 5.10.

Tabla 5.10: Representación de la Interacción para el Ambiente de B2B.

```
color Tinteraction = record
                buyer:Tlistbuyer *
                seller: TMarketplace *
                supplier:Tlistsupplier *
                theproduct: Tproduct *
                nextaction: TPorts;
```

La flexibilidad del modelo generado, así como el uso de las Redes de Petri Coloreadas, permite definir tipos de datos complejos para cada componente del token que representa la interacción. En este caso, para el Comprador, Proveedor, Plaza de Mercado y Producto, se definió su propio token de color y se presentan en las Tablas 5.11, 5.12, 5.13 y 5.14 respectivamente.

Tabla 5.11: Representación del Comprador para el Ambiente de B2B.

```
color Tbuyer = record
            id:Tid *
            n:Tname *
            c:Tcredit *
            d:Tdiscount;

color Tlistbuyer = list Tbuyer;
```

Tabla 5.12: Representación del Proveedor para el Ambiente de B2B.

```
color Tsupplier = record
              id:Tid *
              n:Tname *
              p:Tlistproduct;

color Tlistsupplier = list Tsupplier;
```

Tabla 5.13: Representación de la Plaza de Mercado para el Ambiente de B2B.

```
color TMarketplace = record
              id:Tid *
              n:Tname;
```

Tabla 5.14: Representación del Producto para el Ambiente de B2B.

```
color Tproduct = record
        n:Tname *
        p:Tprice *
        s:Tstock *
        d:Tdescription;

color Tlistproduct = list Tproduct;
```

Las adecuaciones que realizamos al mecanismo de interacción se basan en modificaciones sobre los tokens, como *Tinteraction*, *Tproduct*, entre otros, así como cambios en las funciones que calculan el siguiente acto comunicativo, con el objetivo de representar el problema.

Podemos observar que no se realizan cambios sobre la estructura de la Red de Petri Coloreada que representa el Ciclo Básico de Acción, *Loop*. Un resultado de nuestro análisis es que podemos ver la flexibilidad y robustez del modelo para adaptarse a diferentes situaciones de interacciones múltiples así como a varios dominios de aplicación.

Modelo Dinámico

Para observar la dinámica de las interacciones múltiples en el Ambiente de Negocio a Negocio se simularon las siguientes situaciones de interacción sobre el modelo, con el objetivo de analizar su funcionamiento y la expresividad en la representación del mismo.

Interacciones en el ambiente de Negocio a Negocio en una Plaza de Mercado:

1. **Compra básica:** Un Comprador, una Plaza de Mercado, *n* Proveedores y una petición sin conflicto.
2. **Interacciones simultáneas:** *m* Compradores, una Plaza de Mercado, *n* Proveedores y *m* peticiones sin conflicto.
3. **Compra básica con conflicto:** Similar a la primera, pero con un conflicto al realizar la compra, como la no existencia de un producto y que genera una contraoferta a la petición.
4. **Interacciones simultáneas con conflicto:** Similar a la segunda, pero con un conflicto al realizar la compra, como la no existencia de un producto y que genera una contraoferta a la petición.

Para explicar los resultados observados en la herramienta Design/CPN, vamos a identificar los estados iniciales y finales de las conversaciones.

Estado Inicial:

- *Preparation*

Estados Finales:

- *Closed Satisfied,* termina la conversación de manera satisfactoria para las partes.
- *Closed Canceled,* se cancelo la petición en algún momento de la conversación y las partes están de acuerdo.
- *Closed Declined,* alguna de las partes declino la petición y están de acuerdo.
- *Closed Revoked,* alguna de las partes revoca la petición y están de acuerdo.

Al momento de simular cada una de las situaciones, es importante que sea cual sea, debe terminar con una marca en alguno de los estados finales. En caso contrario podemos concluir, que la interacción nunca llega a terminación y esto sugeriría un problema potencial, como un abrazo mortal, falta de algún recurso, o alguna situación de este estilo.

Análisis de las simulaciones

La Figura 5.8 presenta la marca inicial de la simulación para múltiples interacciones simultáneas, en este caso tres, en el único estado inicial de las misma, *Preparation*. Analicemos que pasa en el momento que las diferentes interacciones representadas por los tokens se activan y las transacciones se van disparando.

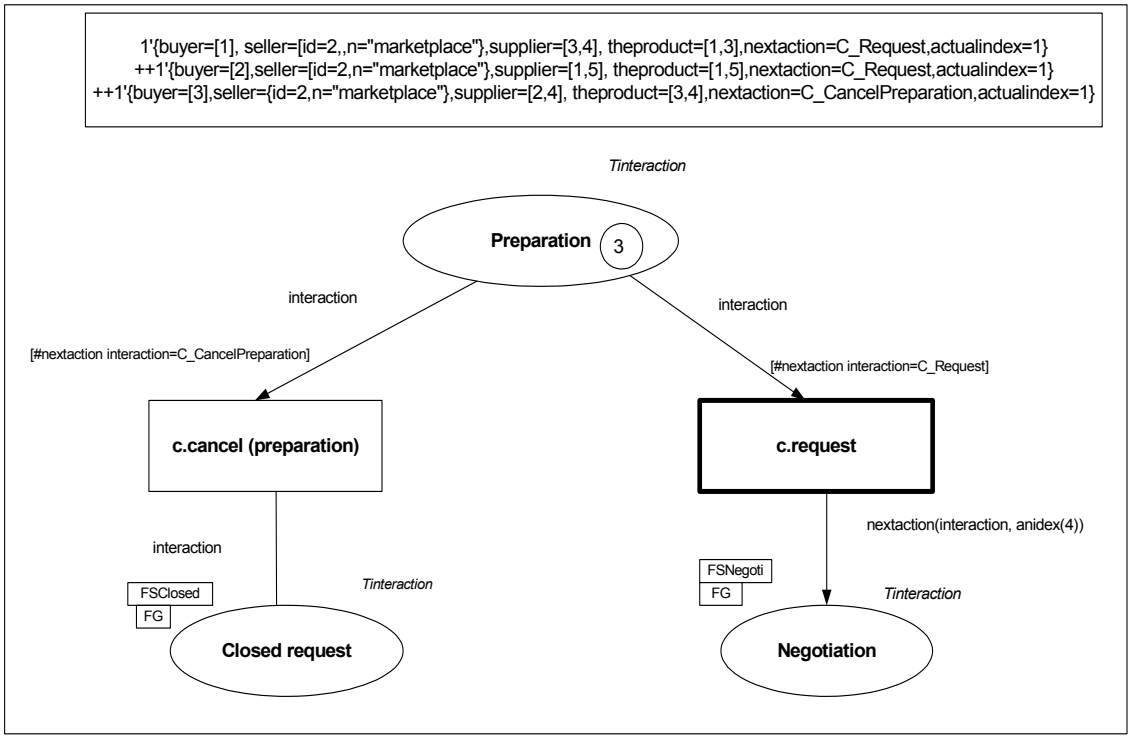


Figura 5.8: Estado Inicial *Transacción B2B*, Múltiples Interacciones Simultáneas.

La Figura 5.9 presenta el estado intermedio de las tres interacciones simultáneas, donde podemos observar el estado de las mismas, de acuerdo a los valores que tienen los tokens. Analizando la situación podemos observar como los compradores, identificados en los tokens con los números 1 y 2, se encuentran en la etapa de negociación, mientras que el comprador identificado con el número 3 ya recibio una contraoferta y se encuentra decidiendo su siguiente acto de habla. En este momento podemos observar que las

transacciones por habilitarse son las correspondiente a los actos de habla de *agree* y *reportcompletion(no agr),* las cuales se seleccionarian de acuerdo al estado de cada interacción. Podemos concluir de nuestro análisis, que sobre una misma estructura de la Red de Petri Coloreada, podemos modelar y controlar múltiples interacciones, como lo estamos observando en la figura 5.9, de una manera sencilla y expresiva.

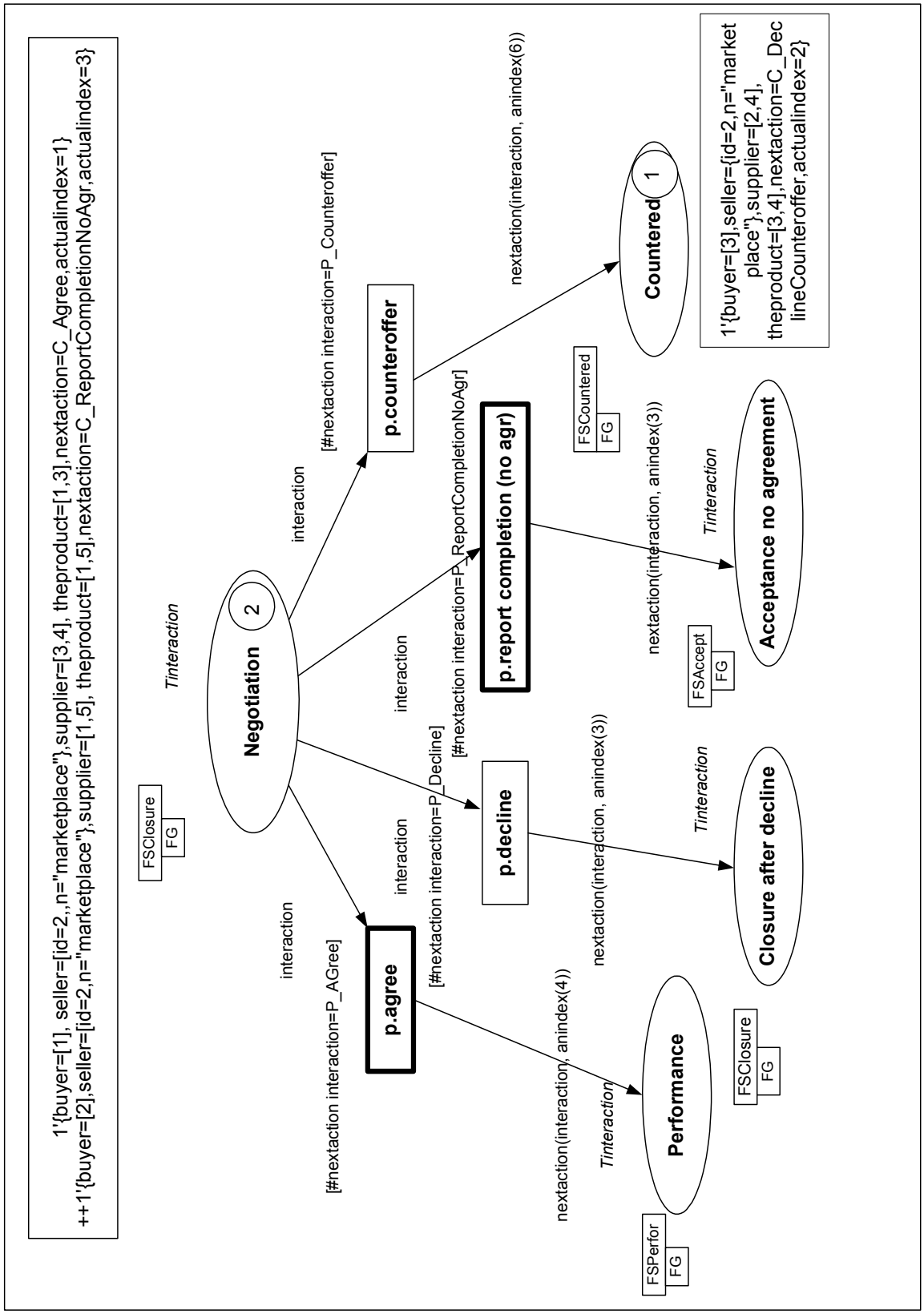


Figura 5.9: Estado Intermedio *Transacción B2B*, Múltiples Interacciones Simultáneas.

La Figura 5.10 presenta el estado final de una interacción de las anteriores. Podemos concluir, con base en el análisis de las interacciones del sistema, que el comprador identificado con el número 1, cancelo su petición y las dos partes, tanto el comprador como el proveedor estuvieron de acuerdo con la decisión.

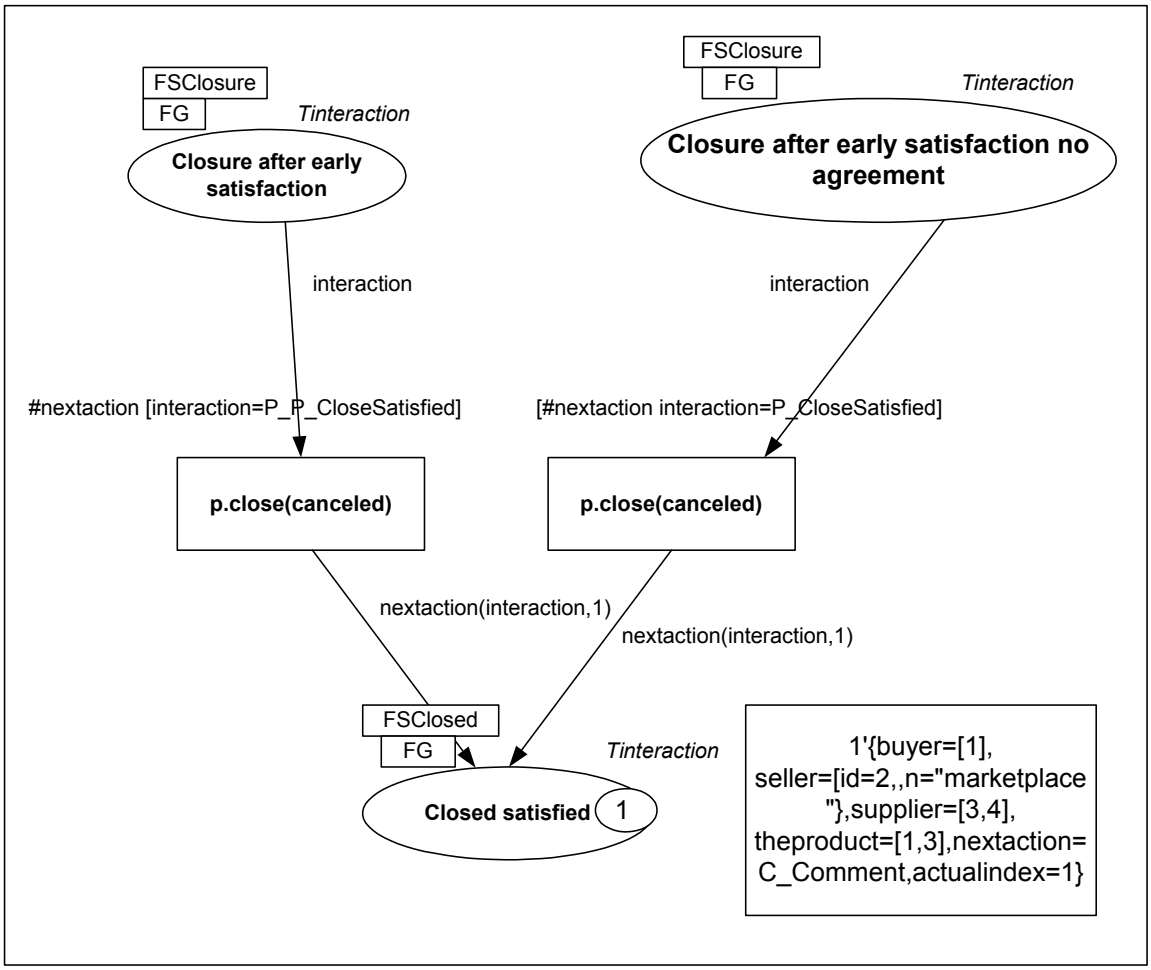


Figura 5.10: Estado Final *Transacción B2B*, Múltiples Interacciones Simultáneas.

5.3. Modelo de Interacciones Múltiples en un Ambiente de Centros de Contacto

Los Centros de Contacto, son espacios únicos de atención a clientes, donde ellos pueden obtener información confiable y realizar procesos o trámites de manera impecable. Estos centros requieren de sistemas de alta coordinación que permitan atender y controlar múltiples interacciones simultáneas entre los clientes del centro y los diferentes centros de proceso, donde se realiza el trabajo necesario para resolver las peticiones. Los diferentes centros de proceso tienen asignados técnicos especializados de acuerdo a la función que realice cada uno de ellos.

En un centro de contacto, cuando una petición es recibida por parte de un usuario, la función principal del centro es resolverla en tiempo y forma mediante la asignación oportuna al centro de proceso que resolverá la petición y el seguimiento al centro de proceso en la asignación efectiva y eficiente de los técnicos requeridos. La figura 5.11 representa un ambiente típico de centro de contacto, donde la necesidad de representar y controlar múltiples interacciones es evidente.

Algunas de las características de estos sistemas, que aprovecharemos para presentar la aplicación del modelo propuesto, son:

- Requieren la interacción simultánea de más de una entidad, llamese centro de proceso, técnico o usuario, entre otras.
- Están basados en un ambiente de trabajo cooperativo y requieren coordinación de acciones e interacción entre las partes para resolver la petición.
- Utilizan sistemas de comunicación y diversos medios de contacto de los usuarios hacía el centro de contacto, como teléfono, internet, fax, entre otros.
- Requieren de trabajo cooperativo de los agentes para llegar a la solución.
- Tienen una tarea común a resolver.
- Requieren mantener una alta interacción entre sus componentes.
- Están inmersos en ambientes distribuidos, heterogéneos y altamente dinámicos.

