PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

COMPARACIÓN DE METODOLOGÍAS ORIENTADAS A AGENTES

NATALIA ANDREA CALBUL TAPIA

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

DICIEMBRE, 2007

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

COMPARACIÓN DE METODOLOGÍAS ORIENTADAS A AGENTES

NATALIA ANDREA CALBUL TAPIA

Profesor Guía: **Claudio Cubillos Figueroa**

Profesor Co-referente: **Alexander Cristian Rusu**

Carrera: **Ingeniería Civil en Informática**

Diciembre, 2007

Dedicatoria

A Dios, por brindarme fuerza.
A mis Padres por su inagotable esfuerzo, apoyo y comprensión.
A mi Profesor Guía, por los conocimientos entregados.

Resumen

En los últimos años se han propuesto diversas metodologías para el desarrollo de sistemas multiagente. A pesar de esto, ninguna de ellas ha alcanzado el nivel de madurez necesario para ser utilizada en la industria del software. Uno de los pasos para suplir esta deficiencia es entender las fortalezas y debilidades de las principales metodologías orientadas a agentes. En este contexto se propone la comparación de las metodologías *MASE* y *PASSI* a través de la utilización de los frameworks de evaluación de *Pedro Cuesta* y *Arnon Sturm*. La comparación se complementa con la aplicación de ambas metodologías al caso de estudio de un *Sistema Multiagente de Transporte de Pasajeros en Respuesta a la Demanda (DRTS)*, la cual se lleva a cabo en la etapa inicial. Posteriormente, se evalúa cada metodología utilizando los frameworks de evaluación especificados. Los resultados obtenidos establecen que la metodología *PASSI* posee un proceso de desarrollo más completo que *MASE*, razón por la cual su aplicación se ajusta de mejor manera al caso de estudio planteado.

Abstract

In the last years several methodologies have been proposed for multiagent systems development. Despite this, none one has reached the maturity level required to be used in the software industry. One of the steps towards fulfilling this demand is to understand the strengths and weaknesses of the main agent-oriented methodologies. In this context the comparison of *MASE* and *PASSI* methodologies through the use of *Pedro Cuesta's* and *Arnon Sturm's* evaluation frameworks is proposed. The comparison is complemented with the application of both methodologies to the case study of a *Demand Responsive Transport Multiagent System (DRTS)*, which takes place at the initial stage. After this, each methodology is evaluated using the specified evaluation frameworks. The results obtained establish that *PASSI* methodology has a more complete development process than *MASE*, which is why its application fits better to the case study.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

Presentación del tema

1.1 Introducción

Desde hace tiempo es sabido que el software tiene ciertas características esenciales que hacen difícil obtener un desarrollarlo eficiente, robusto y correcto. De manera similar, se ha reconocido que ciertos tipos de sistemas informáticos son más difíciles de construir que otros. Dicha complejidad no está referida a la complejidad algorítmica, sino a aquella presente en la especificación, diseño y construcción del software desde la perspectiva de la Ingeniería de Software [1].

Actualmente se reconoce que la interacción es probablemente la característica más importante de la complejidad del software. Como consecuencia, el tópico de mayor investigación en ciencias de la computación durante las últimas dos décadas ha sido el desarrollo de herramientas y técnicas para modelar, entender e implementar sistemas en donde la interacción es la norma.

Las más recientes adiciones son las nociones de *Agente Inteligente* y *Sistemas Multiagente (MAS - MultiAgent System)*. Actualmente han crecido en lo que es ahora una de las áreas más activas de investigación y desarrollo en computación. Existen muchas razones para explicar el actual interés, pero una de las más importantes es que el concepto de *agente* como un sistema autónomo, capaz de interactuar con otros agentes en orden de satisfacer sus objetivos de diseño, es una forma natural para los diseñadores de software [2].

Esta evolución ha implicado la creación de diversas metodologías para la construcción de sistemas multiagente [3], existiendo más de dos docenas de ellas. A modo de ejemplo podemos citar a: *MAS-CommonKADS, MASSIVE, TROPOS, MESSAGE, INGENIAS, GAIA, MASE* y *PASSI*. Sin embargo, ninguna de ellas ha alcanzado la completitud y el nivel de madurez suficiente como para ser utilizada en la industria del software; principalmente porque no poseen un proceso de desarrollo que abarque todas las etapas de un proyecto. La mayoría de las propuestas intentan cubrir fundamentalmente las etapas de análisis y diseño, y no consideran el resto de las etapas o bien dejan total libertad para su desarrollo. Como consecuencia de esto, debemos enfrentar los problemas de estandarización que se producen dadas las diferencias entre las etapas cubiertas por un proceso de desarrollo y otro, y las diferencias entre las notaciones gráficas que utilizan.

Con la intención de solucionar estos problemas, se han realizado comparaciones entre las diversas metodologías para la construcción de sistemas multiagente, con el propósito de entender la relación entre dichas metodologías e identificar sus concordancias y diferencias.

Sin embargo, estos presentan carencias en la manera de realizar las evaluaciones. Por ejemplo, en [4] el autor propone una lista de preguntas para evaluar una metodología, pero no se muestra como se deben aplicar. Otro estudio [5], sugiere la utilización de un framework de evaluación basado en criterios que sólo se refieren a la expresividad de las metodologías. En [6] el autor realiza una evaluación de cinco metodologías orientadas a agentes, sin embargo, se evalúan sólo a algunos conceptos, como por ejemplo diseño de la organización y cooperación, y no al amplio conjunto de atributos que constituyen una completa metodología. En [7], los autores realizan una evaluación sólo del modelado de una metodología. Otros estudios que tratan con la evaluación de metodologías orientadas a agentes comparan dos o tres metodologías, principalmente en lo que respecta a la expresividad y a los conceptos apoyados por la metodología. A esto se debe añadir la falta de utilización de un caso de estudio común al cual se le apliquen las metodologías a comparar, en la mayoría de las publicaciones.

En este contexto, se hace necesaria la realización de comparaciones que evidencien de manera clara y precisa las principales fortalezas, debilidades, concordancias y diferencias de las metodologías, a través de la utilización de un caso de estudio común y de la utilización de frameworks de evaluación que consideren un amplio conjunto de criterios.

1.2 Objetivo general

Comparar las metodologías orientadas a agente *MASE* y *PASSI* mediante la utilización de frameworks de evaluación.

1.3 Objetivos específicos

- Comprensión detallada de las metodologías orientadas a agente *MASE* y *PASSI*, y de los frameworks de evaluación existentes para este tipo de metodologías.

- Aplicación de ambas metodologías mediante la utilización del caso de estudio de un *Sistema Multiagente de Transporte de Pasajeros en Respuesta a la Demanda* (*DRTS*).

- Comparación de ambas metodologías mediante los frameworks de evaluación de *Pedro Cuesta* y *Arnon Sturm.*

1.4 Metodología de trabajo

La metodología de trabajo propuesta para llevar a cabo este proyecto contempla las siguientes etapas:

- Se inicia la investigación con la concepción de la idea central, con la cual se plantea el problema a través de los objetivos, cuestionamientos asociados y justificaciones referentes su viabilidad.

- Se desarrolla un marco teórico, lo cual contempla la revisión de la literatura desde un punto de vista de la obtención, selección y la extracción de información relevante. Esto se asocia directamente con todas las referencias utilizadas para la investigación y los contenidos teóricos que se verán a través de los informes desarrollados en el proyecto.

- Se realizan las decisiones de diseño necesarias para comenzar la evaluación. Esto es, se definen las herramientas de experimentación, en donde se contempla tanto el instrumento utilizado como las muestras para realizar los experimentos.

- Se realiza un análisis profundo de los resultados obtenidos, a partir del cual se obtendrán las conclusiones del proyecto.

- Se elaboran y presentan los reportes de investigación adecuados.

1.5 Plan de trabajo

Las actividades necesarias para cumplir con los objetivos establecidos en las secciones 1.2 y 1.3 son presentadas en las Figuras 1.1 y 1.2.

	Nombre de tarea	Duración	Comienzo	Fin
1	Estudio y evaluación de Metodologías AOSE	13 días	mié 07-03-07	lun 26-03-07
2	Estudio y evaluación de las Técnicas de Evaluación de Metodologías	16 días	mar 27-03-07	mar 17-04-07
3	Estudio y evaluación de Frameworks de Evaluación para AOSEM	16 días	mié 18-04-07	mié 09-05-07
4	Estudio y evaluación de Casos de Estudio	16 días	jue 10-05-07	jue 31-05-07
5	Elección de Metodologías AOSE a evaluar	2 días	mar 27-03-07	mié 28-03-07
6	Elección de Técnicas para la Evaluación de Metodologías	2 días	mar 17-04-07	mié 18-04-07
7	Elección de Frameworks de evaluación	2 días	mié 18-04-07	vie 20-04-07
8	Elección de un Caso de Estudio	2 días	vie 01-06-07	lun 04-06-07
9	Modelado del Caso de Estudio con la Metodología PASSI	30 días	lun 06-08-07	vie 14-09-07
10	Aplicación del Framework Pedro Cuesta a la Metodologías PASSI	30 días	lun 06-08-07	vie 14-09-07
11	Aplicación del Framework de Arnon Sturn a la Metodologías PASSI	30 días	lun 06-08-07	vie 14-09-07
12	Modelado del Caso de Estudio con la Metodología MASE	30 días	lun 17-09-07	lun 29-10-07
13	Aplicación del Framework Pedro Cuesta a la Metodologías MASE	30 días	lun 17-09-07	vie 26-10-07
14	Aplicación del Framework de Arnon Sturn a la Metodologías MASE	30 días	lun 17-09-07	vie 26-10-07
15	Comparación de Resultados Obtenidos	15 días	lun 29-10-07	vie 16-11-07

Figura 1.1 Planificación detallada del proyecto.