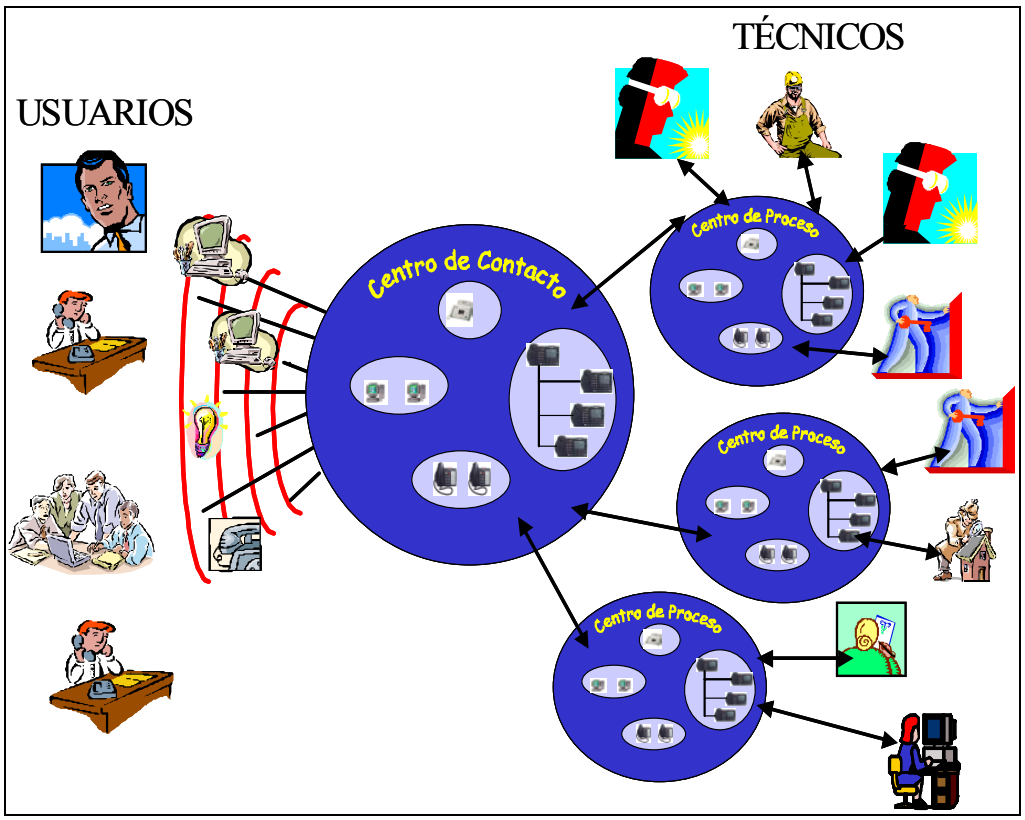


Figura 5.11: Esquema Conceptual del Centro de Contacto.

Modelo Individual

El primer paso es identificar los agentes y sus intenciones. Aquí tenemos cuatro agentes:

- Usuario.
- Centro de Contacto.
- Centro de Proceso.
- Técnico.

Construimos el conjunto de agentes A, con los agentes y sus intenciones:

A = {(Usuario, Solicitar solución efectiva a sus necesidades de información o trámites), (Centro de Contacto, Facilitar la obtención de información confiable y la realización de procesos o trámites de manera impecable para los Usuarios cuidando la relación con los Centros de Proceso), (Centro de Proceso, Realizar el mejor trabajo de acuerdo a los estándares definidos para satisfacer las necesidades del Usuario), (Técnico, Realizar el trabajo asignado de manera impecable y eficiente)}.

Modelo Estructural

La Figura 5.12 presenta el diagrama de agentes derivado del segundo paso, donde se identifican las diferentes conversaciones entre los agentes, como son: *Realiza Petición*, *Evalúa Servicio*, *Solicita Atención*, *Asigna Petición*, *Cierra Petición*.

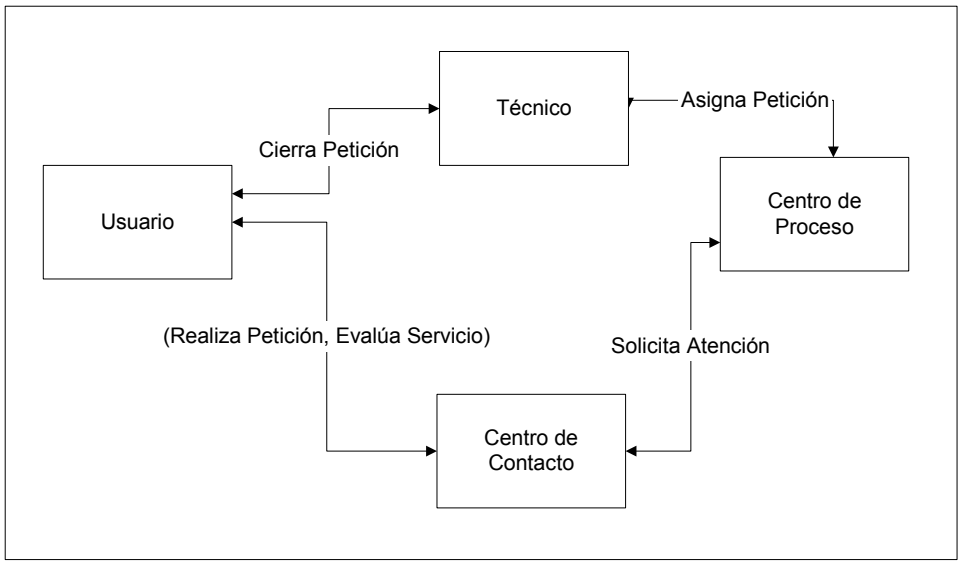


Figura 5.12: Diagrama de Agentes Centro de Contacto.

La interacción principal identificada del sistema es: *Realiza Petición*, la cual está compuesta por cinco conversaciones:

- Usuario *Realiza Petición* Centro de Contacto.
- Centro de Contacto *Solicita Atención* Centro de Proceso.
- Centro de Proceso *Asigna Petición* Técnico.

- Técnico *Cierra Petición* Usuario.
- Usuario *Evalúa Servicio* Centro de Contacto.

Con base en lo anterior, construimos el diagrama de interacciones, el cual se presenta en la Figura 5.13.

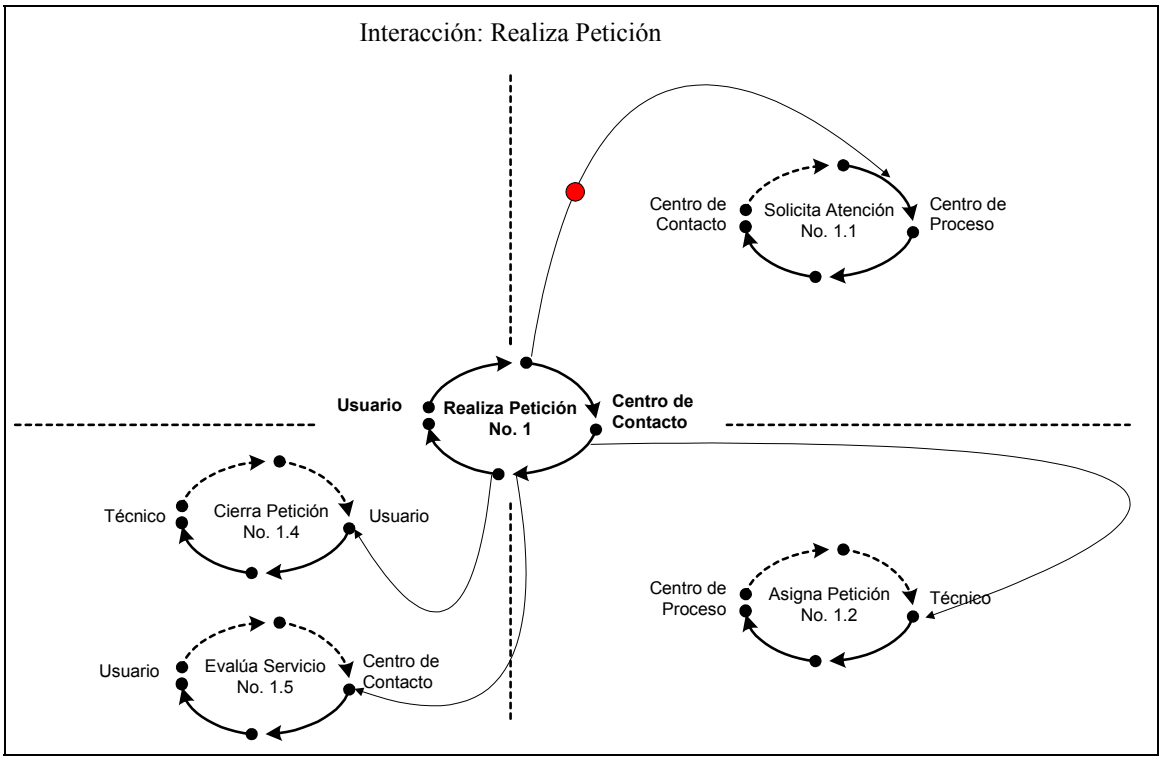


Figura 5.13: Diagrama de Interacción para *Realiza Petición*.

La especificación de la conversación *Solicita Atención* entre el Centro de Contacto y el Centro de Proceso se presenta en las Figuras 5.14 y 5.15. Aquí describimos la información acerca de la interacción y sus conversaciones en las dos partes que componen la forma:

- La Descripción de la Conversación, presentada en la Figura 5.14.
- La Especificación de la Interacción, presentada en la Figura 5.15.

Descripción de la Conversación:

Nombre de la Interacción: Realiza Petición	ID de la Interacción: *1*	Fecha: *Octubre 2001*	Versión *1.0*
Nombre de la Conversación: Solicita Atención		ID de la Conversación: *1.1*	
Objetivo Principal de la Conversación: Brindar solución a la necesitad del Usuario de acuerdo a los estándares establecidos.			
Cliente: *Centro de Contacto*	Proveedor: *Centro de Proceso*		

Figura 5.14: Descripción de la Conversación *Solicita Atención*.

En la especificación de la conversación, se describe el intercambio de actos comunicativos entre los agentes *Centro de Contacto* (con el rol cliente) y *Centro de Proceso* (con el rol proveedor) en cada paso del Ciclo Básico de Acción, *Loop*.

Todas las conversaciones en el diagrama de interacción deben tener una descripción y especificación de la conversación como las presentadas en las Figuras 5.14 y 5.15. Los mensajes son representados por los actos comunicativos, en este caso en particular los de Flores. Como referencia estos actos se presentaron en las Tablas 4.1 y 4.2.

En el modelo de la conversación *Solicita Atención*, el *Centro de Contacto* pide a un *Centro de Proceso* la solución a una petición realizada por el *Usuario*, de acuerdo a los estándares de atención que se definen en el nivel de servicio. El *Centro de Proceso* puede renegociar los tiempos de cumplimiento, pero el *Centro de Contacto* es quien acepta o no la propuesta. Cuando un agente está participando en una interacción, el flujo de la conversación es controlado por el Ciclo Básico de Acción, *Loop*.

Por ejemplo, en la Figura 5.15, en el paso de *Ejecución*, el *Centro de Proceso* inicia la conversación *Asigna Petición* con un *Técnico*, en el estado de Ejecución para la conversación *Solicita Atención* con el *Centro de Contacto*. Posteriormente que cierra la conversación con el *Técnico*, envía el mensaje *P_ReportCompletion* para que la conversación con el *Centro de Contacto* avance a la etapa de *Aceptación*.

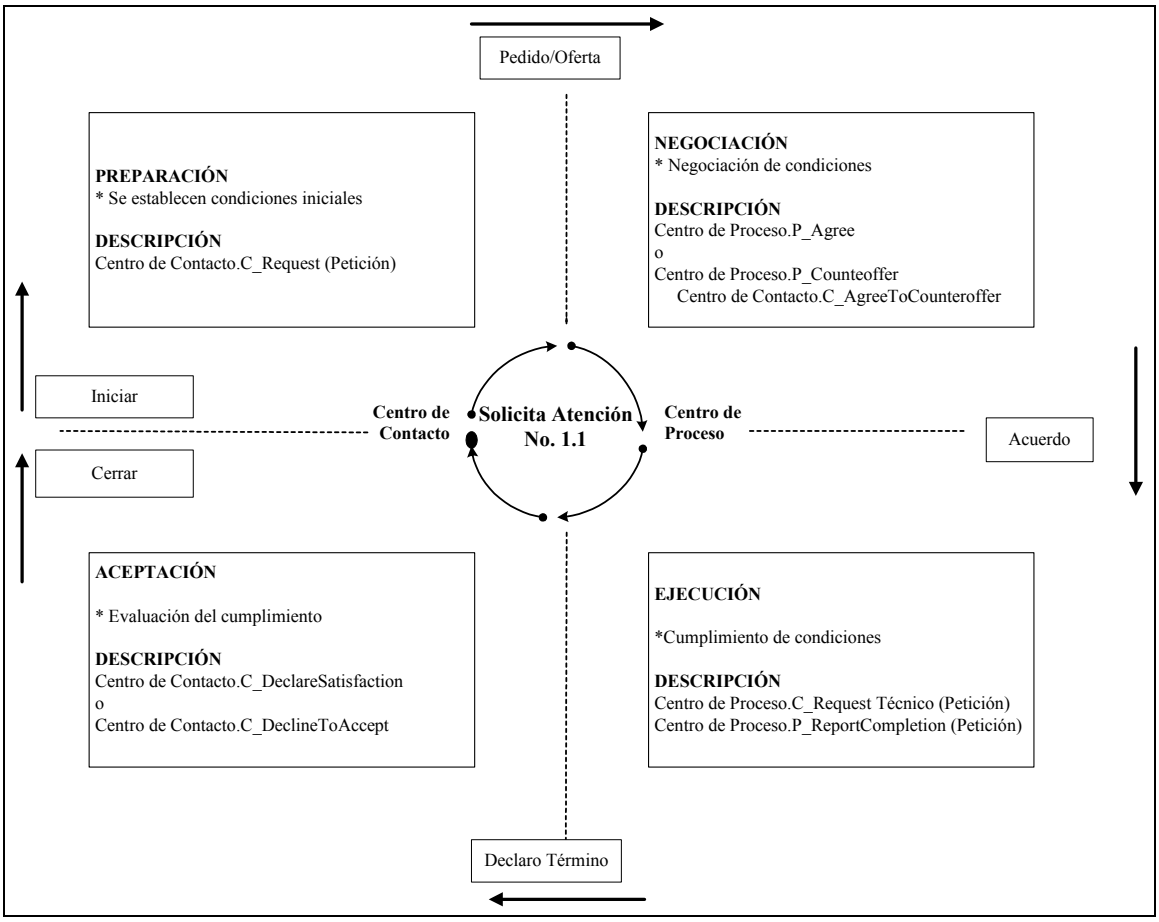


Figura 5.15: Especificación de la Conversación *Solicita Atención*.

Al finalizar este paso, podemos representar el Sistema de Información Cooperante (SIC) por medio de la siguiente expresión: SIC={A, Cg, Gk, I}, donde:

- Conjunto de Agente, *A = {(Usuario, Solicitar solución efectiva a sus necesidades de información o trámites), (Centro de Contacto, Facilitar la obtención de información confiable y la realización de procesos o trámites de manera impecable para los Usuarios cuidando la relación con los Centros de Proceso), (Centro de Proceso, Realizar el mejor trabajo de acuerdo a los estándares definidos para satisfacer las necesidades del Usuario), (Técnico, Realizar el trabajo asignado de manera impecable y eficiente)}*.
- Objetivo Común, Cg= Resolver las peticiones de los usuarios de una manera coordinada y eficiente en un solo punto de atención.
- Conocimiento global, Gk={Petición, Disponibilidad de técnicos, Manuales Técnicos}.
- Conjunto de Interacciones, *I={Realiza Petición}*.

Ahora, diseñaremos a detalle los diferentes puertos del sistema, de acuerdo a la interacción principal:

Interacción:
Realiza Petición (Usuario, Centro de Contacto, Centro de Proceso, Técnico)

Conversaciones:

- {Usuario *Realiza Petición* Centro de Contacto,
- Centro de Contacto *Solicita Atención* Centro de Proceso,
- Centro de Proceso *Asigna Petición* Técnico,
- Técnico *Cierra Petición* Usuario,
- Usuario *Evalúa Servicio* Centro de Contacto}.

Puertos:
Conformados por los actos comunicativos que se utilizan de acuerdo al Ciclo Básico de Acción, *Loop*. Los agentes *Usuario*, *Centro de Contacto, Centro de Proceso* y *Técnico*, juegan dos roles diferentes en una conversación, el de Cliente y el de Proveedor.

Un agente en cada conversación en la que participa tiene dos puertos, el puerto de entrada (Pin) donde recibe los mensajes y el puerto de salida (Pout) a través del cual envía los mensajes. En la Figura 5.16 se presenta el diagrama de puertos.

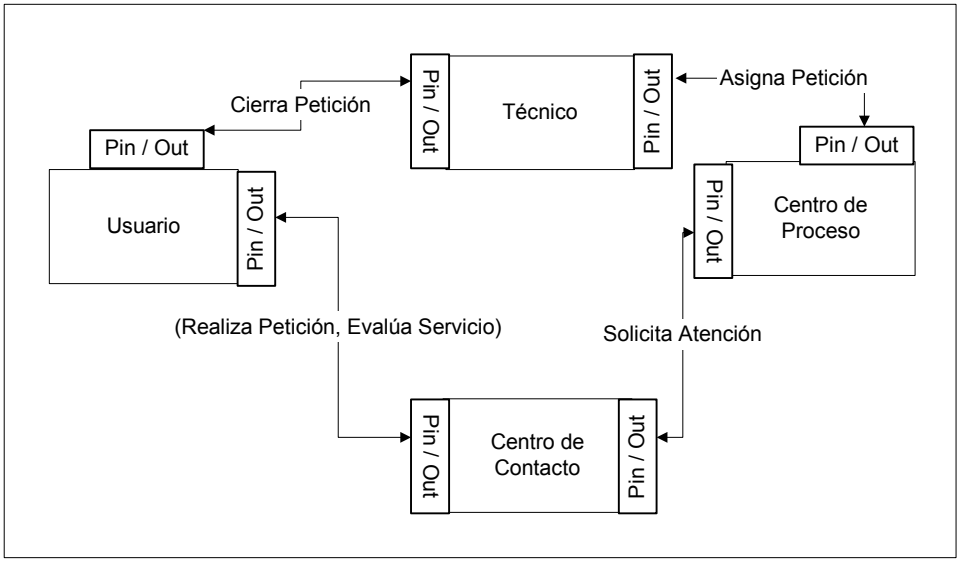


Figura 5.16: Diagrama de Puertos del Ambiente de Centro de Contacto.

Los puertos del *Usuario* son:

- Usuario.Pin $_{\text{Realiza Petición, Evalúa Servicio}}$, Centro de Contacto (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type), Comment).
- Usuario.Pout $_{\text{Realiza Petición, Evalúa Servicio}}$, Centro de Contacto (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to

Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer, Comment).