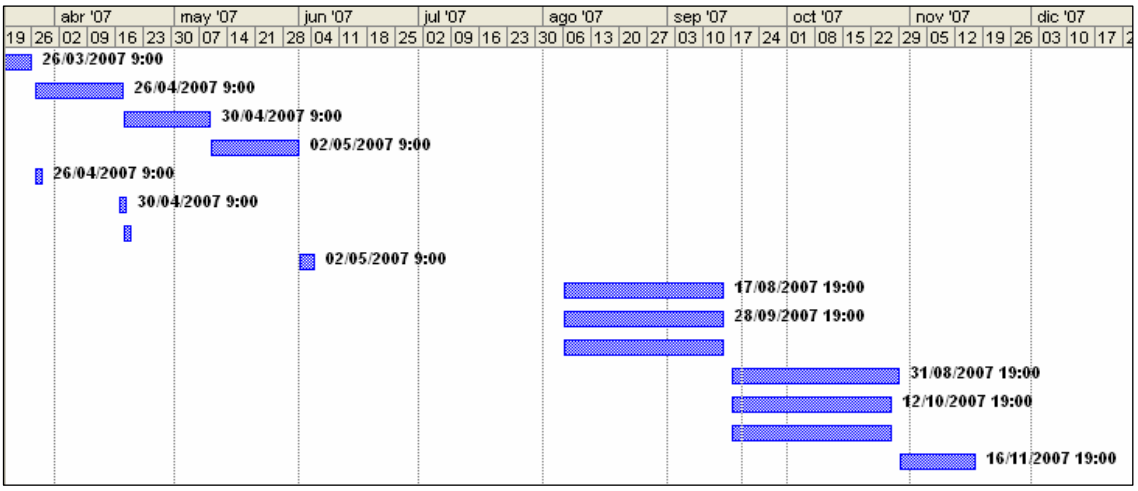

Figura 1.2 Calendarización planificación del proyecto.

1.6 Organización del texto

Este documento está estructurado de la siguiente manera:

- En el Capítulo 2, *Paradigma de agentes*, se definen los conceptos de agente y sistemas multiagente, se presentan los tipos de metodologías orientadas a agente y se describen aquellas que son más dominantes en el área.

- En el Capítulo 3, *Evaluación de metodologías AOSE*, se presentan las técnicas de evaluación y los frameworks de evaluación existentes.

- En el Capítulo 4, *Alcance de la comparación*, se definen las técnicas de evaluación a utilizar, se fundamenta la elección de las metodologías a comparar y se presenta el caso de estudio.

- En el Capítulo 5, *Desarrollo de la comparación*, se presenta el modelado del caso de estudio bajo las metodologías orientadas a agente *PASSI* y *MASE*, y la aplicación de los frameworks de evaluación a dichas metodologías.

- En el Capítulo 6, *Comparación de metodologías*, se presentan los resultados obtenidos tras la evaluación y su respectivo análisis comparativo.

- En el Capítulo 7, *Conclusiones*, se presentan las conclusiones obtenidas tras la realización de la comparación.

Capítulo 2

Paradigma de agentes

2.1 Agente inteligente y sus características

Se define como agente inteligente a un sistema informático, situado en algún entorno, dentro del cual actúa de forma autónoma y flexible para cumplir sus objetivos de diseño. Además de la interacción con el medio, un agente se caracteriza por las siguientes propiedades [8]:

- *Autonomía*, un agente tiene la capacidad de actuar sin intervención humana directa o de otros agentes.

- *Habilidad social*, un agente tiene la capacidad de interactuar con otros agentes, utilizando como medio algún lenguaje de comunicación entre agentes.

- *Reactividad*, un agente está inmerso en un determinado entorno (hábitat), del que percibe estímulos y ante los que debe reaccionar en un tiempo preestablecido.

- *Proactividad*, un agente no sólo debe reaccionar a los cambios que se produzcan en su entorno, sino que ha de tener un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe satisfacer.

Cuando se define un agente con las características anteriores, se dice que éste posee una *noción débil de agente*. Cuando se define un agente con las características anteriores y más además como una entidad cuyo estado es visto como un conjunto de componentes mentales, tales como creencias, capacidades, elecciones y acuerdos, se dice que éste posee una *noción fuerte de agente*. Las características añadidas son las siguientes:

- *Movilidad*, es la capacidad del agente para moverse a través de una red electrónica.

- *Veracidad*, seguridad de que el agente no transmitirá información falsa conscientemente.

- *Benevolencia*, se asume que el agente hará aquello para lo que se creó.

- *Racionalidad*, se asume que el agente intentará alcanzar las metas que le fueron asignadas de la mejor forma posible, basándose en la información que posee del mundo exterior.

La *noción fuerte de agente* está más cercana al mundo de la *Ingeniería del Conocimiento (KE- Knowledge Engineering)*, y es por tanto la más aceptada dentro de dicho campo.

2.2 Sistemas multiagente

Un sistema multiagente es aquel que se diseña e implementa pensando en que estará compuesto por varios agentes que interactuarán entre sí, de forma que juntos permitan alcanzar la funcionalidad deseada [9]. Es importante distinguir la diferencia entre un sistema basado en agentes y un sistema multiagente [10]; ya que el primero es aquel que utiliza el concepto de agente como mecanismo de abstracción, pero aunque sea modelado en términos de agentes podría ser implementado sin ninguna estructura de software correspondiente a éstos.

Los sistemas multiagente son adecuados para solucionar problemas para los que hay múltiples métodos de resolución y/o múltiples entidades capaces de trabajar conjuntamente para solucionarlos. Por ello, uno de los aspectos básicos en estos sistemas es la interacción entre los diferentes agentes que los forman, esto es, la definición de modelos concretos de cooperación, coordinación o negociación entre agentes.

Una de las razones de por qué los sistemas multiagente han despertado tanto interés, es que estos son una metáfora natural para el modelado de gran cantidad de dominios, donde aparecen una serie de componentes que interaccionan. Además, los sistemas multiagente permiten la distribución de forma natural tanto de los datos como del control de ejecución, la integración sencilla de sistemas que aún siendo obsoletos deben seguir funcionando por razones de necesidad (*legacy systems*), y la creación de sistemas abiertos donde se conozca en tiempo de ejecución los componentes exactos que tendrá el sistema.

2.3 Tipos de metodologías orientadas a agente

En la actualidad existe una gran cantidad de metodologías orientadas a agente, las que poseen diversos orígenes. Algunas están basadas en conceptos de la *Inteligencia Artificial* (*AI - Artificial Intelligence*), otras son extensiones directas de las *Metodologías Orientadas a Objeto* (*OO - Object Oriented*) existentes, mientras otras intentan combinar estos dos enfoques tomando una aproximación más purista, permitiendo ideas *OO* sólo cuando éstas muestran ser suficientes para abordar el problema.

La **¡Error! No se encuentra el origen de la referencia.** muestra las influencias y genealogía de algunas metodologías orientadas a agente [11].

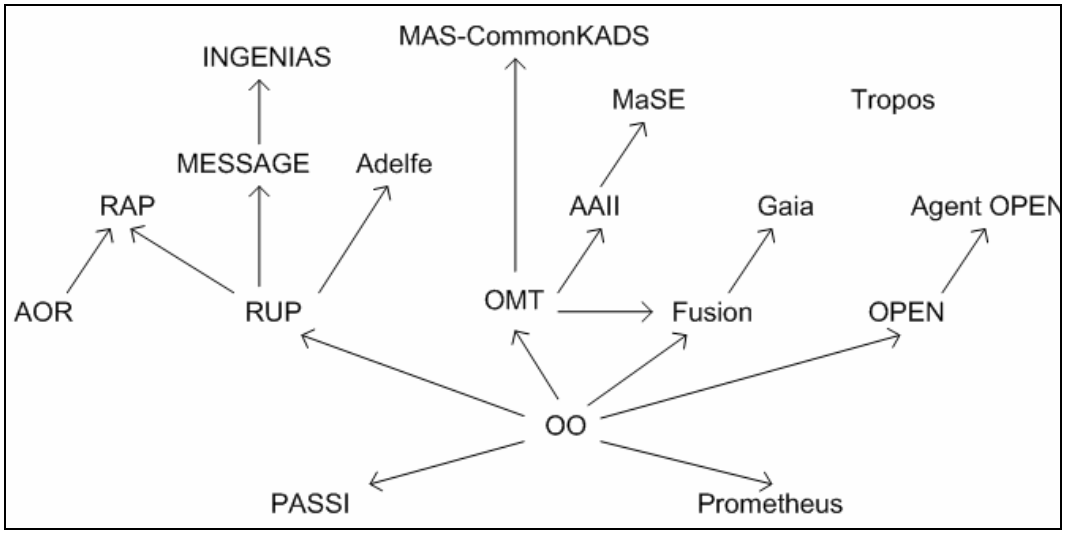

Figura 2.1 Influencias directas e indirectas de metodologías OO sobre metodologías AO [11].

La principal ventaja de la utilización de metodologías basadas en *OO* deriva de las similitudes que presentan agentes y objetos, ya que los agentes pueden ser considerados como objetos activos con un determinado estado mental y ambos paradigmas utilizan paso de mensajes para comunicarse, pudiendo utilizar la herencia y la agregación para definir su arquitectura. La principal diferencia radica en las restricciones que se aplica a los tipos de mensajes en el paradigma orientado a agentes, así como en la definición del estado del agente, basado en sus creencias, deseos e intenciones.

Por otro lado, la popularidad de las metodologías *OO* es otra de las ventajas, ya que actualmente se están utilizando distintas metodologías orientadas a objetos con éxito en la industria, como por ejemplo: *Ingeniería de Software Orientada a Objetos* (*OOSE - Object Oriented Software Engineering*) o el *Proceso Unificado de Desarrollo* [12], lo cual puede ser fundamental para facilitar la integración de la tecnología de agentes, pues los ingenieros de software no tendrían que aprender una nueva metodología desde cero, y además en una empresa siempre se prefiere utilizar una metodología que ya ha sido probada con éxito en el campo industrial.