- Usuario.Pin $_{\text{Cierra Petición}}$, Técnico (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type), Comment).
- Usuario.Pout $_{\text{Cierra Petición}}$, Técnico (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer, Comment).

Los puertos de la *Centro de Contacto* son:

- Centro de Contacto.Pin $_{\text{Realiza Petición, Evalúa Servicio}}$, Usuario = Usuario.Pout $_{\text{Realiza Petición, Evalúa Servicio}}$, Centro de Contacto.
- Centro de Contacto.Pout $_{\text{Realiza Petición, Evalúa Servicio}}$, Usuario = Usuario.Pin $_{\text{Realiza Petición, Evalúa Servicio}}$, Centro de Contacto.
- Centro de Contacto.Pin $_{\text{Solicita Atención}}$, Centro de Proceso (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type), Comment).
- Centro de Contacto.Pout $_{\text{Solicita Atención}}$, Centro de Proceso (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer, Comment).

Los puertos de la *Centro de Proceso* son:

- Centro de Proceso.Pin $_{\text{Solicita Atención}}$, Centro de Contacto = Centro de Contacto.Pout $_{\text{Solicita Atención}}$, Centro de Proceso.
- Centro de Proceso.Pout $_{\text{Solicita Atención}}$, Centro de Contacto = Centro de Contacto.Pin $_{\text{Solicita Atención}}$, Centro de Proceso.
- Centro de Proceso.Pin $_{\text{Asigna Petición}}$, Técnico (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type), Comment).
- Centro de Proceso.Pout $_{\text{Asigna Petición}}$, Técnico (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer, Comment).

Los puertos del *Técnico* son:

- Técnico.Pin $_{\text{Asigna Petición}}$, Centro de Proceso = Centro de Proceso.Pout $_{\text{Asigna Petición}}$, Técnico.
- Técnico.Pout $_{\text{Asigna Petición}}$, Centro de Proceso = Centro de Proceso.Pin $_{\text{Asigna Petición}}$, Técnico.
- Técnico.Pin $_{\text{Cierra Petición}}$, Usuario = Usuario.Pout $_{\text{Esperar por la entrega del producto}}$, Técnico.
- Técnico.Pout $_{\text{Cierra Petición}}$, Usuario = Usuario.Pin $_{\text{Esperar por la entrega del producto}}$, Técnico.

El token de color que representa la interacción es un registro con una lista de Usuarios (*user*), un Centro de Contacto (*cc*), una lista de Centros de Proceso (*pc*), una lista de Técnicos (*lt*), una Petición (*therequest*) y un acto comunicativo que pertenecen a los puertos (*nextaction*). Este token se presenta en la tabla 5.15.

Tabla 5.15: Representación de la Interacción para el Ambiente de Centro de Contacto.

```
color Tinteraction = record
                user:Tlistuser *
                cc: TCCenter *
                pc:TlistPCenter *
                lt: TlistTech *
                therequest: Trequest *
                nextaction: TPorts;
```

La flexibilidad del modelo generado, así como el uso de las Redes de Petri Coloreadas, permite definir tipos de datos complejos para cada componente del token que representa la interacción. En este caso, para el Usuario, Centro de Contacto, Centro de Proceso, Técnico y Petición, se definió su propio token de color y se presentan en las Tablas 5.16, 5.17, 5.18, 5.19 y 5.20 respectivamente.

Tabla 5.16: Representación del Usuario para el Ambiente de Centro de Contacto.

```
color Tuser = record
            id:Tid *
            n:Tname *
            l:Tlocation *
            p:Tphone;

color Tlistuser = list Tuser;
```

Tabla 5.17: Representación del Centro de Contacto.

```
color TCCenter = record
                id:Tid *
                n:Tname;
```

Tabla 5.18: Representación del Centro de Proceso para el Ambiente de Centro de Contacto.

```
color TPCenter = record
              id:Tid *
              n:Tname *
```

```
            t:TlistTech *
            e: Tstandard;

color TlistPCenter = list TPCenter;
```

Tabla 5.19: Representación del Técnico para el Ambiente de Centro de Contacto.

```
color TTech = record
            id:Tid *
            n:Tname *
            l:Tlocation *
            p:Tphone;

color TlistTech = list TTech;
```

Tabla 5.20: Representación de la Petición para el Ambiente de Centro de Contacto.

```
color Trequest = record
            id:Tid*
            d:Tdate *
            h:Thour *
            d:Tdescription;

color Tlistrequest = list Trequest;
```

Las adecuaciones que realizamos al mecanismo de interacción se basan en modificaciones sobre los tokens, como *Tinteraction*, *Trequest*, entre otros, así como cambios en las funciones que calculan el siguiente acto comunicativo, con el objetivo de representar el problema. Resaltamos que no se realizan cambios sobre la estructura de la Red de Petri Coloreada que representa el Ciclo Básico de Acción, *Loop*, por lo que vemos la flexibilidad y robustez del mismo para adaptarse a diferentes situaciones de interacciones múltiples así como a varios dominios de aplicación, como se valido en esta tesis.

Modelo Dinámico

Para observar la dinámica de las interacciones múltiples en el Ambiente de Centro de Contacto se simularon las siguientes situaciones de interacción sobre el modelo, con el objetivo de ver su funcionamiento y la expresividad en la representación del mismo.

Interacciones en un Centro de Contacto, con una petición por usuario:

1. **Reporte básico:** Un usuario, un centro de contacto, *m* centros de proceso y un técnico, sin conflicto.

2. **Interacciones simultáneas:** *n* usuarios, el centro de contacto, *m* centros de proceso y *k* técnicos, sin conflicto.
3. **Reporte básico con conflicto:** Similar a la primera, pero con un conflicto al realizar el reporte, como la no disponibilidad de un técnico, donde se genere una contraoferta a la petición.
4. **Interacciones simultáneas con conflicto:** Similar a la segunda, pero con un conflicto al realizar el reporte, como la no disponibilidad de un técnico, donde se genere una contraoferta a la petición.

Para explicar los resultados observados en la herramienta Design/CPN, vamos a identificar los estados iniciales y finales de las conversaciones.

Estado Inicial:

- *Preparation*

Estados Finales:

- *Closed Satisfied,* termina la conversación de manera satisfactoria para las partes.
- *Closed Canceled,* se cancelo la petición en algún momento de la conversación y las partes están de acuerdo.
- *Closed Declined,* alguna de las partes declino la petición y están de acuerdo.
- *Closed Revoked,* alguna de las partes revoca la petición y están de acuerdo..

Al momento de simular cada una de las situaciones y con fines de análisis, es importante que sea cual sea debe terminar con una marca en alguno de los estados finales. En caso contrario la interacción nunca llega a terminación y esto sugeriría un problema potencial, como un abrazo mortal, falta de algún recurso, entre otros.

Análisis de las simulaciones

La Figura 5.17 presenta la marca inicial de la simulación para múltiples interacciones simultáneas, en este caso tres, en el único estado inicial de las misma, *Preparation*. Analicemos que pasa en el momento que las diferentes interacciones representadas por los tokens se activan y las transacciones se van disparando.

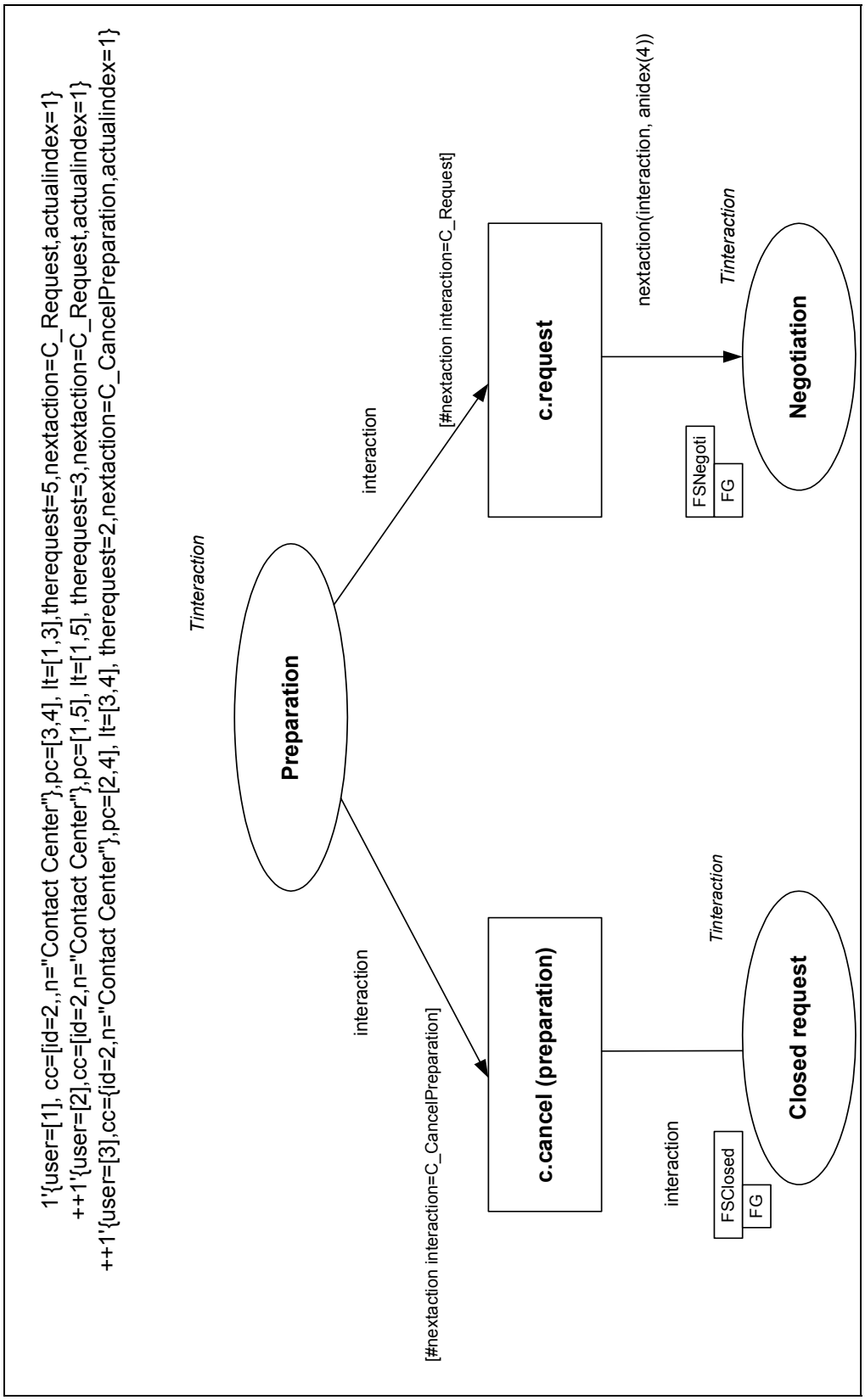


Figura 5.17: Estado Inicial *Realiza Petición*, Múltiples Interacciones Simultáneas.

La Figura 5.18 presenta el estado intermedio de tres interacciones simultáneas, donde podemos observar el estado de las mismas, de acuerdo a los valores que tienen los tokens. Analizando la situación podemos observar como los usuarios, identificados en los tokens con los números 1 y 2, se encuentran en diferentes etapas, tales como contraoferta y aceptación, mientras que el usuario identificado con el número 3 está en algún estado, que no aparece en esta RPC. En este momento podemos observar que las transacciones por habilitarse son las correspondiente a las etapas de contraoferta y aceptación, las cuales se seleccionarian de acuerdo al estado de cada interacción y a actos de habla como los siguientes: *C_CancelNoAgr*, *C_DeclineToAcceptNoAgr* y *C_CancelMakeNewRequest*, en el caso del *Acceptance (no agr)*. En el caso del *Countered*, los actos de habla son: *C_Counter*, *C_DeclineCounteroffer*, *C_CancelMakeNewRequest*, *C_CancelNoAgr*, *C_AgreeToCounteroffer* y *C_DeclareSatisfactionNoAgr*. Podemos concluir de nuestro análisis, que sobre una misma estructura de la Red de Petri Coloreada, podemos modelar y controlar múltiples interacciones, como lo estamos observando en la figura 5.18, de una manera sencilla y expresiva.

La Figura 5.19 presenta el estado final de una interacción de las anteriores, donde se puede observar de acuerdo a la marca final. Podemos concluir, con base en el análisis de las interacciones del sistema, que los usuarios identificados con los números 2 y 3, cancelaron su petición y las dos partes, tanto el comprador como el proveedor estuvieron de acuerdo con la decisión. El usuario 1 se encuentra en algún estado que no se representa en la RPC mostrada.

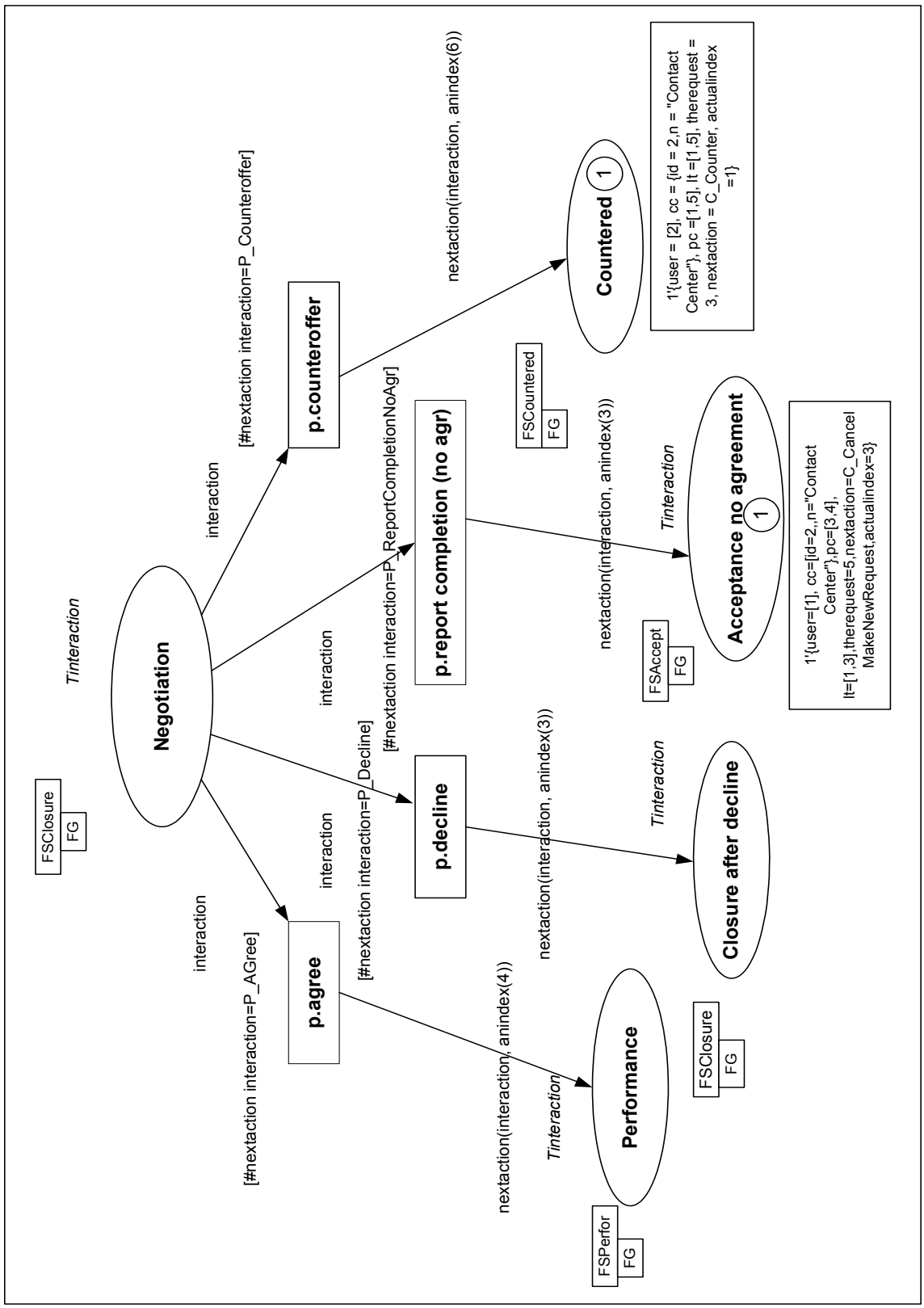


Figura 5.18: Estado Intermedio *Realiza Petición*, Múltiples Interacciones Simultáneas.