2.4 Metodologías orientadas a agente

En esta sección se presentan las metodologías más dominantes en el campo de la *Ingeniería de Software Orientada a Agentes*, a saber*: MAS-CommonKADS*, *INGENIAS*, *TROPOS*, *GAIA*, *MASE* y *PASSI.*

2.4.1 MAS-CommonKADS

MAS-CommonKADS es una metodología que extiende de *CommonKADS* [13] y que toma ideas de las metodologías orientadas a objetos para su aplicación a la producción de *MAS* [14] [15]. La metodología *CommonKADS* gira alrededor del modelo de experiencia y está pensada para desarrollar sistemas expertos que interactúen con el usuario. De hecho, considera sólo dos agentes básicos: el usuario y el sistema. *MAS-CommonKADS* extiende los modelos de *CommonKADS* para tener en cuenta la posibilidad de que dos o más componentes del sistema interactúen.

Esta metodología ha sido la primera en plantear un desarrollo de *MAS* integrado con un ciclo de vida de software, concretamente el espiral dirigido por riesgos [16]. Propone siete modelos para la definición del sistema: *agente, tareas, experiencia, coordinación, comunicación, organización* y *diseño*. Cada modelo presenta referencias a la teoría sobre la que se basa. El modelo en sí parte de una descripción gráfica que luego se complementa con explicaciones en lenguaje natural de cada elemento. Existe por cada modelo una descripción de las dependencias respecto de otros modelos y de las actividades involucradas. Estos modelos se hayan descritos ampliamente en [14] en lenguaje natural, complementándose con otras notaciones como *SDL* (*Specification and Description Language*) o *MSC* (*Message Sequence Chart*) para describir el comportamiento de los agentes cuando interaccionan.

MAS-CommonKADS posee a *MAST* (*MultiAgent Systems Tool*) como herramienta de soporte. Sin embargo, no apoya el análisis y diseño, sino que es un conjunto de arquitecturas y frameworks de agentes.

2.4.2 Ingenias

Nace como evolución de las ideas de *MESSAGE* [17]. *INGENIAS* profundiza en los elementos mostrados en el método de especificación, en el proceso de desarrollo, y además incorpora nuevas herramientas de soporte y ejemplos de desarrollo.

INGENIAS, como *MESSAGE*, define un conjunto de meta-modelos (una descripción de alto nivel de los elementos que componen un modelo) con los que hay que describir el sistema. Los meta-modelos indican qué hace falta para describir: agentes aislados, organizaciones de agentes, el entorno, interacciones entre agentes o roles, tareas y objetivos. Estos meta-modelos se construyen mediante un lenguaje de meta-modelado, el *GOPRR* (*Graph, Object, Property, Relationship, and Role*) [18]. En la construcción de estos meta-modelos se integran resultados de investigación en forma de entidades y relaciones entre entidades.

La instanciación de estos meta-modelos produce diagramas similares a los que se usa en *UML*, con la diferencia de que estos diagramas se han creado exclusivamente para definir el sistema multiagente. El proceso de instanciación de los meta-modelos no es trivial, existen muchas entidades y relaciones a identificar, además de dependencias entre distintos modelos. Por ello, *INGENIAS* define un conjunto de actividades cuya ejecución

termina en un conjunto de modelos [19]. Estas actividades a su vez se organizan siguiendo un paradigma de ingeniería del software, a saber: *RUP (Rational Unified Process)*. La ejecución de actividades para producir modelos se basa en la herramienta *INGENIAS IDE*, una herramienta para modelado visual, la cual almacena la especificación del sistema utilizando *XML*. Esta especificación se procesa para generar código (a partir de plantillas de código con información de la especificación). De igual manera, para generar la documentación se parte desde plantillas de documentación que se completan utilizando información de los modelos.

2.4.3 Tropos

TROPOS es una metodología de desarrollo de software basado en agentes que sigue tres ideas principales [20]:

- La noción de agente y todo lo relacionado con sus nociones mentales (objetivos, planes) es tomado en cuenta en todas las fases del proceso de desarrollo del sistema.

- Aborda de forma muy temprana una fase de análisis de requerimientos, lo que permite una mayor comprensión del entorno donde debe operar el sistema resultante, incluyendo los tipos de interacción que deben ocurrir entre los usuarios y el sistema.

- Adopta un proceso de refinado de artefactos. Incluye dentro de su propuesta fases de análisis, diseño y de implementación. Éste último la diferencia de la mayor parte de las metodologías existentes. Además, se apoya en la utilización de *UML* y de extensiones de éste, como por ejemplo *AUML*, así como varios protocolos de comunicación de *FIPA*.

Las fases generales están divididas en cinco: *análisis temprano de requerimientos, análisis tardío de requerimientos, diseño de la arquitectura, diseño de desarrollo o detallado e implementación.*

1. **Análisis Temprano de Requerimientos:** El principal objetivo de esta fase es entender el problema de la organización. Se identifican las dependencias entre los usuarios, distinguiendo cuando sea necesario, objetivos fuertes y débiles en dependencia del nivel de importancia. Los objetivos pueden ser divididos en sub-objetivos, dependiendo del tipo de análisis que se haga.

2. **Análisis Tardío de Requerimientos:** A esta fase llegan todos los objetivos y sub-objetivos que son identificados en la fase anterior y se comienza a especificar más concretamente que es lo que tiene que hacer el sistema dentro del ambiente operacional, centrándose en las funcionalidades relevantes. El resultado final de esta etapa es la identificación de todos los requerimientos funcionales y no funcionales del sistema. De forma general los requerimientos funcionales salen de los objetivos y los no funcionales de los sub-objetivos.

3. **Diseño de la Arquitectura:** El objetivo final de esta etapa es la definición de la arquitectura del sistema de objetivos en términos de sub-sistemas (actores), interconectándolos a través de controles de flujos (dependencias). Se proponen tres pasos fundamentales:

   - Refinar el diagrama de actores del sistema, introduciendo sub-actores sobre el análisis de los requerimientos funcionales y teniendo en cuenta el diseño de patrones definidos.
   - Capturar la capacidad del actor, desde el análisis de las tareas que debe realizar cada actor o sub-actor para poder cumplir los requerimientos finales.

   - Definir un conjunto de tipos de agentes (componentes) y asignar a cada componente uno o varias capacidades diferentes.

4. **Diseño de Desarrollo o Detallado:** Esta fase está dirigida a la especificación de las capacidades de interrelación de los agentes. Generalmente en esta etapa ya se tiene seleccionada la plataforma de desarrollo, por lo que debe ser tomada en cuenta para mejorar dicho diseño detallado. A partir de esta fase se puede realizar una traza hacia atrás hasta el *Análisis Temprano de Requerimientos*. Todas las propiedades que son tratadas en esta fase se modelan usando artefactos *AUML*.

5. **Implementación:** Esta actividad sigue paso por paso lo definido en el *Diseño Detallado*, estableciendo un mapeo entre este resultado y la plataforma de construcción seleccionada.

Estas fases denotan una concepción lógica del proceso de desarrollo de software muy similar a la concepción adoptada en *RUP*.

Adicionalmente, *TROPOS* utiliza un entorno de modelado denominado *i**, que es utilizado sobre todo en el *Análisis Temprano de Requerimientos*. Este entorno aporta actores, objetivos y dependencias entre los actores, en forma de conceptos primitivos y además utiliza los propios diagramas de cada uno de estos elementos que se proponen en la infraestructura de *i** (por ejemplo diagramas de actores y dependencias, y de objetivos). Otro factor importante de justificación del uso de *i**, es que en el *Análisis Temprano de Requerimientos* no sólo es útil capturar el "qué" y el "cómo", sino también identificar el "porqué" de las diferentes piezas que conforman el sistema.

2.4.4 GAIA

GAIA es una metodología para el diseño de sistemas basados en agentes, cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global (no se llega a detallar cuál) [21]. *GAIA* pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño, que según los autores, esté lo suficientemente detallado como para ser implementado directamente.

En *GAIA* se entiende que el objetivo del análisis es conseguir comprender el sistema y su estructura sin referenciar ningún aspecto de implementación. Esto se consigue a través de la idea de organización. Una organización en *GAIA* es una colección de roles, los cuales mantienen ciertas relaciones con otros y toman parte en patrones institucionalizados de interacción con otros roles. Los roles agrupan cuatro aspectos: responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas e interacciones. *GAIA* propone trabajar inicialmente con un análisis a alto nivel. En este análisis se usan los siguientes modelos: *Modelo de Roles* y *Modelo de Interacción*.

Tras esta etapa, se entra en lo que *GAIA* considera diseño a alto nivel. El objetivo de este diseño es generar tres modelos: el *Modelo de Agentes* que define los tipos de agente que existen, cuántas instancias de cada tipo y qué roles realiza cada agente, el *Modelo de Servicios* que identifica los servicios (funciones del agente) asociados a cada rol, y el *Modelo de Comunicación*, que define los enlaces de las comunicaciones que existen entre los agentes.

Los autores de *GAIA* proponen aplicar técnicas clásicas de diseño orientado a objetos para las etapas posteriores. Sin embargo, *GAIA* declara que queda fuera de su ámbito. Esta metodología sólo busca especificar cómo una sociedad de agentes colabora para alcanzar los objetivos del sistema, y qué se requiere de cada uno para lograr esto último.

A continuación, se detallan los modelos previamente mencionados.

1. **Modelo de Roles:** Identifica los roles claves del sistema. Un rol puede ser visto como una descripción abstracta de una funcionalidad esperada de una entidad. Los roles son caracterizados por dos tipos de atributos:

   - **Los permisos:** Un rol tendrá asociado un cierto permiso, de acuerdo al tipo y a la cantidad de recursos que pueden ser explotados al realizar el rol. Este aspecto es representado en el atributo *Permisos del Rol*. Los permisos pueden relacionarse generalmente con cualquier clase de recurso (por ejemplo, un rol se puede dar en un presupuesto monetario, una cierta cantidad de esfuerzo de una persona, etc.). Sin embargo, en *GAIA* los recursos se relaciona sólo con la información o el conocimiento que el agente tiene. Es decir, para realizar un rol, un agente podrá tener acceso a cierta información. Algunos roles pueden generar información, otros pueden necesitar tener acceso a ésta y otros pueden necesitar modificarla. *GAIA* hace uso de una notación formal para expresar los permisos.