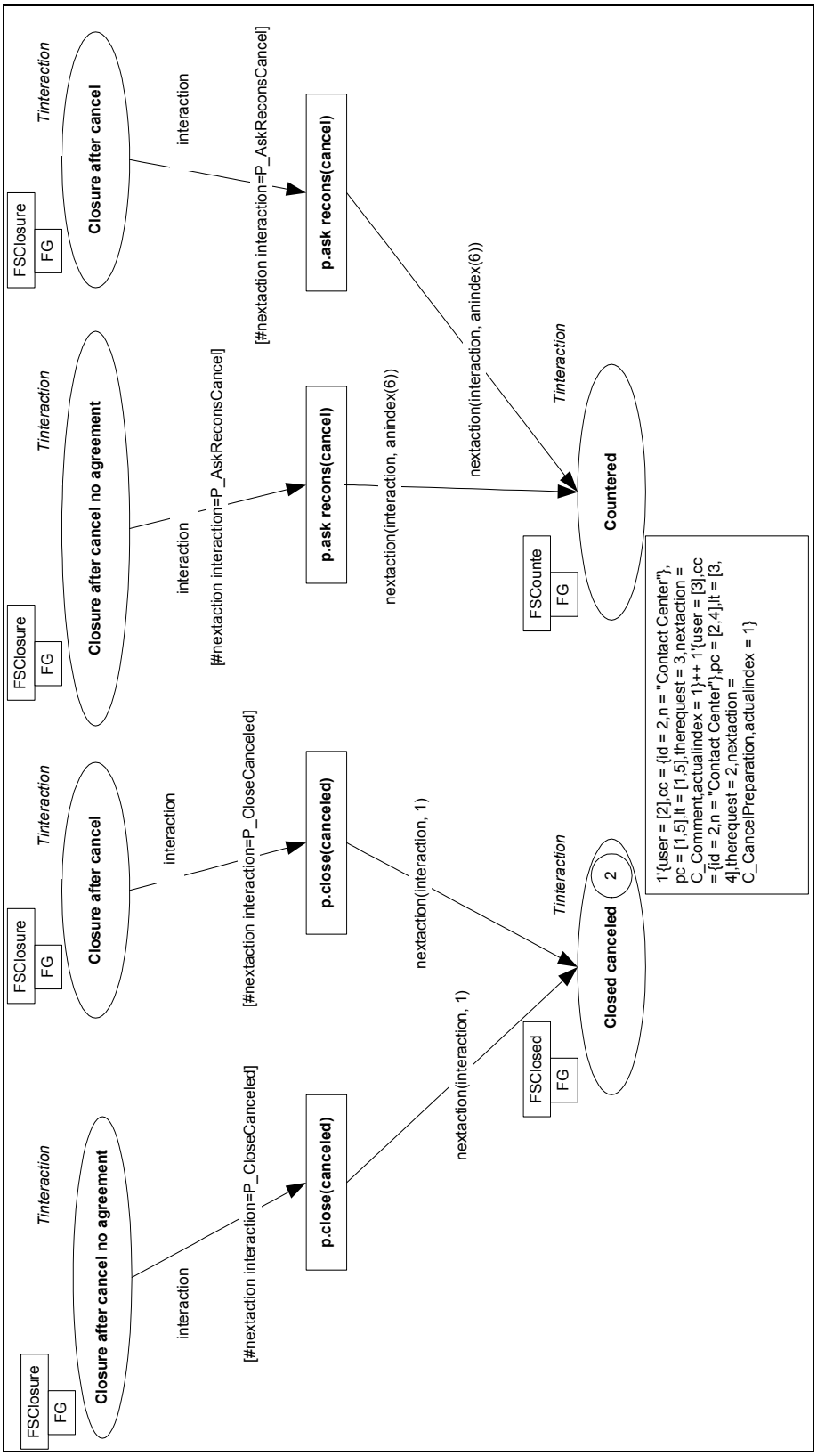


Figura 5.19: Estado Final *Realiza Petición*, Múltiples Interacciones Simultáneas.

Capítulo 6

Conclusiones y Trabajo Futuro

La contribución central de esta tesis es un modelo expresivo de interacciones múltiples simultáneas en sistemas de información cooperantes, bajo un marco formal. El modelo está basado en una representación a varios niveles de cooperación entre agentes, que permite identificar el mecanismo de interacción y su rol como puente entre los protocolos de comunicación y los mecanismos de coordinación. Además, al representar varias interacciones simultáneas en el modelo propuesto, se obtiene una reducción en la dificultad de la modelación, cuya validación se realizó en dos casos de aplicación.

6.1. Conclusiones

Los trabajos actuales investigados y presentados en esta tesis presentan modelos que son aplicables, de manera sencilla a la interacción entre dos agentes, pero que tienen limitantes al querer modelar interacciones múltiples de una manera sencilla y expresiva.

Entre las limitantes encontradas en los enfoques actualmente utilizados para modelar interacciones en SIC´s, están las siguientes: la representación sencilla y expresiva de protocolos complejos con múltiples interacciones simultáneas; la facilidad de adaptación a cambios en los requerimientos del sistema y el hecho que son insuficientes para modelar los cambios de estado en el tiempo, lo cual se relaciona con la dinámica del comportamiento del sistema.

Este trabajo de tesis provee un modelo de interacciones múltiples en Sistemas de Información Cooperantes utilizando Redes de Petri Coloreadas, con el objetivo que los productos obtenidos sean expresivos. Además, el modelo ayuda a reducir y manejar la dificultad asociada con la representación de la dinámica de los sistemas.

El uso de una metodología de modelación genera disciplina en los ingenieros de software y provee de un mecanismo de comunicación común entre ellos. El modelo permite especificar y representar los Sistemas de Información Cooperantes con un enfoque hacía las aplicaciones y la independencia de dominio, para posteriormente razonar sobre los modelos obtenidos.

El modelo propone representar las relaciones de interacción entre los agentes por medio de los Diagramas de Interacción y formalmente especificarlas utilizando Redes de Petri Coloreadas. Parte fundamental de este trabajo de investigación consistió en validar el potencial de uso del formalismo de las RPC, para verificar si ofrece ventajas en la modelación de interacciones múltiples en sistemas de información cooperantes. Podemos concluir que con el uso de las RPC en el modelo propuesto se tienen las siguientes ventajas:

- Facilita la modelación del estado de las interacciones simultáneas entre más de dos agentes.
- Facilita la representación de varios mensajes, para diferentes agentes, en diversos estados.
- Facilita la simulación de la dinámica de las interacciones para monitorear el comportamiento del sistema.

Uno de los problemas fundamentales en la modelación de la interacción en SIC´s es el relacionado con el mecanismo de interacción. El utilizar diversos mecanismos incrementa la dificultad para la modelación, por lo que en este trabajo de investigación nos enfocamos a determinar la factibilidad de utilizar esquemas probados en las organizaciones, como el Ciclo Básico de Acción, *Loop*, con el objetivo de modelar las interacciones representadas en los Sistemas de Información Cooperantes. Concluimos que el utilizar el *Loop* como mecanismo de interacción nos ayuda a entender y representar diferentes situaciones mediante la utilización de un lenguaje de comunicación y coordinación de acciones común, basado en los actos de habla de FIPA o de Flores.

Se propuso en esta tesis, una arquitectura de capas que integra diferentes herramientas, permitiendo construir el modelo de comportamiento del SIC a partir de la especificación explícita y observar la evolución de las interacciones del sistema a partir de la especificación implícita. Las especificaciones explícitas e implícitas permiten al ingeniero de software manejar la estructura y la dinámica del SIC reduciendo la complejidad en la modelación.

Elementos de las RPC que ayudan a manejar la complejidad en la modelación de la interacción en SIC son:

- Las Plazas de Fusión (*Fusion Place* en la notación de RPC), para modelar el sistema jerárquicamente y representar el medio de comunicación.
- Los Colores, para representar tipos de datos complejos y por medio del manejo de los mismos modelar y controlar interacciones múltiples simultáneas entre agentes.
- El uso de *Design/CPN*, la herramienta para representar y simular Redes de Petri Coloreadas, con el objetivo de entender la dinámica de las interacciones múltiples en sistemas de información cooperantes.

La modelación del Ciclo Básico de Acción, mecanismo de interacción utilizado, permite incorporar, de una manera sencilla, modelos que representen diversos dominios de aplicación a partir de los tokens de colores, como se presentó en el trabajo.

Los dominios de aplicación en los que se ha validado el modelo son los siguientes:

- *Negocios Electrónicos*, en especial en el área de Negocio a Negocio (B2B, de las siglas en Inglés Business to Business), modelando un ambiente de Plaza de Mercado (Marketplace en Inglés), donde se dan diversas interacciones simultáneas entre los participantes como: compradores, proveedores y la plaza de mercado, que necesitan ser representadas y controladas.
- *Centros de Contacto* (Contact Center en Inglés), los cuales son espacios únicos de atención a clientes, que requieren de sistemas de alta coordinación que permitan atender y controlar múltiples interacciones simultáneas entre los clientes, los diferentes centros de proceso y los técnicos asignados a éstos.

En la validación del modelo en las diferentes aplicaciones observamos que los productos obtenidos son expresivos para el Ingeniero de Software.

6.2. Trabajo Futuro

La aplicabilidad del modelo en más dominios de aplicación, es un trabajo a realizar, para incorporar una gama más amplia de SIC´s y validar el modelo de interacciones múltiples y no solamente en ambientes donde se tengan relaciones Cliente- Proveedor.

Como trabajo futuro, dentro del área de negocios electrónicos, las aplicaciones de Negocio a Consumidor (B2C, de las siglas en Inglés Business to Consumer), poseen características relevantes para nuestro trabajo, como son la alta dependencia de las decisiones del usuario en un ambiente de interacción intenso, además que los modelos de B2C requieren utilizar intervalos de tiempo para ciertos procesos. Con este tipo de aplicaciones probablemente necesitamos considerar la modelación de aspectos temporales, que por ahora no son considerados en el marco de referencia presentado, para incrementar la expresividad de los modelos actuales y para explorar el uso de las Redes de Petri Coloreadas como una alternativa de modelación.

El representar las aplicaciones en el modelo propuesto, es un enfoque factible y ofrece ventajas con respecto a otros enfoques, como se justificó en esta tesis. Es necesario explorar diferentes alternativas para representar la toma de decisiones de los agentes durante las interacciones. Algunos ejemplos son: ¿cómo toman decisiones relativas a una negociación?, ¿cómo se puede optimizar la función que calcula el siguiente acto del habla en una interacción?, ¿cómo influye el conocimiento local y global del agente en la toma de decisiones en la interacción?, ¿cómo influyen las creencias, intenciones, objetivos y deseos de un agente en la evolución de la interacción?.

Bibliografía

[Booch 91] G. Booch. "Object Oriented Design With Applications". The Benjamin/Cummings Publishing Company, Inc., California, USA 1991.

[Bradshaw 97] J. Bradshaw, S. Dutfield, P. Benoit and J. Woolley. "KaoS: Toward An Industrial-Strength Open Agent Architecture". Software Agents, Capítulo 17, Menlo Park, California, AAAI Press, 1997.

[Camargo 96] F. Camargo-Santacruz., R. Martínez-Casanova., F. Ramos-Quintana. "Negociación Distribuida y Semi-centralizada en Sistemas Multiagentes: Dos Modelos Propuestos", Memorias de la XII Conferencia Latinoamericana de Informática, CLEI PANEL 96, Bogotá, Colombia, 1996.

[Camargo 2001] F. Camargo-Santacruz, J. Frausto-Solis and F. Ramos-Quintana. "An Expressive Coloured Petri Nets Methodology Applied To A Business To Business Environment", submitted to Electronic Commerce, Research and Applications, http://www.elsevier.nl, 2001.

[Cheikes 93] B. Cheikes. "Methodological Issues in the Design of Intelligent and Cooperative Information Systems", 0-8186-3135-X/93, IEEE, 1993.

[Christensen 2001] S. Christensen, Aarhus University, Associate Profesor, Departament of Computer Science, http://www.daimi.au.dk/~sorenchr, Denmark, 2001.

[Cohen 90] P. Cohen and H. Levesque. "Rational Interaction as a Basis for Communication", Cohen P. R., Morgan, J., and Pollack, M. E. (eds.), in Intentions in Communication, SDF Benchmark Series, MIT Press, pp. 221-255, 1990.

[Cohen 95] P. Cohen and H. Levesque. "Communicative Actions for Artificial Agents", Proceedings of the International Conference on Multi-Agent Systems, AAAI Press, San Francisco, June, http://www.cse.ogi.edu/CHCC/Personnel/pcohen.html, 1995.

[Cordero 99] J. Cordero and M. Toro. "A Components Model Based on Interaction-Nets", ISAS/SCI 1999, Orlando, Florida, 1999.

[Cost 99a] R. Cost, Y. Chen, T. Finin, Y. Labrou and Y. Peng, "Modeling Agent Conversations with Coloured Petri Nets", To appear in Working Notes of the Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents ´99, Seattle, WA, May 1999.

[Cost 99b] R. Cost, Y. Chen, T. Finin, Y. Labrou and Y. Peng, "Using Coloured Petri Nets for Conversation Modeling ", IJCAI ´99, 1999.

[Cost 99c] R. Cost, Y. Chen, T. Finin, Y. Labrou and Y. Peng, "A Negotiation-based Multi-agent System for Supply Chain Management ", in Working Notes of the Agents ´99 Workshop on Agents for Electronic Commerce and Managing the Internet-Enable Supply Chain, Seattle, WA, April 1999

[Deloach 2001] S. Deloach, M. Wood and C. Sparkman. "Multiagent System Engineering", International Journal of Software and Knowledge Engineering, Vol 11, No 3(2001), 231-258, World Scientific Publishing Company, 2001.

[Demazeau 98] Y. Demazeau, J. Koning and G. Françoise. "Formalization and Pre-validation for Interaction Protocols un Multiagent Systems", Distributed AI and Multiagent Systems, p- 298-302, 1998.

[Demazeau 2001] Y. Demazeau. "MAGMA Group Home Page", http://www-leibniz.imag.fr/MAGMA, 2001.

[Decker 95] K. Decker. "Environment Centered Analysis and Design of Coordination Mechanisms", Tesis de Doctorado en la Universidad de Massachusetts Amherst, 1995.

[El Fallah 96] A. El Fallah and S. Haddad. "A Recursive Model for Distributed Planning", ICMAS-96, p.307-314, 1996.

[El Fallah 98] A. El Fallah, S. Haddad and H. Mazouzi, "Observation Répartie et Analyse Des Interaction Dans un Système Multi-agents", JFIADSMA-98, Eds Hermès, Nancy 1998.

[El Fallah 99a] A. El Fallah, S. Haddad and H. Mazouzi, "Une Demarche Méthodologique Pour L´ingénierie Des Protocols D´interaction", JFIADSMA-99, Eds Hermès, Nancy 1999.

[El Fallah 99b] A. El Fallah, S. Haddad and H. Mazouzi, "Protocol Engineering for Multi-agent Interaction", 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW´99, Springer, 1999.

[Fipa 2001] FIPA, Foundation for Intelligent Physical Agents, "Agent Communication Language Specification", http://www.fipa.org, 2001.

[Flores 86] F. Flores and T. Winograd. "Understanding Computer and Cognition, a New Foundation for Design", Addison Wesley, 1986.

[Flores 93a] F. Flores. "Conversaciones para la Acción", Business Design Associates, Inc., 1993.

[Flores 93b] F. Flores. "Estados de Animo", Business Design Associates, Inc., 1993.

[Flores 96] F. Flores, "Introducción al Ciclo Básico de la Acción *Loop*", Business Design Associates, Inc., 1996.

[Frausto 2001] J. Frausto-Solis, F. Camargo-Santacruz and F. Ramos-Quintana. "An Application of an Expressive Coloured Petri Nets Modeling Methodology to a Business to Business Environment", Modelling of Objects, Components, and Agents, MOCA'01, Aarhus, Denmark, August 2001.

[Haddadi 98] A. Haddadi. "Towards a Pragmatic Theory of Interactions", afsaneh@Dbresearch-berlin.de, Morgan Kaufmann Publishers, San Francisco California, Estados Unidos de América, 1998.

[Huhns 98] M. Huhns and M. Singh. "Readings in Agents", Morgan Kaufmann Publishers, San Francisco California, Estados Unidos de América, 1998.

[Huhns 2001] M. Huhns. "Interaction-Oriented Software Develpment", International Journal of Software and Knowledge Engineering, Vol 11, No 3(2001), 259-279, World Scientific Publishing Company, 2001.

[Hong 2001] J. Hong and D. Bae. "Incremental Scenario Modeling Using Hierarchical Object-Oriented Petri Net", International Journal of Software and Knowledge Engineering, Vol 11, No 3(2001), 357-386, World Scientific Publishing Company, 2001.

[Jennings 95] N. Jennings and H. Wooldidge. "Intelligent Agents: Theory and Practice", The Knowledge Engineering Review, 10 (2), p. 115-152, 1995.

[Jensen 97a] K. Jensen. "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", volumen 1, segunda edición, Springer, Alemania, 1997.

[Jensen 97b] K. Jensen. "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", volumen 2, Springer, Alemania, 1997.

[Jensen 97c] K. Jensen. "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", volumen 3, Springer, Alemania, 1997.

[Jensen 2001] K. Jensen. "Coloured Petri Nets Home Page", University of Aarhus, Denmark, http://www.daimi.aau.dk/~kjensen/, 2001.

[Krovi 99] R. Krovi and M. Lind. "Organizations as Societies of Agents? Some Modelling Considerations", [krovir, lindm] @aurora.ncat.edu, 1999.

[Laudon 2000] K. Laudon and J. Laudon. "Management Information Systems", sixth edition, Prentice Hall, New Jersey, 2000.

[Lemaître 99a] C. Lemaître, C. Excelente and A. El Fallah. "Multi-Agent Organization Approach to Electronic Business Automation", http://www.lania.mx, 1999.

[Lemaître 99b] C. Lemaître and C. Excelente. "Multi-Agent Network for Cooperative Work", Expert Systems With Applications: An International Journal, Elsevier Science Publishers, Vol. 14, pp. 117-127, 1999.

[Lesser 98a] V. Lesser. "Reflections on the Nature of Multi-agent Coordination and its Implications for the Agent Architecture", Autonomous Agents and Multi-agent Systems, Kluwer academic publishers, 1, 89-111, http://mas.cs.umass.edu/index.shtml, Julio de 1998.

[Lesser 98b] V. Lesser and T. Sandholm. "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework", Morgan Kaufmann Publishers, San Francisco California, United States of America, 1998.

[Lesser 2001] V. Lesser. "University of Massachusetts Amherst,The Multi-Agent Systems Laboratory", página hogar, http://mas.cs.umass.edu/index.shtml, 2001.

[Levesque 91] H. Levesque and P. Cohen. "Teamwork", Nous 25(4), Special Issue on Cognitive Science and Artifical Intelligence, pp. 487-512. To appear in: Handbook of MultiAgent Systems, http://www.cs.utoronto.ca/DCS/People/Faculty/hector.html, 1991.

[Liñán 98] J. Liñán. "Una Contribución a los Modelos de Comunicación entre Agentes Cooperantes". Tesis de Doctorado en Ciencias Computacionales. ITESM, Campus Morelos, México, 1998.

[Malpica 2000] D. Malpica, F. Reyna, L. Galán, M. Ramírez y Otros. "Mapeo y Rediseño de Procesos", Departamento de Sistemas de Información, Dirección de Informática, ITESM, Campus Estado de México, http://sidi.cem.itesm.mx/sidi, 2000.

[Moldt 97] D. Moldt and F. Wienberg, "Multi-agent Systems Based on Coloured Petri nets", Proceedings of the 18th International Conference on Application and Theory of Petri Nets (ICATPN´97), number 1248 in Lecture Notes in Computer Science, p-82-101, Toulouse, France, June 1997.