- **Las Responsabilidades:** Un rol se crea para hacer algo. Es decir, un rol tiene una cierta funcionalidad, la cual es representada por un atributo conocido como *Responsabilidades de los Roles*. La funcionalidad de un rol está definida por sus responsabilidades. Estas responsabilidades se pueden dividir en dos categorías: *Vida* y *Seguridad*. Las *Responsabilidades de Vida (Liveness)* son intuitivas e indican que algo sucederá, por lo que el agente que ejecuta el rol continúa vivo. *En GAIA* dichas responsabilidades son especificadas por *Expresiones de Vida*, las cuales definen el ciclo de vida del rol.
  Las *Responsabilidades de Seguridad* (*Safety*) son las invariantes que requieren los agentes para realizar un rol. En *GAIA* se especifican por medio de una lista de predicados. Estos predicados se expresan típicamente sobre las variables enumeradas en los atributos de *Permisos de un Rol*.

2. **Modelo de Interacción:** Define las relaciones y dependencias entre los roles en un sistema multiagente, mediante un protocolo de intercambio de mensajes, como el *FIPA-Request*. Se define un conjunto de protocolos, uno para cada tipo de interacción entre roles. Un protocolo se puede ver como patrón institucionalizado de la interacción. Es decir, un patrón de la interacción que ha sido formalmente definido y abstraído de cualquier secuencia particular de ejecución de pasos. La atención está centrada en la naturaleza y el propósito esencial de la interacción.

   La definición de protocolos consiste en la definición de los siguientes atributos:

   - **Propósito:** Breve descripción textual de la naturaleza de la interacción (por ejemplo, la petición de la información, calendario de actividades y asignación de tareas).

   - **Iniciador:** Los roles responsables de comenzar la interacción.

   - **Respondedor:** Los roles con los cuales el iniciador interactúa.

   - **Entradas:** La información usada por el rol iniciador mientras se establece el protocolo.

   - **Salidas:** Información proporcionada por el protocolo durante el curso de la interacción.

   - **Proceso:** Breve descripción textual de la iniciación del proceso realizada durante el transcurso de la interacción.

3. **Modelo de Agente:** El propósito de este modelo es documentar los distintos tipos de agentes que serán utilizados en el sistema y las instancias de agentes que se utilizarán en la ejecución del sistema. De hecho, hay muchas correspondencias entre los roles y los tipos de agentes (como las consideradas en el *Modelo de Roles*). Sin embargo, un diseñador puede elegir empaquetar un número de roles

relacionados en el mismo tipo del agente para los propósitos que estime conveniente.
La eficacia también es un punto importante en esta etapa. Un diseñador puede optimizar el diseño agregando un número de roles de un tipo de agente.
El *Modelo de Agente* se define usando un tipo simple árbol de agentes, en el cual los nodos hoja corresponden a los roles (según lo definido en el *Modelo de Roles*), y los otros nodos corresponden a los tipos del agente.

4. **Modelo de Servicios:** Este modelo permite identificar los servicios asociados con cada rol de agente y especificar las propiedades principales de cada servicio. Un servicio es una función del agente. En términos de *OO*, un servicio correspondería a un método, sin embargo, en *GAIA* los servicios no están disponibles para otro agentes de la misma forma que los métodos de un objeto están disponibles para otros objeto. Más bien, un servicio es un bloque particular y coherente de la actividad que desempeña el agente. Debe estar claro que cada actividad de la etapa de análisis corresponde a un servicio, aunque no cada servicio corresponde a una actividad.
   Para cada servicio que es ejecutado por un agente, es necesario documentar sus características. Específicamente, debemos identificar las entradas, las salidas, las pre-condiciones y las post-condiciones de cada servicio. Las entradas y las salidas a los servicios serán derivadas de manera obvia del *Modelo de Protocolos*. Las pre-condiciones y post-condiciones representan las restricciones de los servicios y se derivan de las características de *Seguridad (Safety)* de un rol. Note que por definición, cada rol debe tener al menos un servicio.
   Los servicios que un agente realizará se derivan de la lista de protocolos, de actividades, de responsabilidades y de las características de *Liveness* de un rol.
   El *Modelo de Servicios* de *GAIA* no prescribe una implementación para los servicios que documenta. El diseñador está libre de realizar los servicios en cualquier framework que estime apropiado. Por ejemplo, puede decidir implementar los servicios como métodos en un lenguaje OO.

5. **Modelo de Comunicación:** Este modelo determina los links de comunicación que existen entre los tipos de agentes. No define qué se envía en los mensajes o cuando estos son enviados, sino que indica simplemente que existen caminos de comunicación. Particularmente, el propósito de un *Modelo de Comunicación* es identificar cualquier embotellamiento potencial en la comunicación, que pueda causar problemas en la ejecución.
   El *Modelo de Comunicación* puede ayudar a garantizar que los sistemas estén débilmente acoplados. En base al modelo de comunicación, puede ser necesario revisar la etapa de análisis y rehacer el diseño del sistema para eliminar estos problemas.
   Un modelo de comunicación del agente es simplemente un grafo dirigido, donde los nodos corresponden a los tipos de agente y los arcos a la comunicación entre ellos.

2.4.5 MASE

MASE (Multiagent Systems Software Engineering) parte del paradigma orientado a objetos y asume que un agente es sólo una especialización de un objeto. La especialización consiste en que los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema. En *MASE* los agentes son sólo una abstracción conveniente, que puede o no poseer inteligencia. En este sentido, los componentes inteligentes y no inteligentes se gestionan igualmente dentro de la misma estructura.

La metodología propone siete fases, a saber: *capturar objetivos, transformar objetivos a roles, aplicar casos de uso, crear clases de agentes, construir conversaciones, ensamblar clases de agente* y *despliegue del sistema*; de las cuales se obtienen diferentes productos, que a su vez son utilizados como entradas a las siguientes fases. Estas fases sólo abarcan las etapas de análisis y diseño del proceso de desarrollo de un *MAS*. En particular, la etapa de análisis abarca las fases: *capturar objetivos, transformar objetivos a roles* y *aplicar los casos de uso*; mientras que la etapa de diseño abarca las fases: *crear clases de agentes, construir conversaciones, ensamblar clases de agente* y *despliegue del sistema*.

La Figura 2.2 muestra las fases de la metodología *MASE* [22].

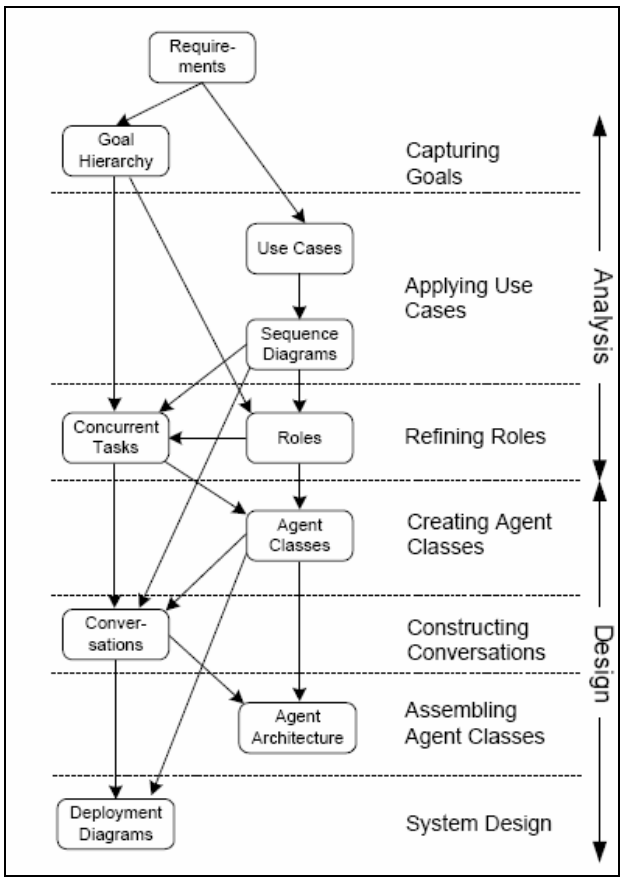

Figura 2.2 Fases de la metodología MASE [22].

A continuación, se detallarán cada de una de fases propuesta por la metodología.

1. **Capturar Objetivos:** Esta fase toma lo más importante de la especificación inicial del sistema y la transforma en un conjunto estructurado de objetivos del sistema. Los objetivos son utilizados para encapsular los requerimientos; porque contienen lo que el sistema trata de alcanzar y por lo general, se mantienen constantes en el resto del proceso de análisis y diseño. El primer paso en esta fase es identificar los objetivos a partir de los requerimientos del sistema. Se comienza extrayendo escenarios de la especificación inicial y describiendo el objetivo de éstos. El siguiente paso es crear casos de uso a partir de los requerimientos existentes o de otros recursos disponibles. Los casos de uso pueden ayudar a recoger más información acerca de los objetivos del sistema o a clarificar la existente. El paso final de esta fase es estructurar los objetivos en un diagrama de jerarquía de objetivos. Los objetivos más importantes deben colocarse en lo más alto del diagrama y todos los sub-objetivos deben pertenecer a un objetivo padre en la jerarquía.

2. **Transformar Objetivos a Roles:** En esta fase los objetivos definidos son transformados en roles y sus tareas asociadas. Cada una de los objetivos están asociados a un rol y cada uno de los roles es ejecutado o representado por una clase de agentes. Los roles están compuestos por: responsabilidades, permisos (disponibilidad de recursos de información), actividades (acciones privadas) y protocolos, y tienen la posibilidad de compartir tareas.

3. **Aplicar Casos de Uso:** El propósito de esta fase es apoyar la posterior construcción de conversaciones en la metodología. Aplicar los casos de uso requiere tomar los casos de uso identificados en la *Captura de Objetivos* y reestructurarlos como diagramas de secuencia. El diagrama de secuencia es utilizado para determinar el conjunto mínimo de mensajes que deben intercambiar los roles. Si un mensaje es traspasado entre dos roles, entonces debe existir la correspondiente comunicación entre ellos. Además, se utilizan para identificar los roles del sistema que participan en los eventos. Cada participante en los diagramas de secuencia es un rol; por lo tanto, si hay un participante en los casos de uso que no es un rol en el sistema, se debe crear un nuevo rol.

4. **Crear Clases de Agentes:** En esta fase se crean las clases de los agentes pertenecientes a la solución. De esta fase de obtiene el *Diagrama de Clases de Agentes* que es muy similar al *Diagrama de Objetos*, aunque las relaciones en el primero se interpretan como conversaciones.

5. **Construir Conversaciones:** En esta fase se utiliza el *Diagrama de Transición de Estados* en los que se modelan los agentes y la conversación entre ellos. Esto se explica ya que no sólo se modela la transición de estado de un agente sino el estado de la conversación entre dos agentes que provoca sus cambios de estado. Una conversación define un protocolo de coordinación entre dos agentes. Específicamente, una conversación consiste en dos diagramas de comunicación de

clases. Un diagrama de comunicación de clases es un autómata de estado finito que define el estado de la conversación de dos clases de agentes participantes.

6. **Ensamblar Clases de Agente:** En esta fase se procede a la interconexión de los subsistemas (componentes del agente). Definición de la arquitectura del agente mediante componentes.

7. **Despliegue del Sistema:** En esta fase se elabora el *Diagrama de Desarrollo*, muy semejante al *Diagrama de Despliegue* de *UML* en el que se representan los diferentes agentes, sus comunicaciones y su despliegue físico.

El proceso de desarrollo se lleva a cabo de forma iterativa y es apoyado por la herramienta CASE *AgentTool*. Esta herramienta permite generar los diversos diagramas propuestos por la metodología y verificar la consistencia entre las conversaciones que sostiene los agentes.

2.4.6 PASSI

PASSI (*Proceso para la especificación e implementación de Sociedades de Agentes*) es una metodología paso a paso, de requerimiento a código, para diseñar y desarrollar sociedades multiagente, integrando modelos y conceptos tanto de Ingeniería de Software Orientada a Objeto como de Inteligencia Artificial, usando notación *UML* [23].

Para comprender esta metodología, es importante tener en cuenta la definición de agente. Un agente es una instancia de una clase agente que es la implementación de software de una entidad autónoma capaz de lograr sus objetivos a través de sus decisiones autónomas, sus acciones y sus relaciones sociales. Un agente puede tener varios roles para alcanzar sus objetivos, siendo un rol una función temporal asumida por el agente en la sociedad mientras busca alcanzar un sub-objetivo.

PASSI se compone de cinco modelos, a saber: *Modelo de Requerimientos del Sistema*, *Modelo de Sociedad de Agentes, Modelo de Implementación de Agentes, Modelo de Código* y *Modelo de Despliegue*, los cuales son desarrollados en las distintas etapas del proceso de desarrollo. El *Modelo de Requerimientos del Sistema* es desarrollado en la etapa de análisis, el *Modelo de Sociedad de Agentes* es desarrollado en la etapa de diseño y los *Modelos de Implementación de Agentes*, *Código* y *Despliegu*e son desarrollados en las etapas de implementación y prueba.

La Figura 2.3 muestra los modelos y fases de la metodología *PASSI* [24].

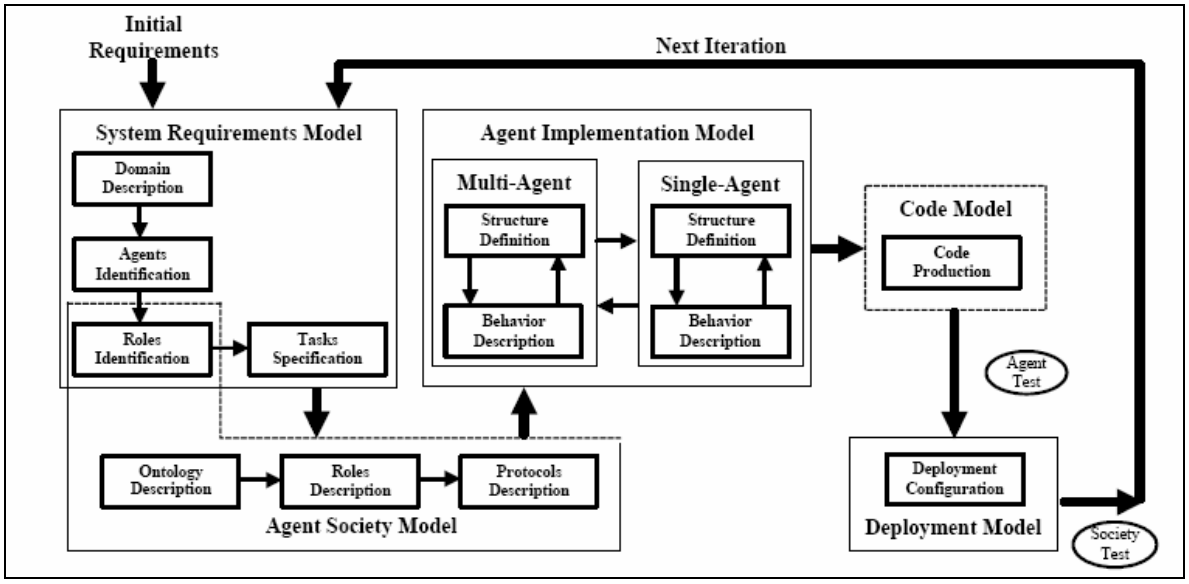

Figura 2.3 Modelos de la metodología PASSI [24].

Como se aprecia en la figura 2.3, cada modelo a su vez esta compuesto por diversas fases que ayudan a desarrollar sus propósitos. A continuación, se detallaran cada uno de los modelos y sus respectivas fases.

1. **Modelo de Requerimientos del Sistema:** Es un modelo de requerimientos del sistema en términos de propósito y organización. Se compone de cuatro fases:

   - **Descripción del Dominio (D.D.):** Esta fase es común tanto en las prácticas de diseño orientadas a objeto [25] como en las orientadas a agente [26], durante el análisis de requerimientos, en la cual se realiza una descripción funcional del sistema, enfocándose en la identificación de los objetivos del sistema. Los requerimientos son expresados utilizando diagramas de casos de usos o a través de la aplicación informal de métodos basados en escenarios, tales como *GBRAM* o *ScenIC* [27]. El resultado es una serie jerárquica de diagramas de casos de uso donde los escenarios podrían ser detallados usando diagramas de secuencia.

   - **Identificación de Agentes (A.Id.):** Muchos métodos se han propuesto en la literatura para identificar agentes. Algunos autores [21] después de discutir los roles definen los agentes asignándoles uno o más roles. *PASSI* propone descomponer el sistema en sus funcionalidades y a partir de éstas, proceder a identificar los agentes agrupando algunas funcionalidades en un agente. De acuerdo con la definición de agente, los agentes se piensan como un caso de uso o como un paquete de ellos. Por lo tanto, a partir de los diagramas de caso de uso de la fase anterior se utilizan paquetes para ilustrar las funcionalidades asignadas a cada agente. Lo más importante es cubrir todos los casos de uso que componen la descripción funcional del

sistema. Este es el primer paso donde esta metodología es fuertemente diferente a las demás.
Desde el diagrama de *Descripción del Dominio*, utilizando un add-in de *Rational Rose (PTK- PASSI Tool Kit)* se identifican los agentes seleccionando la opción "*Identificar Nuevo Agente*" del menú, después de seleccionar todos los casos de uso que lo compondrán. El resultado obtenido es: la creación del *Diagrama de Identificación de Agentes* donde todos los agentes son automáticamente representados como un paquete y sus funcionalidades son descritas por los casos de uso seleccionados como parte del agente, y la creación del *Diagrama de Actividad* desde la especificación de tareas de este agente.

- **Identificación de Roles (R.Id.):** En la Figura 2.3 podemos apreciar que esta fase es parte tanto del *Modelo de Requerimientos del Sistema* como del *Modelo de Sociedad de Agentes*. Esto es debido a que es una descripción funcional (conducta) de los agentes, por consiguiente parte del *Modelo de Requerimientos del Sistema*, y es también una representación de su relación con otros agentes, por consiguiente parte del *Modelo de Sociedad de Agentes.* Se utiliza un conjunto de diagramas de secuencia para ilustrar los escenarios, el cual es definido a partir del análisis de los diferentes caminos que son posibles identificar en el diagrama *A.Id*. Los roles que el agente desempeña no son diferentes del concepto de rol de un objeto en *OO*. Estos son representados en el diseño como objetos en el diagrama de secuencia de la fase *R.Id.*, usando la sintaxis: *<nombre del rol>*:*<nombre del agente>*. De acuerdo al estándar *FIPA* [28], la comunicación entre los distintos agentes está caracterizada por: un agente que envía mensajes y uno que recibe, un protocolo de interacción, una ontología y un lenguaje de comunicación. Los agentes que envían y reciben mensajes son identificados en esta fase para describir los otros elementos de los diagramas de *Descripción de Ontologías* del sistema.

- **Especificación de Tareas (T.Sp.*):*** En esta fase se decide que tareas son necesarias para realizar las funcionalidades descritas en los pasos anteriores. Se tiene un diagrama de actividad para cada agente. Después de haber identificado los agentes y sus roles, se especifican las tareas. Cada tarea se puede realizar con una subclase de la clase principal del agente y cada una de las acciones dentro de las tareas puede realizarse con los métodos de las tareas/clases. Cada diagrama *T.Sp* se divide en dos caminos: en el derecho se colocan las tareas de un agente específico, en el izquierdo se colocan las tareas del resto de los agentes con los que interactúa para representar las relaciones que posee. Si dos agentes interactúan entre sí, el *PTK* lo registra automáticamente en los *T.SP* de ambos agentes.

2. **Modelo de Sociedad de Agentes:** Es un modelo de interacciones sociales y dependencias entre los agentes involucrados en la solución. Desarrollar este modelo involucra tres fases más una del modelo anterior.