[Morrison 95] J. Morrison and J. George. "Exploring the Software Engineering Component in MIS (Management Information Systems) research", Comunication of the ACM, Vol. 38, No. 7, july 1995.

[Odell 2001] J. Odell, B. Bauer and J. Müller. "Agent UML: A Formalism for Especifying Multiagent Software Systems", International Journal of Software and Knowledge Engineering, Vol 11, No 3(2001), 207-230, World Scientific Publishing Company, 2001.

[Papazoglou 98] M. Papazoglou and G. Schlageter. "Cooperative Information Systems, Trends and Directions", Academic Press, San Diego, 1998.

[Ramos 95] F. Ramos, G. Sanchez, E. Espinoza and M. Elliot. "A Fuzzy Temporal Reasoning Mechanism for Planning in Multi-Agent Domains", Third IASTED International Conference in Robotics and Manufacturing, June 14 -16, 1995, Cancún, México.

[Ramos 96] F. Ramos and J. Liñan. "Un Modelo de Comunicación de Soporte a la Planificación Flexible en un Sistema de Agentes Cooperantes", Memorias de la XII Conferencia Latinoamericana de Informática, CLEI PANEL 96, Bogotá, Colombia, 1996.

[Ramos 2001] F. Ramos-Quintana, J. Frausto-Solis and F. Camargo-Santacruz, "A Methodology for Modeling Interactions in Cooperative Information Systems Using Coloured Petri Nets", to appear in the International Journal of Software Engineering and Knowledge Engineering, World Scientific, http://www.ksi.edu/ijsk.html, 2001.

[Reisig 92] W. Reisig. "A Primer in Petri Net Design", Springer-Verlag, Alemania, 1992.

[Rosenschein 94] J. Rosenschein and G. Zlotkin. "Rules of Encounter, Designing Conventions for Automated Negotiation Among Computers", The MIT Press, Cambridge, 1994.

[Rosenschein 98a] J. Rosenschein and G. Zlotkin. "Designing Conventions for Automated Negotiation", Morgan Kaufmann Publishers, San Francisco California, United States of America, 1998.

[Rosenschein 98b] J. Rosenschein, M. Fenster and S. Kraus. "Coordination without Communication: Experimental Validation of Focal Point Techniques", Morgan Kaufmann Publishers, San Francisco

California, United States of America, 1998.

[Senn 90] J. Senn. “Sistemas de Información para la Administración”, grupo editorial Iberoamérica, México, 1990.

[Sycara 91] K. Sycara and M. Lewis. “Modeling Group Decision Making and Negotiation in Concurrent Product Design”, International Journal of Systems Automation: Research and Applications (SARA), Vol. 1, number 3, 1991.

[Sycara 98] K. Sycara and J.Liu. “Multiagent Coordination in Tightly Coupled Task Scheduling”, Morgan Kaufmann Publishers, San Francisco California, United States of America, 1998.

[Tagg 97] R. Tagg and C. Freyberg. “Designing Distributed and Cooperative Information Systems”, International Thomson Computer Press, Londres, 1997.

[Wooldridge 98] M. Wooldridge and N. Jennings. “Formalizing the Cooperative Problem Solving Process”, Morgan Kaufmann Publishers, San Francisco California, United States of America, 1998.

[Wooldridge 2001] M. Wooldridge, N. Jennings and F. Zambonelli . “Organizational Rules”, International Journal of Software and Knowledge Engineering, Vol 11, No 3(2001), 307-314, World Scientific Publishing Company, 2001.

Anexo A

Manual de Diagramación de Procesos

A continuación presentaremos la técnica de diagramación a utilizar [Malpica 2000], la cual adaptaremos para modelar la interacción en sistemas de información cooperantes.

A.1. El Loop

Es la representación gráfica de una conversación. Está compuesto por cuatro flechas en arco formando una elipse. Cada flecha representa una etapa de la conversación. En su conjunto, las flechas en el loop representan el orden de las etapas, y el hecho que una conversación es un ciclo. Esto lo vemos en la figura A.1.

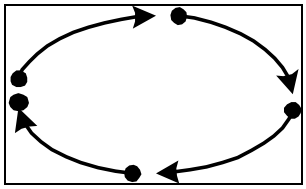


Figura A.1: El Loop.

Una conversación se abre en preparación y fluye en el sentido de las manecillas del reloj. Es factible que una conversación puede empezar en otras etapas.

Los círculos rellenos donde se originan cada una de las flechas forman parte de ésta. Por ejemplo en la figura A.2, el punto B forma parte de negociación mientras que los puntos A y E son casos especiales, pues no son parte de alguna etapa, sino que marcan el inicio y el cierre de una conversación. A es el inicio de la conversación y E el cierre. Es importante notar que no debe confundirse el cierre con la etapa de aceptación.

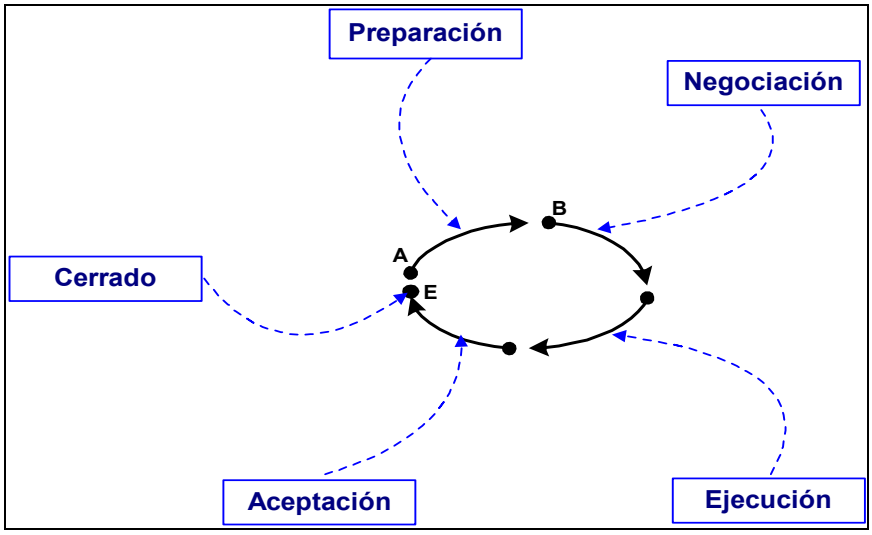


Figura A.2: Etapas en el *Loop*.

En una interacción, el *loop* principal es la conversación que arranca la misma, y que representa la preocupación principal. Varias conversaciones pueden apoyar a que se cumplan las distintas etapas de la conversación principal. Estos *loops* son secundarios y se derivan del principal. El *loop* principal normalmente tiene el mismo nombre que la interacción.

En la figura A.3 podemos ver cómo un par de ejes coordenados con su origen en el *loop* principal dividen al plano en cuatro cuadrantes. Cada uno corresponde a una etapa y tiene su flecha correspondiente.

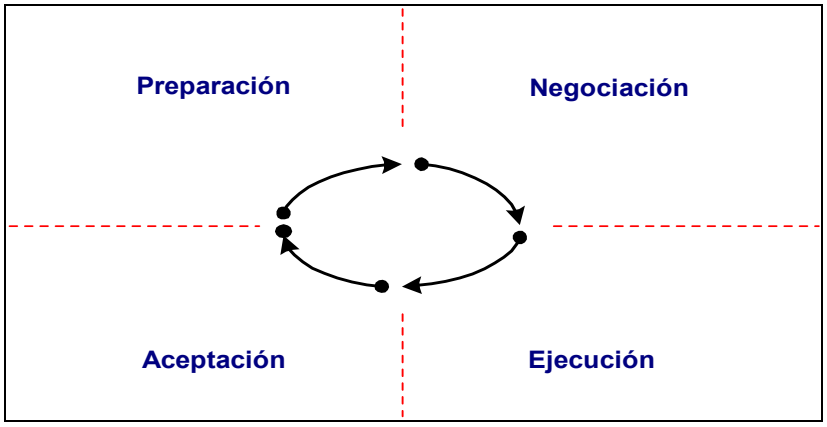


Figura A.3: Cuadrantes en el *loop*.

Para efectos de un diagrama de interacción es recomendable señalar con líneas punteadas los cuadrantes correspondientes al *loop* principal y colocar cada uno de los *loops* secundarios en el cuadrante que les corresponde según la etapa que los origina. Más adelante explicaremos esto a detalle cuando enlacemos varias conversaciones que forman una interacción.

Los Componentes del *Loop*

Los siguientes componentes de una conversación se representan gráficamente en el *loop*, como lo observamos en la figura A.4.

Actores: del lado izquierdo se coloca al cliente y del lado derecho al proveedor.

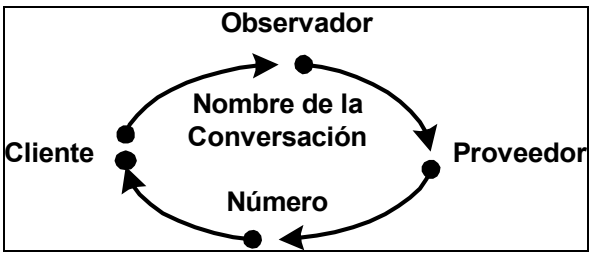


Figura A.4: Representación de un Pedido.

Nombre y número de conversación: se colocan al centro del *loop*. El nombre debe reflejar la inquietud o preocupación principal del cliente de la conversación. Los números de conversación sirven como referencia. Más adelante veremos la convención para numerar conversaciones. La conversación principal es siempre la número 1.

Pedidos vs. Ofertas: en la figura A.4 vimos la representación de un pedido. Para distinguir a una oferta de un pedido se pone un asterisco entre paréntesis del lado derecho del proveedor, como lo vemos en la figura A.5.

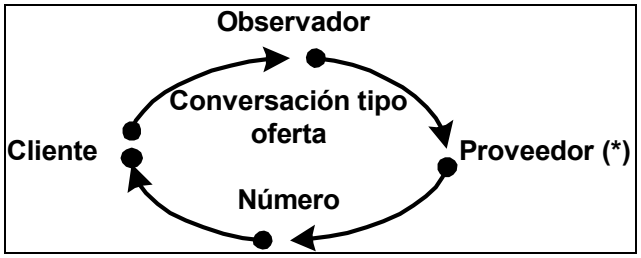


Figura A.5: Representación de una Oferta.

A.2. Enlazando Conversaciones

Un proceso de negocios está formado por una o más conversaciones que se enlazan para articular la coordinación entre personas o áreas. En el diagrama de interacción, debemos representar está situación. Los enlaces, en el diagrama de interacción, están representados por flechas que parten de una conversación desde cierta etapa y llegan a otra conversación en cierta etapa. Un enlace denota, generalmente, el inicio o activación de otra conversación.

En la figura A.6 presentamos un ejemplo de conversaciones enlazadas dentro de una misma interacción.

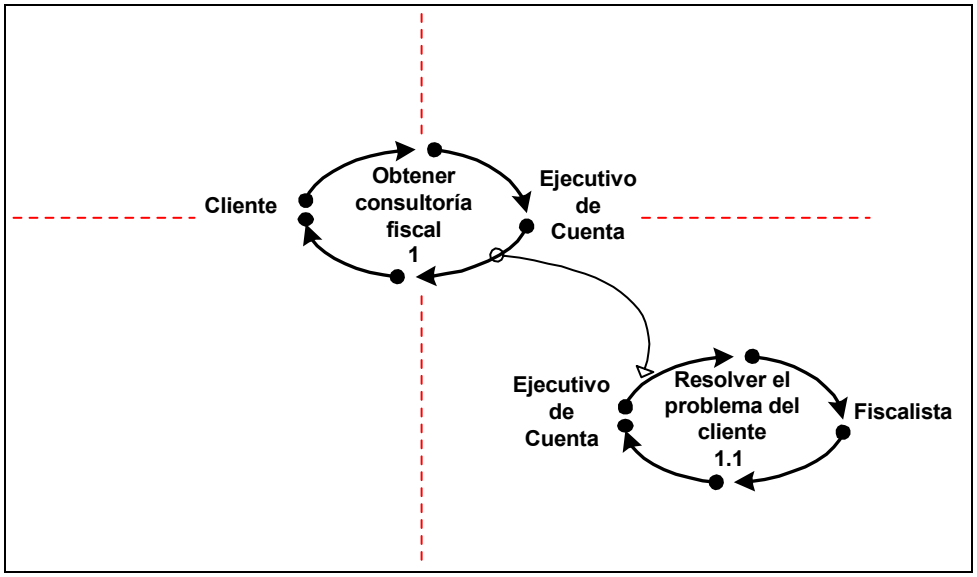


Figura A.6: Conversaciones Enlazadas.

Podemos ver que las dos conversaciones 1 y 1.1 están unidas por una flecha, la cuál representa el enlace o relación entre ambas. La interpretación del diagrama es la siguiente: Cuando la conversación "Obtener Consultoría Fiscal" llegue a ejecución, se iniciará una conversación entre el Ejecutivo de Cuenta y el Fiscalista, llamada "Resolver el Problema del Cliente". A partir de este momento las dos conversaciones continuarán en paralelo.

En esta serie de conversaciones podemos observar como el agente Ejecutivo de Cuenta está interactuando con más de un agente de manera simultánea y debe coordinar el trabajo en todas las conversaciones donde participe.

En un *loop* no hay distinción de orden para los enlaces que parten de una misma etapa por lo que si dos conversaciones se derivan de una, ellas inician al mismo tiempo en la misma etapa.

Notación para Numerar Conversaciones

El número de conversación es una etiqueta que sirve para identificar de manera sencilla las conversaciones y facilita la manipulación al modelar sistemas de información cooperantes complejos. En la figura A.7 se presenta un ejemplo de numeración.

El sistema de numeración es jerárquico, observando las siguientes reglas:

1. La conversación principal se identifica con el número 1.

2. Las conversaciones que se derivan directamente de la principal se numeran 1.1, 1.2, 1.3 y subsecuentes. Las que se derivan de la conversación 1.1 se numeran 1.1.1, 1.1.2 y subsecuentes. Se sigue esta notación en niveles sucesivos.

3. Se debe seguir siempre la secuencia de la numeración, sin brincar números. Por ejemplo, si la conversación 1.2 tiene dos conversaciones derivadas, debemos numerarlas como 1.2.1, y 1.2.2.

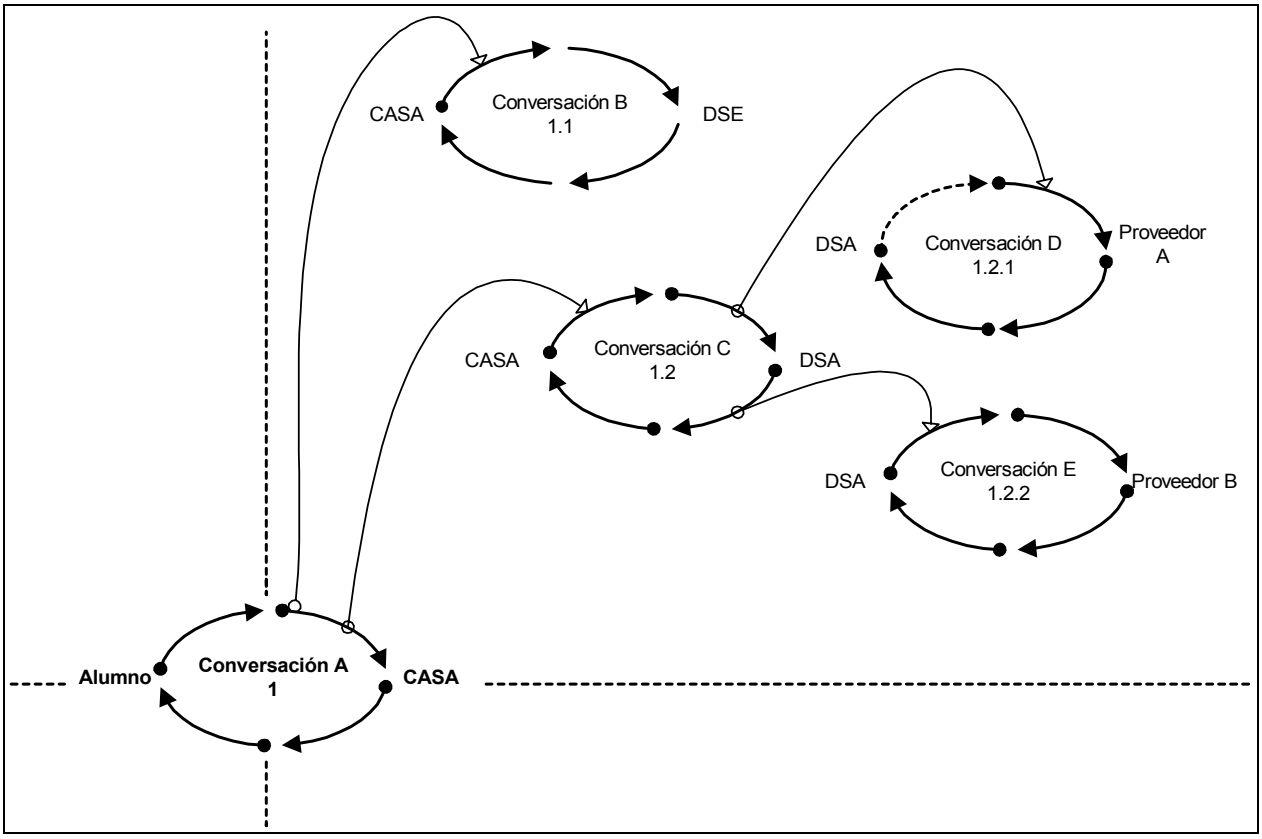


Figura A.7: Ejemplo de Numeración de Conversaciones.

Terminar y Cerrar

Los conceptos de terminación y de cierre de una conversación son necesarios para avanzar en las conversaciones.

Terminar: se dice que una conversación está terminada cuando el cliente declara satisfacción en la etapa de aceptación.

Cerrar: una conversación está cerrada cuando no tiene ningún pendiente. Los actores pueden cerrar una conversación en cualquier etapa de la misma.

Lo opuesto al cierre de una conversación es cuando la conversación está abierta o sea, tiene pendientes.