   - **Identificación de Roles (R.Id.):** Ver el *Modelo de Requerimientos del Sistema.*

   - **Descripción de la Ontología (O.D.):** Esta fase intenta describir la sociedad de agentes desde el punto de vista ontológico. Dos elementos son importantes: el dominio de la ontología y el intercambio de información entre agentes, para los cuales se tienen los siguientes diagramas:

     i. *Diagrama de Descripción de la Ontología del Dominio (D.O.D.):* La ontología se representa utilizando *XML schema*. El dominio de la ontología se describe en términos de conceptos (entidades del dominio), predicados (que indican el valor de los atributos de las entidades del dominio) y acciones (que puedan operar en el dominio). Representamos estos elementos y sus relaciones usando un diagrama de clase *UML*.

     ii. *Diagrama de Descripción de la Ontología de Comunicación (C.O.D):* Utiliza diagramas de clase para describir el conocimiento de los agentes y la ontología de su comunicación. De acuerdo al estándar *FIPA*, la comunicación está compuesta de *Actos del Habla* de la siguiente forma: *<i, act (j, C)>*, donde *i* es el autor del acto del habla, *act* es el nombre del acto, *j* es el receptor, y *C* es el contenido semántico.

   - **Descripción de Roles (R.D.):** Esta fase modela el ciclo de vida de un agente teniendo los roles que cada agente desempeña, las tareas implicadas, las capacidades de comunicación y las dependencias entre agentes. La Fase *R.D.* produce una colección de diagramas de clases, en donde las clases son utilizadas para representar los roles. Cada agentes es representado con un paquete que contiene clases de roles. Los roles se obtienen componiendo varias tareas en el comportamiento resultante. En este diagrama también describimos el cambio de roles, de comunicaciones y de dependencias entre agentes. Comúnmente, se utilizan sólo los protocolos de interacción del estándar *FIPA*.

   - **Descripción del Protocolo (P.D.):** Varios protocolos de interacción se pueden encontrar en el estándar *FIPA*, pero es posible que los diseñadores necesiten un patrón de interacción específico y por lo tanto decidan crear un nuevo. Si éste es el caso, pueden describir el nuevo protocolo a través de diagramas de secuencia de *AUML*. De esta manera, se definirá el acto comunicativo inicial, las posibles replicas y los pasos que concluirán la comunicación.

3. **Modelo de Implementación de Agentes:** Es un modelo típico de arquitectura en términos de clases y métodos. La diferencia más importante con el enfoque *OO* es que se tienen dos niveles distintos de abstracción: el nivel social (sociedad de agentes) y el nivel de agente. Este modelo está compuesto de las siguientes fases:

   - **Definición de la Estructura del Agente (MA.S.D./S.A.S.D.):** La descripción de la estructura a nivel social (*M.A.S.D.*) hace referencia a la arquitectura general del sistema, en donde podemos encontrar agentes y sus tareas. La descripción de la estructura a nivel de agente (*S.A.S.D.*) se centra en la estructura interna de cada agente, revelando todas sus cualidades y métodos. Es interesante observar que ambos diagramas son construidos automáticamente por el *PTK* usando la información proveniente de los pasos anteriores, lo cual asegura un de alto nivel de coherencia entre el nivel de implementación y las etapas previas de diseño.

   - **Descripción del Comportamiento del Agente (M.A.B.D./S.AB.D.):** La descripción del comportamiento a nivel social (*M.A.B.D.*) se realiza utilizando diagramas de actividades para representar el comportamiento del sistema a nivel de detalle, llegando a considerar los métodos de cada agente/tarea. Cada clase (que representa la implementación de un agente/tarea) se representan como caminos en el diagrama de actividad y cada actividad representa los métodos de la clase. Cada actividad está conectada con otra, para representar el flujo del control y eventos del sistema. Este diagrama representa el nivel más útil de detalle para la representación de un *MAS*. La descripción del comportamiento a nivel de agente (*S.A.B.D.*) se centra en la implementación de los métodos y clases. Éstas se pueden diseñar utilizando un enfoque clásico (por ejemplo diagramas de estado) o incluso utilizando una descripción narrativa semi-formal.

4. **Modelo de Código:** Es un modelo de la solución a nivel de código, el cual sólo requiere la fase de *Generación de Código*. El *PTK* puede generar el código de todas las estructuras (esqueletos) de los agentes, tareas y otras clases incluidas en el proyecto. No utiliza las funcionalidades estándares de *Rational Rose*, puesto que también permite la reutilización de patrones que vienen en su repositorio. El repositorio de patrones consiste en una serie de partes (de agentes y tareas) reutilizables. Por ejemplo, el diseñador puede tomar del archivo un agente genérico (que tenga la capacidad de registrarse el mismo a los servicios básicos de la plataforma) y puede introducirlo en el proyecto real. El repositorio también incluye una lista de tareas que se pueden aplicar a los agentes existentes. Cuando un patrón es introducido en el diseño, no sólo algunos diagramas como los *M.A.S.D., S.A.S.D., M.A.B.D.* y *S.A.B.D.* del *Modelo de Implementación de Agentes* son actualizados, sino que también el código resultante.

5. **Modelo de Despliegue:** Es un modelo de distribución de las partes del sistema, a través de las unidades de hardware de procesado. Esto implica la fase *Configuración del Despliegue (D.C.):* Esta fase es particularmente importante si tratamos con agentes móviles y con problemas significantes en la difusión de los agentes del sistema. El *Diagrama de Configuración del Despliegue* describe donde están localizados los agentes y cuales unidades se necesitan para permitir la comunicación entre agentes. Como es usual, las unidades son mostradas en cajas *3D*. Los agentes se muestran como componentes y la comunicación entre agentes es representada por líneas discontinuas. También se utiliza una extensión de *UML* para representar los agentes móviles que se mueven de un computador a otro.

La actividad de Testing en *PASSI* se realiza en dos niveles: a nivel de agente y a nivel de sociedad de agentes. El *Testing a Nivel de Agente* se dedica a verificar el comportamiento de cada agente con respecto a los requerimientos originales del sistema resuelto por un agente específico. El *Testing a Nivel de Sociedad de Agentes* continua con la validación de los resultados globales de esta iteración y de la integración de los diversos agentes. En la metodología *PASSI* podemos identificar varias iteraciones. Los requerimientos evolutivos/iterativos se utilizan para el diseño incremental del software. Es una práctica común y útil también para el software basado en agentes (*Modelo de Implementación del Agente*). Describe las dependencias entre el nivel de sociedad de agentes (multiagente) y el nivel de agente.

Capítulo 3

Evaluación de metodologías AOSE

3.1 Técnicas de evaluación

En la literatura se han encontrado cinco principales técnicas para la evaluación de metodologías *AOSE*, a saber: *Comparación Basada en Características, Meta-modelado, Métricas, Evaluación Ontológica* y *Técnicas Empíricas de Evaluación* [29].

1. **Comparación Basada en Características:** Esta técnica, también llamada análisis de características, es el enfoque más popular que se utiliza para la comparación de metodologías. Es un método cualitativo que involucra desarrollar una lista de características o criterios, que se considera, debe tener una metodología. La lista se puede utilizar para evaluar metodologías provenientes del mismo paradigma, como por ejemplo metodologías orientadas a agentes, o para evaluar metodologías provenientes de distintos paradigmas, por ejemplo metodologías orientadas a agentes v/s metodologías orientadas a objeto.

   Estos criterios pueden describir una metodología lo suficientemente bien como para realizar una evaluación y comparación con un propósito particular, sin embargo, su selección es subjetiva; ya que no hay consenso universal sobre qué características debe poseer una metodología. La elección de criterios puede reflejar la opinión subjetiva de un grupo particular de investigadores, los que dependen de sus intereses y conocimientos. Además, los criterios son analizados por investigadores que no necesariamente son los mismos que crearon la lista, debiéndose interpretar las descripciones proporcionadas por los creadores de la metodología.
   Por otra parte, esta técnica es relativamente fácil de ejecutar si el conjunto de criterios está bien definido.

2. **Meta-Modelos:** Esta técnica es un enfoque formal utilizado para la comparación de metodologías, la cual utiliza meta-modelos como base para el análisis. En su forma más simple podemos decir que un meta-modelo es un modelo conceptual de una metodología de desarrollo. Por lo tanto, el meta-modelado se puede definir como un proceso de modelado que está en un nivel de abstracción y de lógica más alto que el proceso de modelado estándar. Un meta-modelo captura información sobre los conceptos, las formas de representación y las aplicaciones de una metodología.

   La técnica consiste en utilizar el meta-modelo de la metodología a evaluar o construir uno en caso de que no posea, y compararlo con un meta-modelo de

referencia. Este último puede estar creado o se debe construir en caso contrario. Esta técnica es similar a la *Comparación Basada en Características*, sin embargo la comparación es objetiva, ya que se hace de manera formal al utilizar meta-modelos.

3. **Métricas:** Este enfoque analiza la complejidad de una metodología basándose en sus características. Analiza un meta-modelo formal de una metodología, a través de un conjunto de métricas estándar. Esto puede consistir en analizar el número de construcciones o bloques de una metodología. Por ejemplo, en *UML* son construcciones una clase, un objeto, asociaciones, un mensaje, un estado y una actividad. Estas métricas son calculadas y sus valores son comparados con valores de referencia.

   La principal desventaja de esta técnica es la subjetividad en la elección del número de construcciones a analizar y en su granularidad. Además, muchos trabajos empíricos son necesarios para validar una métrica.

4. **Evaluación Ontológica:** Esta técnica propone utilizar conceptos ontológicos para evaluar metodologías. Usando el modelo ontológico de *Bunge*, las construcciones de una metodología son mapeadas a construcciones ontológicas. La evaluación comprueba si el mapeo entre la metodología y la ontología existe. Si no existe tal mapeo, se detectan en las metodologías debilidades como el exceso o déficit de construcciones.