Una conversación se puede cerrar de distintas maneras:

1. Cuando el cliente declara satisfacción.
2. Cuando el cliente o el proveedor cancelan la conversación.

3. Cuando el proveedor declina un pedido o el cliente una oferta.

El primer caso es especial, pues se refiere a aquéllas conversaciones que son cerradas satisfactoriamente, es decir, cuando el cliente ve resuelta su preocupación. Denominaremos este caso como “Cerrado Terminado”.

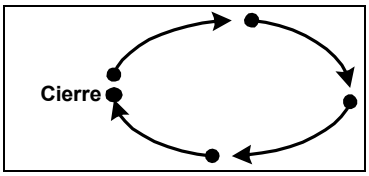


Figura A.8: Cierre de una Conversación.

En el *loop*, el último punto del ciclo representa el cierre de una conversación, es decir, nos indica que una conversación se pudo haber cerrado por cualquiera de las razones enlistadas arriba. Es importante recordar esto al momento de enlazar conversaciones: una flecha que sale del cierre de una conversación indica que el enlace se activa cuando la conversación se cierra, no únicamente cuando se termina. En la figura A.8 se presenta.

Dependencia de Conversaciones

Muchas veces en las interacciones, es necesario tener conversaciones que dependen una de la otra. Dos conversaciones A y B son dependientes cuando se requiere que la conversación B se cierre para que pueda avanzar a la siguiente etapa la conversación A. Entonces, A depende de B.

En la figura A.9 representamos la dependencia entre conversaciones.

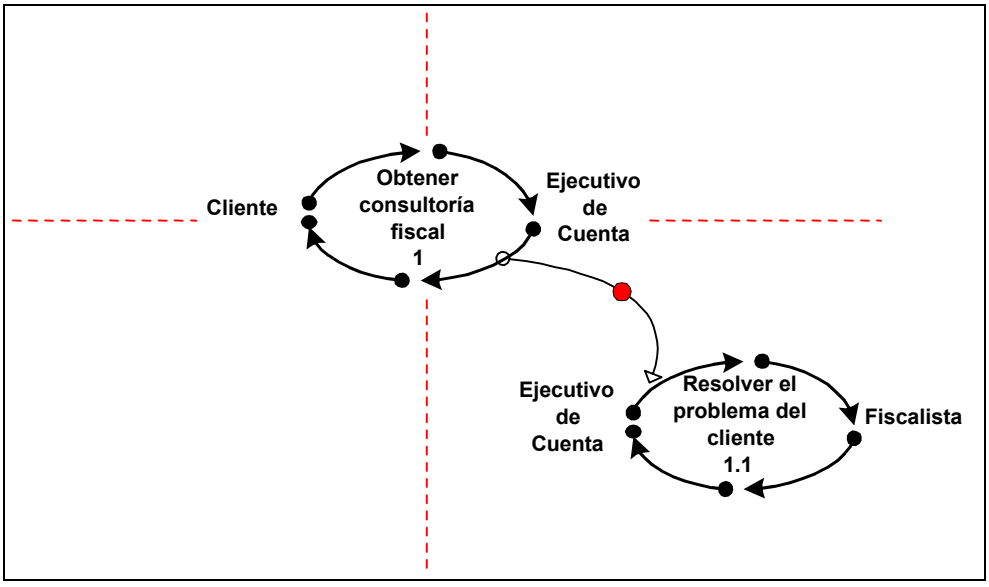


Figura A.9: Dependencia de Conversaciones.

El círculo colocado en el enlace se denomina “espera”, y es la forma de representar gráficamente la dependencia de conversaciones.

Técnicamente una espera se describe de la siguiente forma: “Una espera suprime los actos de habla que producen acción en una conversación, hasta que no se cierre una conversación derivada”.

A.3. Condicionales

El conector condicional sirve para representar las situaciones en las que un enlace entre dos conversaciones se debe tomar solamente cuando se cumple cierta condición en la interacción. En la figura A.10 se presenta el uso de esté conectivo.

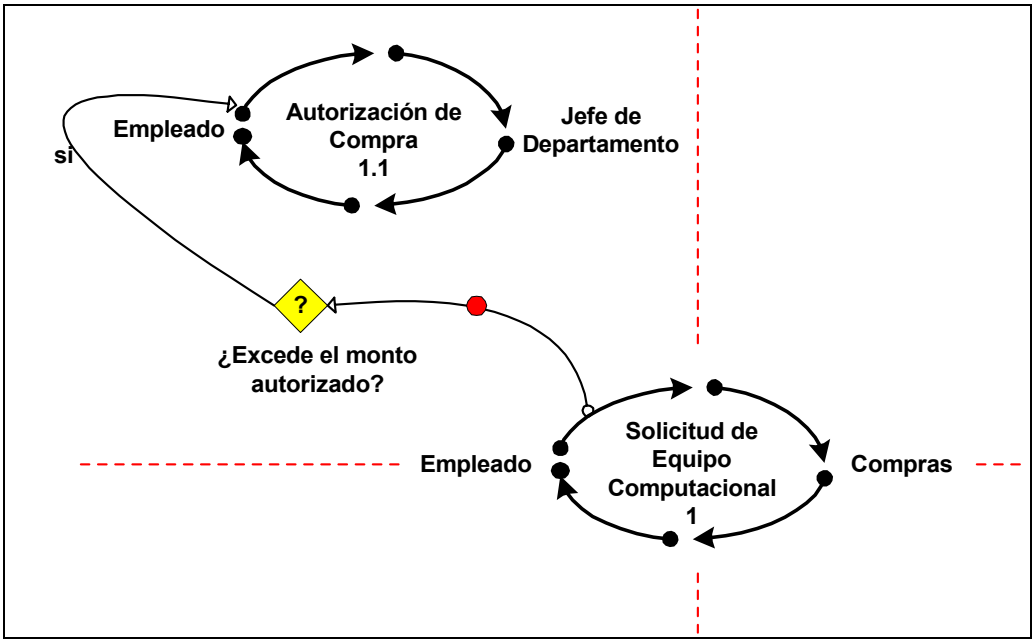


Figura A.10: Representación de los Conectores Condicionales.

Un conector condicional siempre va acompañada de etiquetas, los cuales son textos explicatorios. Primeramente la condición misma, en el ejemplo de la figura A.10, “¿Excede el monto autorizado?”. En segundo lugar, por cada enlace que sale de la condicional, un texto indicando qué valor de la condicional activa el enlace, en el ejemplo, solamente un enlace sale de la condicional, y va acompañado de un “sí”.

Note en el ejemplo que una espera impide al agente empleado hacer la *Solicitud de Equipo Computacional* hasta que tenga la autorización, en caso que su compra exceda el monto autorizado. La espera puede ponerse antes o después de la condicional y el significado del enlace es el mismo.

De una condicional pueden partir tantos enlaces como valores distintos puede arrojar la condición. En la figura A.11 se presenta el siguiente ejemplo: Suponemos que el Centro de

Atención y Servicio a Alumnos (CASA) puede proporcionar a un alumno, el cliente principal, tres tipos de constancias según éste solicite: de beca, de estudios o de servicio social. CASA ejecuta los pedidos obteniendo la constancia con diferentes departamentos, dependiendo del tipo solicitado.

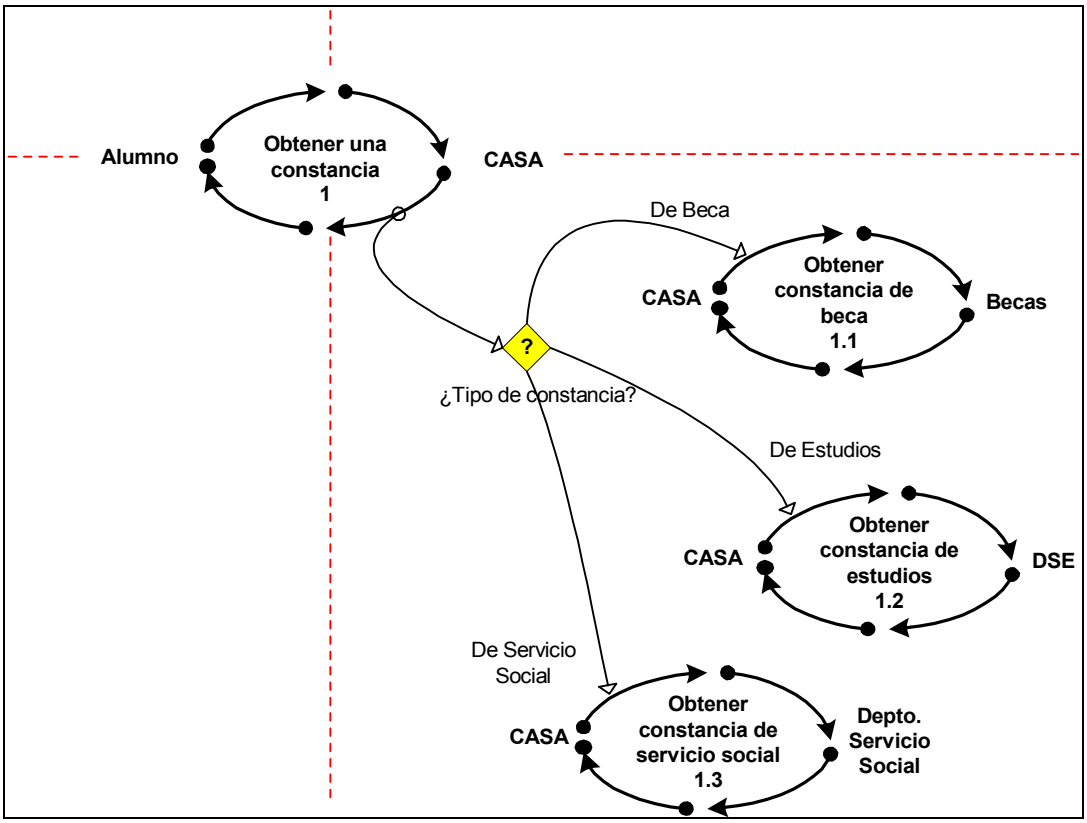


Figura A.11: Representación de Varias Opciones en una Condición.

A.4. Conectores

Un conector sirve para representar un enlace de una a varias conversaciones o de varias conversaciones a una. Tenemos dos tipos: los conectores lógicos, los cuáles se dividen a su vez en conectores "Y" Y y conectores "O" O y los divisores o conectores vacíos .

Los divisores tienen una función puramente estética. Se utilizan para separar un enlace en varias ramas, como lo vemos en la figura A.12. Esto nos permite aumentar la legibilidad de un diagrama de interacción.

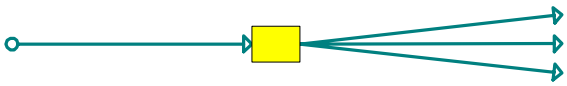


Figura A.12: Uso del Divisor.

Los conectores lógicos, como los vemos en la figura A.13, sirven para juntar dos o más enlaces en uno solo. Cumplen una función estética, pero además son útiles cuando queremos un enlace cuya ocurrencia depende de la ocurrencia de otros enlaces.

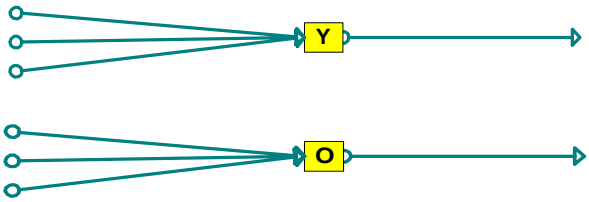


Figura A.13: Uso de los Conectores Lógicos.

- En un conector Y, para que ocurra el enlace que sale del conector, tienen que ocurrir todos los enlaces que llegan al conector.

- En cambio, en un conector O, para que ocurra el enlace que sale del conector, basta con que ocurra uno de los enlaces que llegan al conector.

En la figura A.14 presentamos un ejemplo del uso de los diferentes conectores. Cabe notar que el divisor cumple la función de distribuir un enlace hacia varias conversaciones haciendo el diagrama de interacción más legible. Funcionalmente, el diagrama sería exactamente igual si dos enlaces independientes partieran de la etapa de negociación hacia las conversaciones 1.1 y 1.2, cada una con una espera.

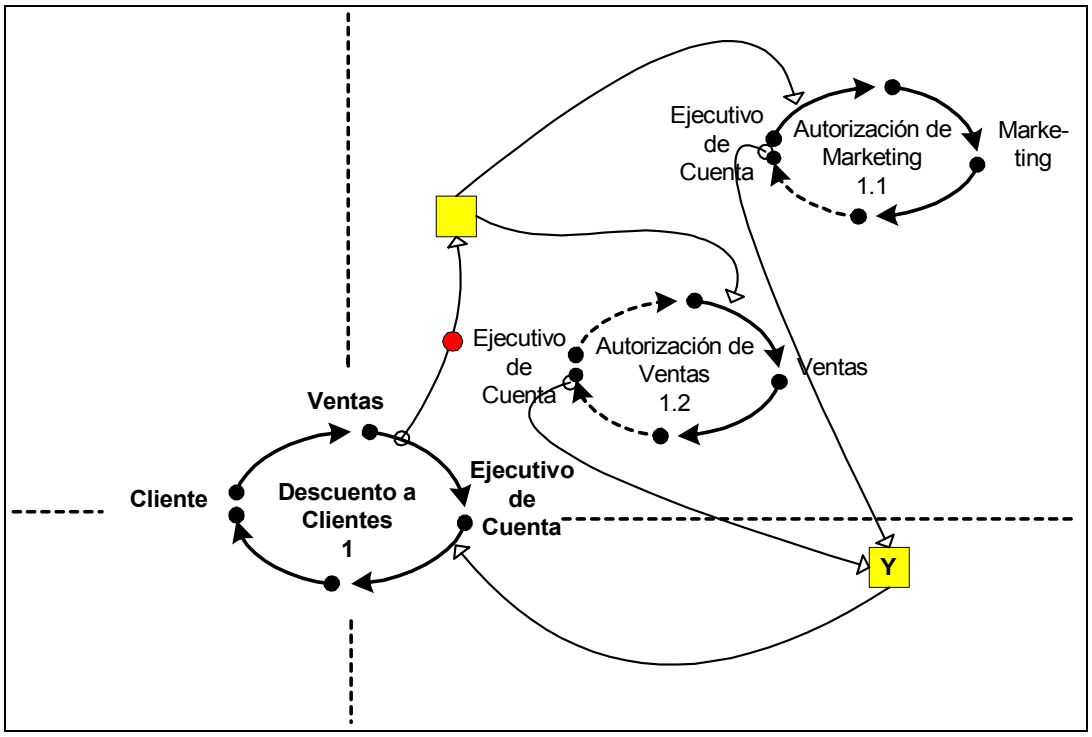


Figura A.14: Ejemplo del Uso de los Conectores.

En la misma figura A.14, el conector junta los dos enlaces en uno, e indica que para que la interacción pueda avanzar, las conversaciones "Autorización de Marketing" **Y** "Autorización de Ventas" deben cerrarse. Si sustituimos el conector "Y" por un conector "O", en el diagrama de interacción de la figura A.14, entonces solamente se requiere que alguna de las dos conversaciones se cierre. En ciertos procesos se presentan conversaciones en las que alguna etapa no tiene uso o razón de ser. En la figura A.14 se representa el concepto por medio de flechas punteadas.

Anexo B

Reglas de Diseño con RPC

A continuación, brindaremos algunas guías para facilitar la construcción de modelos basados en Redes de Petri Coloreadas, inspiradas en [Jensen 97a], con el objetivo que a medida que utilicen los modelos y las RPC se conviertan en ingenieros de software expertos. Para una referencia mayor de Redes de Petri Coloreadas recomendamos la serie completa de [Jensen 97a], [Jensen 97b], [Jensen 97c].

Cuando se inicia en la modelación de RPC se recomienda lo siguiente:

- Analice y modifique modelos simples o medianos creados por otros ingenieros de software.
- Entienda el modelo e intente realizar modificaciones simples sobre el mismo.
- Utilice y modele dominios de aplicación conocidos por usted o que le son familiares.

B.1. Construcción de Modelos de RPC

Para construir modelos de RPC se recomienda:

- Inicie identificando los componentes más importantes del sistema a modelar. Esto se puede realizar generando una lista que contenga agentes del dominio problema, procesos, estados y acciones del sistema.
- Considere el propósito de su modelo y determine el nivel adecuado de detalle. El ingeniero de software debe definir que tan detallado o general realizará el análisis del sistema, de acuerdo al objetivo general del mismo.
- Encuentre nombres que representen adecuadamente y expresivamente los agentes, procesos, estados y acciones.
- No intente cubrir todos los aspectos del sistema analizado en la primera versión del modelo. Éste es un proceso iterativo donde el modelo se va refinando a medida que el ingeniero de software avanza en el análisis y diseño.
- Seleccione uno de los procesos en el sistema que se está modelando y trate de construir una red aislada del mismo. Represente cada estado como una plaza y cada posible cambio de un estado a otro como una transición.

- Utilice la estructura de la red para modelar el control y las inscripciones en la red para modelar el manejo de los datos. La estructura de la red está conformada por las plazas, las transiciones y los arcos, y se puede comparar con las estructuras de control de los lenguajes de programación.
- Diferencie los diversos tipos de tokens, ya que cada uno de ellos representa un tipo de datos, una situación particular, un mensaje y muchas veces los estados lógicos de los agentes.
- Use diferentes tipos de conjuntos de colores, con el objetivo de optimizar la representación del sistema y la simulación del mismo.
- Refine los procesos de la red describiendo como se comunican o interactúan con otros procesos.
- Encuentre procesos similares, con el objetivo de representarlos en una sola red y reutilizar modelos. Este proceso nos ayuda a reducir el tamaño de la RPC.
- Combine las diferentes subredes de un proceso en un modelo general. Esto nos apoya en técnicas de modelación de sistemas jerárquicamente como de arriba abajo (top-down) o de abajo hacia arriba (bottom-up), para modelar sistemas complejos grandes.
- Desarrolle los modelos de RPC de forma similar a la construcción de un programa de cómputo, de tal manera que utilice las mejores prácticas para generar un modelo entendible y ordenado.