   En esta técnica una persona externa define las construcciones requeridas, disminuyendo la subjetividad. Sin embargo, no hay pautas para identificar construcciones dentro de una metodología.

5. **Técnicas Empíricas de Evaluación:** Estas técnicas denotan observaciones y proposiciones basadas en el sentido de la experiencia y/o son derivadas de tal experiencia a través de métodos de lógica inductiva, matemática y estadística.

   Dentro de estas técnicas podemos encontrar las encuestas, las cuales se utilizan para recoger datos sobre las actitudes, opiniones, impresiones y creencias de las personas a través del uso de cuestionarios. Aunque esta técnica es muy popular, existen pocas instancias en donde los investigadores la utilicen para evaluar información acerca de las metodologías.

   Otra técnica empírica de evaluación es el uso de laboratorios experimentales. En un laboratorio experimental los investigadores manipulan variables independientes (por ejemplo diferentes técnicas de modelado, diferentes construcciones) y miden los efectos de las variables independientes en las variables dependientes (por ejemplo precisión del modelo, precisión de la interpretación, nivel de confianza, tiempo tomado). Lo útil de esta técnica es la habilidad de controlar las variables que intervienen y la confusión.

Por último, se encuentran los casos de estudio, los cuales intentan capturar y comunicar la realidad de un entorno particular en un tiempo específico. Esta técnica no requiere que los investigadores intervengan en la operación normal de las organizaciones.

En esta técnica la interpretación de los investigadores puede ser subjetiva, sin embargo, la riqueza de la información entregada por un caso de estudio no puede ser alcanzada por las encuestas o laboratorios experimentales.

3.2 Frameworks de evaluación

Cuando se realiza una evaluación utilizando la técnica de comparación basada en características, se definen un conjunto de criterios y/o guías que se espera que una metodología incluya y se evalúa si ésta los posee o no. En la comunidad orientada a agente estos conjuntos de criterios y/o guías son conocidos como frameworks de evaluación.

Dentro de los frameworks de evaluación encontrados en la literatura podemos citar los siguientes: El de *Joaquín Peña et al.* [30], el de *Arnon Sturm et al.* [31], y el de *Pedro Cuesta et al.* [3].

3.2.1 Joaquín Peña et al.

Este autor propone un *RFC (Request For Comments)* acerca de pautas, técnicas y artefactos de modelado para lidiar con la complejidad en la etapa de análisis de las metodologías *AOSE* [30]. En él la complejidad se relaciona con las interacciones entre los agentes y está estrechamente ligada a los aspectos de la organización.

Este enfoque está específicamente desarrollado para tratar con organizaciones complejas y se basa en tres principios, inicialmente identificados por *N. Jennings*:

1. **Abstracción:** Define los modelos simplificados del sistema, enfatizando algunos detalles mientras que evita otros para enfocar la atención del diseñador.

2. **Descomposición:** Aplica el principio “divide y vencerás” para limitar el alcance del diseñador a una parte del problema.

3. **Composición:** Identifica y maneja las correlaciones entre los diversos subsistemas. Es posible agrupar varios componentes básicos para tratarlos como unidades de análisis de alto nivel.

Estos principios han sido considerados en el nivel de *Modelado del Proceso*. Para esto, se proponen una serie de características que debieran proporcionar las metodologías *AOSE* en la etapa de análisis. Estas características se dividen en: *Técnicas*, que son procedimientos para transformar modelos, *Artefactos de Modelado*, que son íconos para representar gráficamente el sistema y *Pautas*, que indican las mejores prácticas para lidiar con la complejidad usando las técnicas y artefactos de modelado.

Además, el *RFC* presenta las tres clases de principios según aspectos estáticos y dinámicos tales como: *Conocimiento*, modela las relaciones entre los agentes del sistema desde el punto de vista de las interacciones, y *Comportamiento*, modela el orden de aparición de estas relaciones en un cierto tiempo.

Este framework sólo considera la etapa de análisis para la evaluación. Además, deja de lado aspectos como: herramientas de soporte, ciclos de vida, ontologías, características de movilidad, entre otros.

3.2.2 Arnon Sturm et al.

Este framework está enfocado en la evaluación de metodologías orientadas a agentes de propósito general, tales como *GAIA*, *MASE*, *TROPOS*, *PASSI*, entre otras [31]. Su principal objetivo es que pueda ser utilizado tanto por una organización, para seleccionar una metodología de desarrollo para aplicaciones orientadas a agentes, como también por los investigadores, para ayudar a examinar las diferencias y similitudes entre las diversas metodologías *AOSE*.

Considera que una metodología es un conjunto de pautas y actividades: un completo proceso de ciclo de vida, un comprensivo conjunto de conceptos y modelos, un conjunto de técnicas (reglas, guías y heurísticas), un conjunto de entregables, un lenguaje de modelado, un conjunto de métricas, aseguramiento de la calidad, estándar de códigos (y otros), consejos de reutilización y guías para la administración de proyectos.

Para la evaluación utiliza cuatro de los principales aspectos de una metodología: conceptos y propiedades, notación y técnicas de modelado, proceso y pragmática.

1. **Conceptos y Propiedades:** Este criterio evalúa cómo una metodología se adhiere a las nociones básicas (conceptos y propiedades) de agentes y sistemas multiagente. Para revisar como cada metodología *AOSE* define ésos conceptos, se mira esencialmente la magnitud en que la metodología apoya los conceptos y propiedades, o a cuál (es) apoya la construcción de los agentes que poseen estos atributos.

   Un concepto es una abstracción o una noción inferida o derivada de una instancia específica dentro de un dominio. Una propiedad es una capacidad o característica especial. Los conceptos relacionados con los agentes y los *MAS* son de gran importancia para las metodologías orientadas a agentes en general y para los lenguajes de modelado orientados a agentes en particular. Debido a que no hay acuerdo (aún) en la comunidad de agentes acerca de los conceptos básicos de agentes y sistemas multiagente, los siguientes son los conceptos acordados según los cuales serán evaluadas las metodologías *AOSE*:

- **Conceptos Generales:** Los conceptos generales considerados son:

  i. Autonomía, es la habilidad de un agente para operar sin supervisión.
  ii. Reactividad, es la habilidad de un agente para responder en un tiempo y manera determinada para cambiar el entorno.
  iii. Proactividad, es la habilidad de un agente para perseguir nuevas metas.
  iv. Habilidad Social, es la habilidad de un agente para interactuar con otros agentes, enviando y recibiendo mensajes, encaminándolos y entendiéndolos.

- **Bloques de construcción Básicos:** Los bloques de construcción básicos considerados son:

  i. Agente, un programa de computadora que puede aceptar tareas, que puede entender qué acciones realizar en orden de ejecutar estas tareas y que puede realmente ejecutar estas acciones sin supervisión. Es capaz de ejecutar un conjunto de tareas y proveer un conjunto de servicios.
  ii. Creencia, un hecho que se cree cierto acerca del mundo.
  iii. Deseo, un hecho del cual su valor actual es falso y que el agente por su propio deseo prefiere que sea verdadero. Los deseos dentro de una entidad pueden ser contradictorios. Un amplio uso de un deseo es una meta, las cuales deben ser consistentes dentro de un agente.
  iv. Intención, un hecho que representa la manera de realizar un deseo. A veces hace referencia a un plan.
  v. Mensaje, una manera de intercambiar hechos u objetos entre entidades.
  vi. Norma, una guía que caracteriza una sociedad. Un agente que desee ser miembro de una sociedad requiere que siga todas las normas dentro de esta. Una norma puede hacer referencia a una regla.
  vii. Organización, un conjunto de agentes trabajando en conjunto para alcanzar un propósito común. Una organización consiste en reglas que caracterizan a los agentes miembros de la organización.
  viii. Protocolo, un conjunto de mensajes ordenados, que en conjunto definen patrones admisibles de un tipo particular de interacción entre entidades.
  ix. Rol, una representación abstracta de la función de un agente, servicio o identificación dentro de un grupo.
  x. Servicio, una interfaz que es abastecida por un agente al mundo exterior. Es un conjunto de tareas que juntas ofrecen alguna operación funcional. Un servicio puede consistir en entregar otros servicios.
  xi. Sociedad, una colección de agentes y organizaciones que colaboran para promover sus metas individuales.

xii. Tarea, una pieza de trabajo que puede ser asignada a un agente o ejecutada por éste. Puede ser una función a ejecutar y puede tener restricciones de tiempo. A veces se hace referencia a una acción.

2. **Notación y Técnicas de Modelado:** Si los conceptos son la base para cualquier metodología *AOSE*, las notaciones y técnicas de modelado para representar el diseño en términos de estos conceptos son el componente base de cualquier metodología. Este criterio apunta a evaluar las propiedades que la notación y la técnica de modelado de la metodología debieran incluir.

   Una notación son conjuntos técnicos de símbolos utilizados para representar elementos dentro de un sistema. Una técnica de modelado es un conjunto de modelos que describen un sistema en diferentes niveles de abstracción y en diferentes aspectos del sistema. Las propiedades de las notaciones y técnicas de modelado consideradas para la evaluación de una metodología son las siguientes:

   - **Accesibilidad:** Es un atributo que se refiere a la facilidad o simplicidad de entendimiento y uso de una metodología. Se acentúan las capacidades de uso de un nuevo concepto tanto de expertos como novatos.

   - **Capacidad de Análisis:** Es la capacidad de revisar la consistencia interna o implicaciones de los modelos, o de identificar aspectos que parecen nucleares, tales como las interrelaciones entre operaciones que parecen sin conexión. Esta capacidad es usualmente apoyada por herramientas automáticas.

   - **Administración de la Complejidad (abstracción):** Es la habilidad para lidiar con varios niveles de abstracción (por ejemplo varios niveles de detalle). A veces son necesarios requerimientos de alto nivel y en otras ocasiones, se requiere más detalle. Por ejemplo, analizando el nivel más alto de diseño de un sistema multiagente, uno podría querer entender cuales agentes están dentro del sistema, pero no necesariamente cuales son sus atributos y características. Sin embargo, cuando nos concentramos en una tarea específica de un agente el detalle es mucho más importante que la arquitectura del sistema.