B.2. Dibujando Diagramas de RPC

Una parte clave para el ingeniero de software es la habilidad que desarrolle a medida que realice diagramas de Redes de Petri Coloreadas. En la medida que adquiera experiencia sus modelos serán mejores y más sencillos de entender, lo que generará una mayor expresividad de los mismos. El construir diagramas de RPC es análogo a programar, y técnicas ya utilizadas pueden aplicarse acá, como por ejemplo:

- Nombres nemotécnicos.
- Estrategia de presentación consistente y transparente.
- Comentarios, tanto textuales como gráficos.
- Consistencia.

Como una regla general para dibujar las RPC, podemos hablar que si el modelo es estéticamente agradable posiblemente será sencillo de leer y entender, y viceversa, pero además se recomienda utilizar los siguientes efectos gráficos, en lo posible:

- Posición, relativa a los otros objetos.
- Formas (rectángulo/ elipse/ caja redondeada/ arcos rectos o curveados/ diferentes formas de flechas, entre otros.).
- Tamaño de los nodos, textos y flechas de los arcos.
- Grueso de las líneas en los nodos y los arcos.
- Apariencia del texto en lo referente a tipo de letra, tamaño, estilo y alineación.
- Colores de las líneas, interiores y textos.

- Esquema de distribución de todos los objetos.

A continuación presentamos algunas recomendaciones para dibujar las Redes de Petri Coloreadas, para mayor información recomendamos consultar a [Jensen 97a].

- Agrupe los arcos de entrada de un lado del nodo y los arcos de salida en el lado opuesto.
- Distribuya los nodos y arcos del modelo de tal manera que formen una figura geométricamente agradable.
- Mantenga la misma dirección de los flujos.
- Evite los arcos cruzados o sobrepuestos.
- Utilice los arcos de tal manera que su trazo sea regular y simple.
- Evite dibujar arcos paralelos con una distancia mínima entre ellos.
- Dibuje las líneas punteadas correctamente.
- Evite figuras geométricas que puedan confundirse con las formas básicas, como plazas, transiciones u otros elementos.
- Coloque las inscripciones de la red lo mas cerca posible de la plaza, transición o arco correspondiente.
- Sea lo más consistente posible.

Anexo C

Notación del Modelo

C.1. Especificación Explícita

- **Modelo Individual:**

1. Identificar agentes y sus intenciones

1.1. Actividades a Realizar: Identificar las abstracciones clave del sistema y sus intenciones asociadas, las cuales son candidatas a ser agentes.

1.2. Productos a Obtener: un conjunto de agentes *A*, de la forma:

- $A=\{(a_1,C_1), ..., (a_m,C_m)\}$, donde la dupla (a_j,C_j), representa al agente a_j con su conjunto de intenciones C_j, con $j=1..m$, donde m es el número total de agentes en el sistema.
- El conjunto de intenciones: $C_j = \bigcup_{i=1}^{r} cj_i$ donde cj_i es la intención i del conjunto C_j, con $i=1..r$, donde r es el número total de intenciones del agente a_j, y $j=1..m$, donde m es el número total de agentes en el sistema.

1.3. Diagramas a Realizar: Ninguno.

- **Modelo Estructural:**

2. Construir el diagrama de agentes

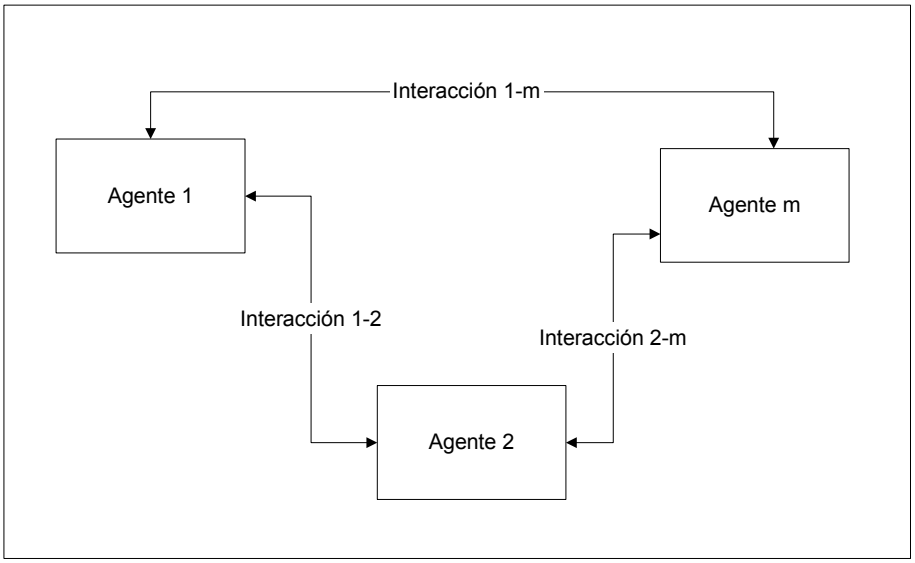
2.1. Actividades a Realizar: Construir el diagrama de agentes que represente la arquitectura del sistema de información cooperante. La estructura del diagrama se encuentra determinada por los diferentes agentes del sistema y las interacciones en las que estos participan

2.2. Productos a Obtener: El diagrama de agentes, que es un grafo dirigido bi-direccional, representado por la dupla DA=(A,R), donde como se explico en el capítulo 4, se satisfacen los siguientes requerimientos:

(i) A es un conjunto finito de Agentes.

(ii) R es un conjunto finito de Relaciones de Interacciones entre los agentes de A.

2.3. Diagramas a Realizar: Diagrama de Agentes:



3. Construir los diagramas de interacción

3.1. Actividades a Realizar: Describir las diferentes conversaciones del sistema, partiendo de la interacción principal, la cual debe estar alineada con el *Objetivo Común* del sistema de información cooperante.

3.2. Productos a Obtener:

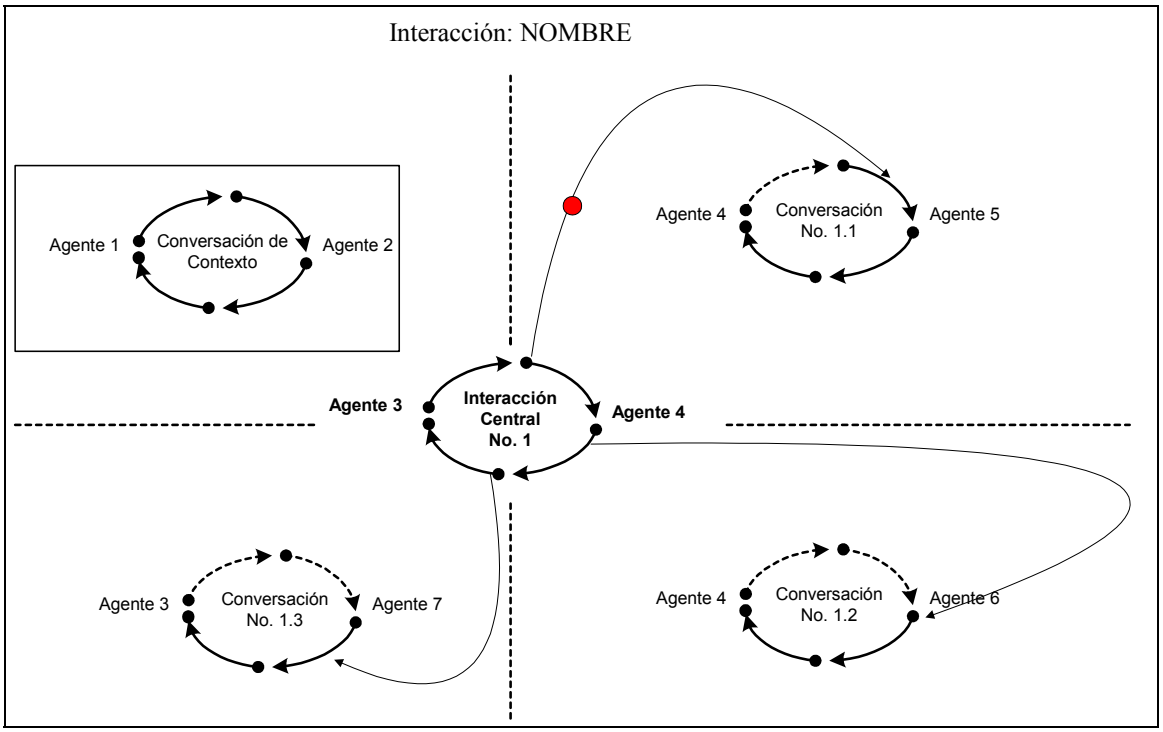
3.2.1. Para cada una de las conversaciones se construye el diagrama de interacción.

3.2.2. Para cada una de las conversaciones que forman parte de las interacciones del sistema, se debe llenar un conjunto de formas de especificación de servicio: Descripción y especificación de la conversación.

3.2.3. Se construye el Diccionario de Datos general del sistema.

3.3. Diagramas a Realizar:

a. Diagrama de Interacción:

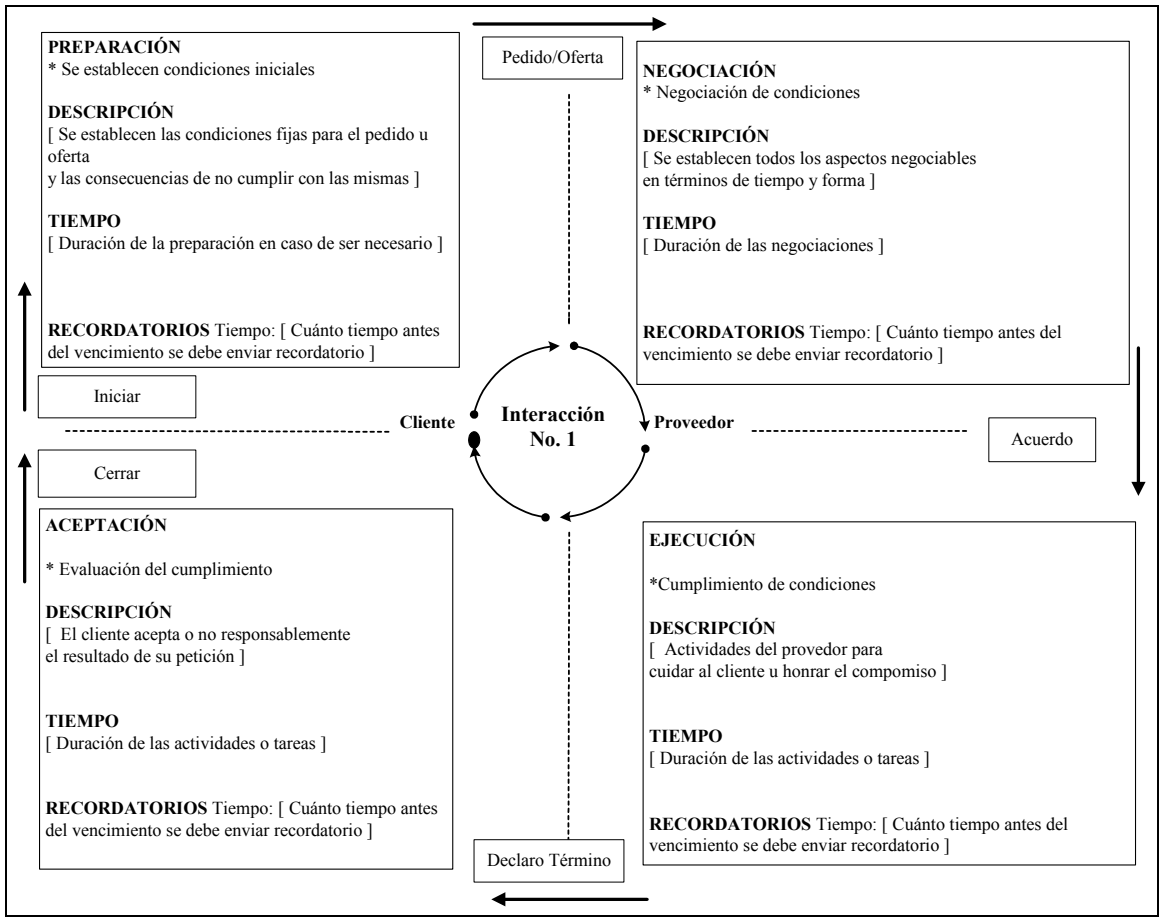


b. Descripción de la Conversación:

DESCRIPCIÓN DE LA CONVERSACIÓN

Nombre de la Interacción:	ID de la Interacción:	Fecha:	Versió n
Nombre de la Conversación:		ID de la Conversación:	
Objetivo Principal de la Conversación:			
Cliente:	Proveedor:		

c. Especificación de la Conversación:



d. Diccionario de Datos:

ESPECIFICACIÓN DEL SERVICIO: DICCIONARIO DE DATOS

* Términos generalmente utilizados por el dueño del servicio

NOMBRE DEL SERVICIO	ID DE SERVICIO	FECHA	VERSIÓN

TERMINO	SIGNIFICADO

ANEXOS

* Sección para documentos adicionales que apoyan a explicar las definiciones.

4. Diseñar los mensajes y puertos de los agentes

4.1. Actividades a Realizar: Diseñar los mensajes y puertos de los diferentes agentes: los Puertos de Entrada (Pin), los Puertos de Salida (Pout).

4.2. Productos a Obtener:

4.2.1. El diagrama de puertos en el que se representan los diversos puertos de comunicación, tanto de entrada como de salida, para los diferentes agentes de acuerdo a las conversaciones en las que participe.

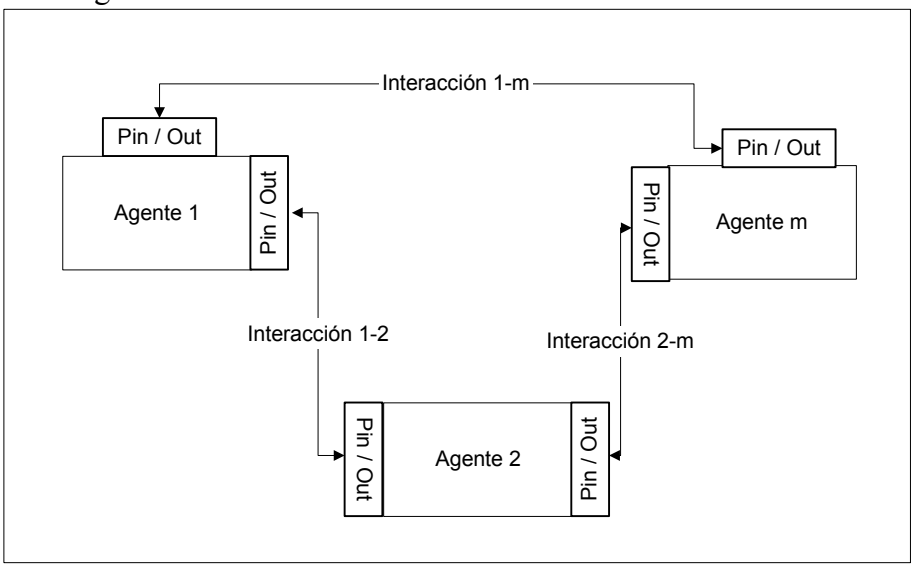
4.2.2. El diseño de los puertos, donde el tipo de datos del puerto esta dado por el número y tipo de mensajes que puede manejar. Los mensajes deben

ser, en nuestro caso, los actos comunicativos de Flores o los de FIPA, pero no una mezcla de los dos.

4.2.3. La construcción de la Especificación Detallada de la Conversación, donde se retoma el modelo generado en el paso anterior, donde tenemos las diferentes especificaciones de las conversaciones, y con base en los actos del habla, representamos las acciones de cada etapa del Loop.

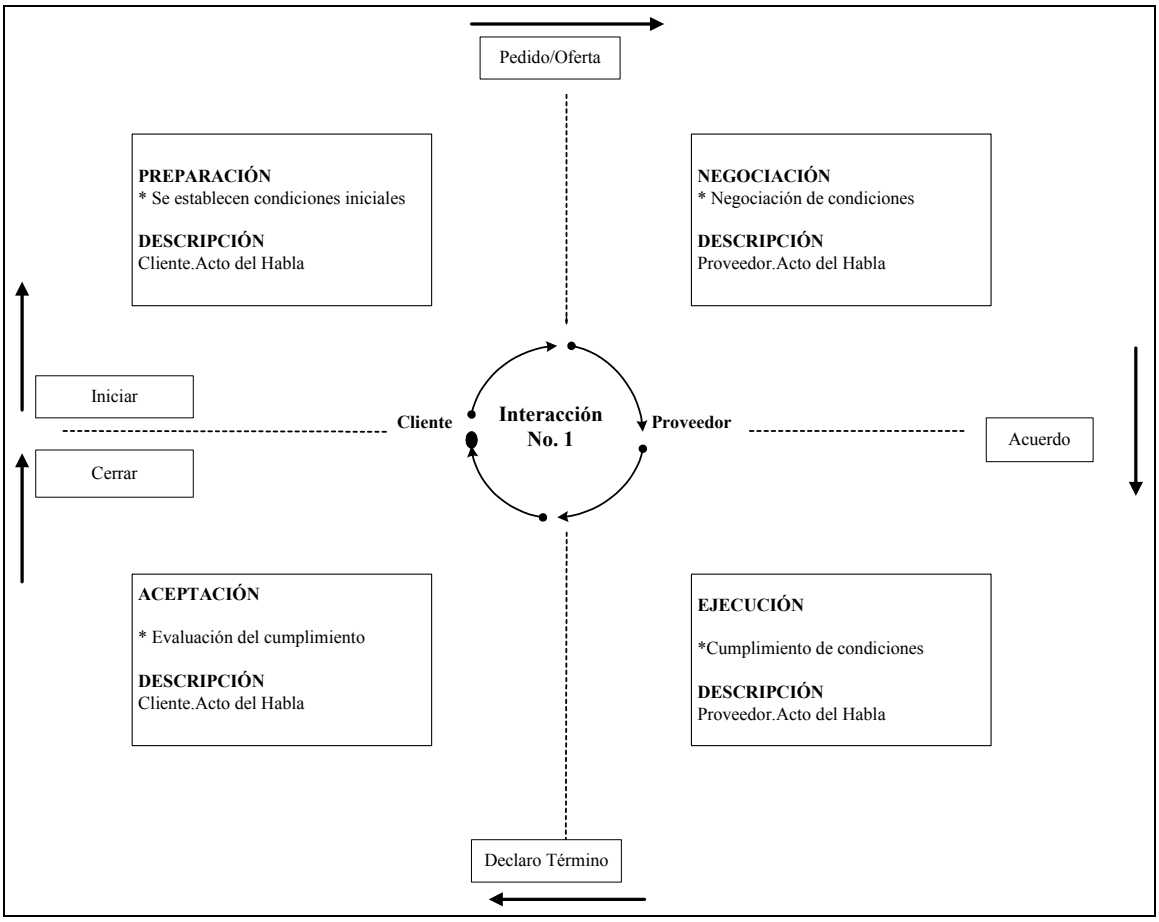
4.3. Diagramas a Realizar:

a. Diagrama de Puertos:

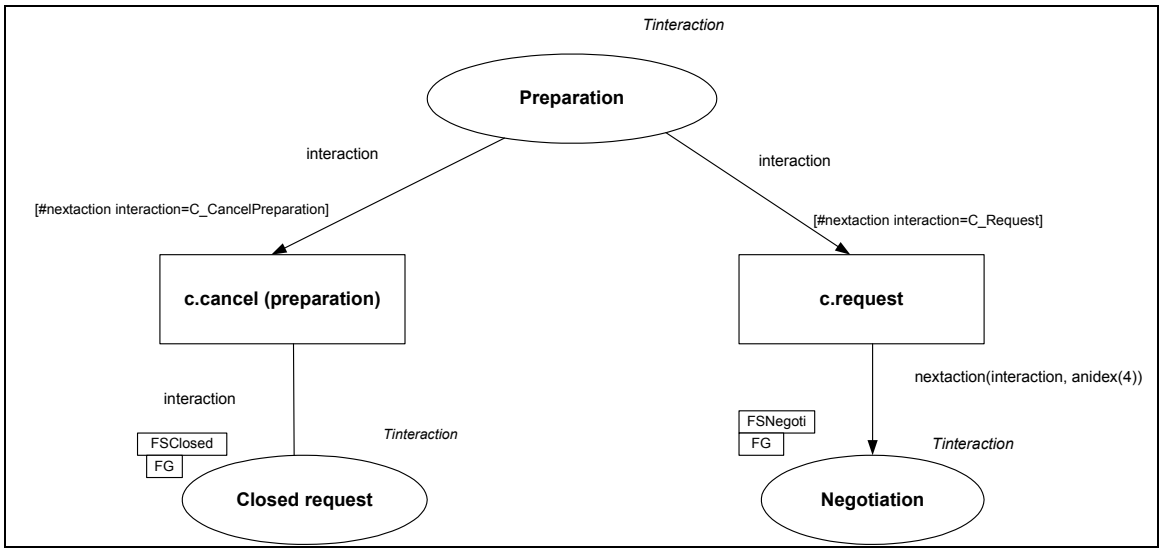


b. Diseño de los puertos: La notación que utilizamos para representar los puertos en una conversación *C*, entre el *Agente 1* y el *Agente 2* es: *Agente* $_1$*.P(in o out)* $_C$*,Agente* $_2$ *(*m_1*,* m_2*, ...,* m_x*)*, donde m_i es el mensaje *i* en el puerto dentro de la conversación *C* entre el *Agente* $_1$ y el *Agente* $_2$, con $i=1..x$, donde *x* es el número total de mensajes en el puerto.

c. Especificación Detallada de la Conversación:



5. **Representar el mecanismo de interacción utilizando Redes de Petri Coloreadas**
   5.1. Actividades a Realizar: La especificación del mecanismo de interacción que le da soporte a los diferentes tipos de mensajes que utilizan los agentes durante sus conversaciones, mediante el uso de Redes de Petri Coloreadas.
   5.2. Productos a Obtener: El conjunto de Redes de Petri Coloreadas que representan el mecanismo de interacción, donde los estados son representados por medio de las plazas y los actos de habla por medio de transiciones. Las relaciones representan transiciones entre plazas, producto de un cambio de estado en la conversación debido a un acto de habla generado por alguno de los agentes.
   5.3. Diagramas a Realizar: Redes de Petri Coloreadas:



C.2. Especificación Implícita

- **Modelo Dinámico:**

6. Simular las interacciones del sistema

6.1. Actividades a Realizar: A partir del modelo estático de la interacción en el Sistema de Información Cooperante en las etapas anteriores, se observar la parte dinámica del mismo, a través de la simulación del mecanismo de interacción en la herramienta Design/CPN.

6.2. Productos a Obtener: En la herramienta Design/CPN, la ejecución de la interacción.

6.3. Diagramas a Realizar: Ninguno.

Anexo D

Traducción de Términos en Inglés

A continuación se presentan parte de la terminología utilizada en Inglés a lo largo del documento y su traducción al español.

Tabla D.1: Actos Comunicativos Básicos de Flores para el Cliente.

Basic Communicative Acts	Actos Comunicativos Básicos
Client	**Cliente**
Request	Petición
Declare satisfaction	Declara Satisfacción
No agr	No Acuerdo
No rep	No Respuesta
Void	Nulo
Cancel	Cancela
No agr	No Acuerdo
Preparation	Preparación
Void	Nulo
Cancel make new request	Cancela y hace una nueva petición
Decline to accept	Declina Aceptar
No agr	No Acuerdo
Void	Nulo
Close	Cierra
Revoked	Revocado
Declined	Declinado
Ask recons	Pide reconsiderar
Revoked	Revocado
Declined	Declinado
Counter	Contraoferta
Decline counteroffer	Declina la contraoferta
Agree to counteroffer	Acepta la contraoferta

Tabla D.2: Actos Comunicativos Básicos de Flores para el Proveedor.

Provider	**Proveedor**
Agree	Acepta
Decline	Declina
Counteroffer	Contraoferta
Revoke and counteroffer	Revoca y Contraoferta
Revoke	Revocado
No agr	No Acuerdo
Void	Nulo
Report completion	Reporta termino
No agr	No Acuerdo
Void	Nulo
Ask recons	Pide reconsiderar
Cancel	Cancelado
Close	Cierra
Canceled	Cancelado
Satisfied	Satisfecho

Anexo E

Declaraciones para las Simulaciones en Design/CPN

A continuación se presentan las declaraciones en CPN-ML, tomadas directamente de la especificación en la herramienta Design/CPN, en el nodo de declaraciones (*Declaration Node)*.

(* Application Declarations, basic colors *)

```
color Tstring = string;
color Tname = Tstring;
color Tdescription = Tstring;
color Tid = int;
```

(* B2B Aplication *)

```
color Tprice = int;
color Tstock = int;
color Tcredit = int;
color Tdiscount = int;
```

(* Contact Center Aplication*)

```
color Tstandard = Tstring;
color Tlocation = int;
color Tphone = int;
color Tdate = Tstring;
color Thour = Tstring;
```

(* Colors declarations for random simulation *)

```
color TOptions = int with 1..8;
```

(* Communicative Acts Declaration *)

```
color TPorts = with  P_Agree |
                     P_Decline |
                     P_Counteroffer |
                     P_Comment |
                     P_ReportCompletion | P_ReportCompletionNoAgr |
                     P_Revoke | P_RevokeNoAgr |
                     P_RevokeAndCounteroffer |
                     P_Close | P_CloseSatisfied | P_CloseCanceled |
                     P_AskRecons | P_AskReconsCancel | (* TPortIn *)
                     C_Request |
                     C_Comment |
                     C_DeclareSatisfaction | C_DeclareSatisfactionNoAgr |
                     C_DeclareSatisfactionNoRep |
                     C_Cancel | C_CancelPreparation | C_CancelNoAgr |
                     C_CancelMakeNewRequest |
                     C_DeclineToAccept | C_DeclineToAcceptNoAgr |
                     C_Close | C_CloseRevoked | C_CloseDeclined |
                     C_AskRecons | C_AskReconsRevoke | C_AskReconsDecline |
                     C_Counter |
                     C_DeclineCounteroffer |
                     C_AgreeToCounteroffer;   (* TPortOut *)
```

(* Application Declarations, complex colors *)

(* B2B Aplication *)

```
color Tproduct = record
                  n:Tname *
                  p:Tprice *
                  s:Tstock *
                 d:Tdescription;
color Tproduct = int;

color Tlistproduct = list Tproduct;

color Tbuyer = record
                id:Tid *
                n:Tname *
                c:Tcredit *
                d:Tdiscount;
color Tbuyer = int;

color Tlistbuyer = list Tbuyer;

color TMarketplace = record
```

```
                id:Tid *
                n:Tname;

color Tsupplier = record
            id:Tid *
            n:Tname *
            p:Tlistproduct;
color Tsupplier = int;

color Tlistsupplier = list Tsupplier;
```

(* Interaction Color Declaration *)

(One Marketplace, Many Buyers, Many Suppliers, One Product *)*

```
color Tinteraction = record
                buyer:Tlistbuyer *
                seller: TMarketplace *
                supplier:Tlistsupplier *
                theproduct: Tproduct *
                nextaction: TPorts *
                actualindex : TOptions;
```

(* Contact Center Aplication *)

```
color Trequest = record
            id:Tid *
            d:Tdate  *
            h:Thour  *
            d:Tdescription;
color Trequest = int;

color Tlistrequest = list Trequest;

color Tuser  = record
           id:Tid *
           n:Tname *
           l:Tlocation *
           p:Tphone;
color Tuser = int;

color Tlistuser = list Tuser;

color TCCenter  = record
            id:Tid *
            n:Tname;
```

```
color TPCenter  = record
                id:Tid *
                n:Tname *
                t:TlistTech *
                e:Tstandard;
color TPCenter = int;

color TlistPCenter = list TPCenter;

color TTech  = record
              id:Tid *
              n:Tname *
              l:Tlocation *
              p:Tphone;
color TTech = int;

color TlistTech = list TTech;
```

(* Interaction Color Declaration *)

(One Contact Center, Many Users, Many Process Center, Many Tech, One Request *)*

```
color Tinteraction = record
                    user:Tlistuser *
                    cc: TCCenter *
                    pc: TlistPCenter *
                    lt: TlistTech *
                    therequest: Trequest *
                    nextaction: TPorts *
                    actualindex : TOptions;
```

(* Interaction Variable *)

```
var interaction : Tinteraction;
var theindex    : TOptions;
```

(* Function to calculate an index to select a communicative act in the simulation *)

```
fun fixtheindex3(thevalue, thelimit : TOptions) =
   let
    val goodindvalue = thevalue <= 6
   in
    case goodindvalue of
     true => thevalue - thelimit |
     false => thevalue - thelimit - thelimit
   end;
```

```
fun fixind2(thevalue, thelimit : TOptions) =
   let
    val goodindvalue = thevalue >= 7
   in
    case goodindvalue of
     true => thevalue - thelimit - thelimit - thelimit |
     false => thevalue - thelimit – thelimit
   end;
```

```
fun fixtheindex2(thevalue, thelimit : TOptions) =
   let
    val goodindvalue = thevalue <= 4
   in
    case goodindvalue of
     true => thevalue - thelimit |
     false => fixind2(thevalue, thelimit)
   end;
```

```
fun fixnewindex(thevalue, thelimit : TOptions) =
   case thelimit of
    2 => fixtheindex2(thevalue, thelimit)|
    3 => fixtheindex3(thevalue, thelimit);
```

```
fun fixtheindex(thevalue, thelimit : TOptions) =
   let
    val goodindvalue = thelimit >= 4
   in
    case goodindvalue of
     true => thevalue - thelimit |
     false => fixnewindex(thevalue, thelimit)
   end;
```

```
fun avalue(thevalue : TOptions) =
   thevalue;
```

```
fun randomindex(theindex:TOptions) =
    ran'TOptions ();
```

```
fun anindex(thelimit : TOptions) =
   let
    val tempind = randomindex(thelimit)
    val arigthindex = tempind <= thelimit
   in
    case arigthindex of
     true  => avalue(tempind) |
     false => fixtheindex(tempind,thelimit)
   end;
```

(* Functions to select the communicative act according to the conversation state *)

(* Provider, Negotiation *)

```
fun pcanegotiation(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => P_Agree |
    2 => P_Decline |
    3 => P_ReportCompletionNoAgr |
    4 => P_Counteroffer ;
```

(* Provider, Performance *)

```
fun pcaperformance(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => P_ReportCompletion |
    2 => P_Revoke |
    3 => P_RevokeAndCounteroffer |
    4 => C_Cancel ;
```

(* Provider, Closure After Cancel No Agreement *)

```
fun pcaclosureaftercancelnoagreement(anaction:TPorts, tindex:TOptions) =
   case tindex of
```

```
    1 => P_CloseCanceled |
    2 => P_AskReconsCancel ;
```

```
(* Provider, Closure After Early Satisfaction No Agreement *)
```

```
fun pcaclosureafterearlysatisfactionnoagreement(anaction:TPorts, tindex:TOptions) =
   P_CloseSatisfied;
```

```
(* Provider, Closure After Cancel *)
```

```
fun pcaclosureaftercancel(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => P_CloseCanceled |
    2 => P_AskReconsCancel ;
```

(* Provider, Closure After Early Satisfaction *)

```
fun pcaclosureafterearlysatisfaction(anaction:TPorts, tindex:TOptions) =
   P_CloseSatisfied;
```

(* Provider, Closed Satisfied *)

```
fun pcaclosedsatisfied(anaction:TPorts, tindex:TOptions) =
   P_CloseSatisfied;
```

(* Provider, Comment *)

```
fun pcacomment() =
   P_Comment;
```

(* Client, Performance *)

```
fun ccaperformance(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => P_ReportCompletion |
    2 => P_Revoke |
    3 => P_RevokeAndCounteroffer |
    4 => C_Cancel |
    5 => C_CancelMakeNewRequest |
```

```
    6 => C_DeclareSatisfactionNoRep ;
```

(* Client, Closure After Decline *)

```
fun ccaclosureafterdecline(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_CloseDeclined |
    2 => C_AskReconsDecline |
    3 => C_CancelMakeNewRequest ;
```

(* Client, Acceptance No Agreement *)

```
fun ccaacceptancenoagreement(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_CancelNoAgr |
    2 => C_DeclineToAcceptNoAgr |
    3 => C_CancelMakeNewRequest ;
```

(* Client, Countered *)

```
fun ccacountered(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_Counter |
    2 => C_DeclineCounteroffer |
    3 => C_CancelMakeNewRequest |
    4 => C_CancelNoAgr |
    5 => C_AgreeToCounteroffer |
    6 => C_DeclareSatisfactionNoAgr ;
```

(* Client, Acceptance *)

```
fun ccaaceptance(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_DeclareSatisfaction |
    2 => C_DeclineToAccept |
    3 => C_Cancel |
    4 => C_CancelMakeNewRequest ;
```

(* Client, Closure After Revoke *)

```
fun ccaclosureafterrevoke(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_CloseRevoked |
    2 => C_AskReconsRevoke |
    3 => C_CancelMakeNewRequest ;
```

(* Client, Closure After Revoke No Agreement *)

```
fun ccaclosureafterrevokenoagreement(anaction:TPorts, tindex:TOptions) =
   case tindex of
    1 => C_CloseRevoked |
    2 => C_AskReconsRevoke |
    3 => C_CancelMakeNewRequest ;
```

(* Client, Comment *)

```
fun ccacomment() =
   C_Comment;
```

(* Interaction Function to select the next communicative act *)

```
fun selectaction(anaction:TPorts, tindex:TOptions) =
   case anaction of
    C_Request => pcanegotiation(anaction, tindex) |
    C_DeclineToAccept => pcaperformance(anaction,tindex) |
    C_Counter => pcanegotiation(anaction, tindex) |
    C_DeclineCounteroffer => pcanegotiation(anaction, tindex) |
    C_DeclineToAcceptNoAgr => pcanegotiation(anaction, tindex) |
    C_CancelMakeNewRequest => pcanegotiation(anaction, tindex) |
    C_CancelNoAgr => pcaclosureaftercancelnoagreement(anaction, tindex) |
    C_AgreeToCounteroffer => pcaperformance(anaction, tindex) |
    C_DeclareSatisfactionNoAgr => pcaclosureafterearlysatisfactionnoagreement(anaction,
tindex) |
    C_Cancel => pcaclosureaftercancel(anaction, tindex) |
    C_DeclareSatisfactionNoRep => pcaclosureafterearlysatisfaction(anaction, tindex) |
    C_DeclareSatisfaction => pcaclosedsatisfied(anaction, tindex) |
    C_AskReconsRevoke => pcanegotiation(anaction, tindex) |
    C_AskReconsDecline => pcanegotiation(anaction, tindex) |
    C_CloseRevoked => pcacomment() |
    C_CloseDeclined => pcacomment() |
    C_Close => pcacomment() |
    C_AskRecons => pcanegotiation(anaction, tindex) |
    P_Agree   => ccaperformance(anaction, tindex) |
```

```
    P_Decline => ccaclosureafterdecline(anaction, tindex) |
    P_ReportCompletionNoAgr => ccaacceptancenoagreement(anaction, tindex) |
    P_Counteroffer => ccacountered(anaction, tindex) |
    P_ReportCompletion => ccaaceptance(anaction, tindex) |
    P_Revoke => ccaclosureafterrevoke(anaction, tindex) |
    P_RevokeAndCounteroffer => ccacountered(anaction, tindex) |
    P_RevokeNoAgr => ccaclosureafterrevokenoagreement(anaction, tindex) |
    P_CloseCanceled => ccacomment() |
    P_CloseSatisfied => ccacomment() |
    P_Close => ccacomment() |
    P_AskRecons => pcanegotiation(anaction, tindex) |
    P_AskReconsCancel => ccacountered(anaction, tindex) ;
```

(* Function to Assign the next interaction values *)

(* B2B Aplication *)

```
 fun nextaction(intein: Tinteraction, tindex: TOptions) =
     {buyer = #buyer intein,
     seller =#seller intein,
     supplier = #supplier intein,
     theproduct =#theproduct intein,
     nextaction =selectaction(#nextaction intein, tindex),
     actualindex =tindex};
```

(* Contact Center Aplication *)

```
fun nextaction(intein: Tinteraction, tindex: TOptions) =
   {user = #user intein,
     cc = #cc intein,
     pc = #pc intein,
     lt = #lt intein,
     therequest =#therequest intein,
     nextaction =selectaction(#nextaction intein, tindex),
     actualindex =tindex};
```