   - **Ejecutabilidad (y testeabilidad):** Es la capacidad de ejecutar una simulación o generar un prototipo de al menos algunos aspectos de una especificación. Estos podrían demostrar posibles comportamientos de un sistema modelado, y ayudar a los desarrolladores a determinar como los requerimientos pretendidos han sido expresados.

   - **Expresividad (y aplicabilidad a múltiples dominios):** Es la capacidad de presentar conceptos del sistema que se refieren a: la estructura del sistema, el conocimiento encapsulado dentro del sistema, la ontología del sistema, el flujo de datos dentro del sistema, el flujo de control dentro del sistema, las actividades concurrentes (y agentes) dentro del sistema, las

restricciones de recurso dentro del sistema (por ejemplo tiempo, *CPU* y memoria), la arquitectura física del sistema, la movilidad de los agentes, la interacción del sistema con sistemas externos, y la definición de interfaces de usuario.

- **Modularidad (incrementabilidad):** Es la habilidad para especificar un sistema en una manera iterativa incremental. Esto es, cuando nuevos requerimientos son añadidos no deberían afectar las especificaciones existentes, pero podrían utilizarlas.

- **Exactitud:** Es un atributo de no ambigüedad. Permite a los usuarios evitar la mal interpretación de modelos existentes.

3. **Proceso:** Según lo discutido anteriormente, las notaciones y técnicas de modelado están consideradas como parte obligatoria de cualquier metodología. Sin embargo, en la construcción de un sistema de software se acentúan una serie de actividades y pasos realizados como parte del ciclo vital del software. Estas actividades y pasos forman el proceso que asiste a los analistas de sistemas, a los desarrolladores y a los administradores en el desarrollo de software.
   Un proceso es una serie de acciones, cambios y funciones que cuando son ejecutadas resultan en un sistema. Este criterio trata acerca del proceso de desarrollo de una metodología, considerando:

   - **Contexto de Desarrollo:** Especifica como una metodología es útil en crear nuevo software, hacer reingeniería o ingeniería inversa al software existente y diseñando para reutilización de componentes.

   - **Cobertura del Ciclo de Vida:** La cobertura de un ciclo de vida de una metodología particular involucra averiguar que elementos de desarrollo de software son tratados dentro de la metodología. Cada metodología puede tener elementos que son útiles para varias etapas del ciclo de vida de desarrollo. En este framework se consideran las siguientes etapas de un ciclo de vida:

     i. Reunión de Requerimientos, es la etapa del ciclo de vida en la cual se crean las especificaciones (generalmente en texto libre) de las necesidades desde el sistema.
     ii. Análisis, es la etapa del ciclo de vida que describe externamente las características observables del sistema, por ejemplo funcionalidad, ejecución y capacidad.
     iii. Diseño, es la etapa del ciclo de vida que define la manera en la cual el sistema alcanzará sus requerimientos. Los modelos definidos en la etapa de análisis son redefinidos o trasformados dentro de los modelos de diseño que describen la naturaleza física y lógica del producto de software.
     iv. Implementación, es la etapa del ciclo de vida que convierte los modelos de diseño desarrollados en software ejecutable dentro del

entorno del sistema. Esta etapa involucra los manuales de usuario de las unidades programadas, la generación automática de código, o el ensamblado de las partes construidas.

v. Prueba, es la etapa del ciclo de vida que se enfoca en asegurar que cada entregable de cada etapa esté conforme a los requerimientos del usuario.

Tener definidas las etapas de desarrollo no es suficiente para el uso de una metodología. Una metodología debe fomentar la elaboración de actividades dentro del ciclo de vida de desarrollo, para proveer al usuario de la metodología medios para usarla adecuada y eficientemente. Proveyendo una descripción adecuada de varias actividades durante el ciclo de vida de desarrollo se podría aceptar el uso de una metodología e incrementar su aceptabilidad como un enfoque bien formado de ingeniería. Por consiguiente, se sugiere examinar el proceso de una manera detallada. Estos detalles pueden ser provistos respondiendo las siguientes preguntas relativas a la metodología evaluada:

i. ¿Cuáles son las actividades dentro de cada etapa de la metodología? Por ejemplo, una actividad puede ser la identificación de un rol, una tarea, etc. Puede consistir en una guía que ayude al desarrollador a alcanzar sus metas (en el desarrollo del sistema).
ii. ¿Qué entregables son generados por el proceso? Esta pregunta se refiere principalmente a la documentación. Por ejemplo, que modelos son especificados y pueden ser entregables al cliente.
iii. ¿El proceso provee verificación? Esta pregunta comprueba como la metodología tiene reglas para verificar la adherencia de sus entregables a los requerimientos.
iv. ¿El proceso provee validación? Esta pregunta comprueba como la metodología tiene reglas para validar que los entregables de una etapa sean consistentes con su etapa antecesora.
v. ¿Son proporcionadas guías de aseguramiento de la calidad?
vi. ¿Hay guías para la administración de proyectos?

4. **<u>Pragmática:</u>** La pragmática trata aspectos prácticos del desarrollo y uso de una metodología. Este criterio trata de la pragmática de adoptar la metodología para un proyecto dentro de una organización. En particular el framework sugiere seguir los siguientes aspectos:

   - Recursos, ¿qué recurso están disponibles para apoyar la metodología? ¿hay un libro disponible? ¿están los grupos de usuario establecidos? ¿hay entrenamiento o consultaría ofrecido por el vendedor y/o terceras partes? Además, ¿están disponibles herramientas *CASE* para apoyar la metodología (por ejemplo editores de gráficos, generadores de código y verificadores)? Estos aspectos pueden ser examinados para permitir a un proyecto/organización apuntar a adoptar una metodología para comprobar

los recursos (en términos de entrenamiento y presupuesto) requeridos y alternativas para adquirirlos.

- Habilidad requerida, ¿cuál es el background requerido para aprender la metodología? Una característica distinguida de muchas metodologías es el nivel de sofisticación matemática requerida para aprovecharla por completo. ¿La metodología asume conocimiento en alguna disciplina? Este aspecto podría ser examinado para permitir que un proyecto/organización apunte a adoptar una metodología para comprobar si la calificación requerida para usarla son encontradas para cada usuario candidato.

- Aplicación de Lenguaje (paradigma y arquitectura), ¿la metodología apunta a un lenguaje de implementación particular? Esto es, ¿la metodología está basada en conceptos de una arquitectura o lenguaje de programación? Por ejemplo una metodología podría estar limitada a aplicaciones basadas en *BDI*. Este criterio comprueba como una metodología se adhiere a una infraestructura y conocimiento de un proyecto/organización.

- Dominio de Aplicación, ¿es adecuado el uso de la metodología para un determinado domino de aplicación (por ejemplo tiempo real, sistemas de información)? Este aspecto será examinado para comprobar como la metodologías se adhiere al dominio deseado.

- Escalabilidad, ¿puede la metodología o un conjunto de ella ser usado para manejar varios tamaños de aplicación? Por ejemplo, ¿puede proveer una versión ligera para simplificar problemas? Este aspecto será examinado para comprobar como la metodología es apropiada para manejar la escala de aplicaciones específicas dentro de un proyecto/organización.

Para permitir la evaluación de las propiedades examinadas por el framework, el autor propone un ranking cuya escala va desde 1 a 7 (ver Tabla 3.1).

Tabla 3.1 Ranking para la evaluación de propiedades del framework de Sturm.

Ranking	Descripción
1	Indica que la metodología no se dirige a esa propiedad.
2	Indica que la metodología se dirige a esa propiedad, pero no se entregan detalles.
3	Indica que la metodología se dirige a la propiedad a un alcance limitado. Esto es, muchos aspectos que se relacionan con las propiedades no son dirigidos.
4	Indica que la metodología se dirige a esa propiedad, pero carece de algunos aspectos principales.
5	Indica que la metodología se dirige a la propiedad, sin embargo, carece de uno o dos aspectos principales relatados en propiedades especificas.
6	Indica que la metodología dirige la propiedad sin mayor problema.
7	Indica que la metodología dirige completamente la propiedad.

3.2.3 Pedro Cuesta et al.

Los criterios utilizados por este framework están clasificados en cinco grupos de acuerdo a su área de interés [3]. El primer grupo es llamado *Proceso de Desarrollo*, el cual incorpora aspectos generales de la metodología y otros relacionados con las etapas en la construcción del sistema. El segundo, *Vistas del Modelo,* intenta reflexionar sobre los conceptos de la metodología y su representación. El tercer grupo, *Agente,* trata las características individuales del agente consideradas por metodología bajo evaluación. Finalmente, los grupos de *Características Adicionales* y *Documentación*, incorporan otros aspectos de interés como la calidad de la documentación proporcionada o de las extensiones definidas para la movilidad, las ontologías, etc.

1. **Proceso de Desarrollo:** Este grupo presenta aspectos relacionados con la construcción del *MAS*. Estos aspectos pueden resumirse en las siguientes preguntas: ¿cuáles son las etapas propuestas por la metodología? y ¿qué tipo de actividades deben realizarse en cada etapa? En particular, los siguientes aspectos son considerados en este grupo:

   - **Dominio de Aplicación:** Evalúa si la metodología está propuesta para cualquier dominio de aplicación o si es para un dominio específico.

   - **Área de Aplicación:** Este aspecto recoge la información presentada por los autores sobre la aplicación de la metodología en diversas áreas. Se muestra una lista de las áreas, subrayando cada una sólo si es mencionada

por los autores o si los resultados de la metodología en esta área están bien documentados (manuales, libros, etc.).

- **Sistemas Abiertos:** Intenta evaluar si la metodología se piensa o no para sistemas abiertos, es decir, si permite la adición o el retiro dinámico de agentes, o sus características, mientras el sistema esté funcionando.

- **Tipo de Ciclo de Vida:** Describe qué tipo de modelo de ciclo de vida se aplica en el proceso del desarrollo: cascada, espiral, incremental, etc.

- **Etapas del Proceso:** Este es el punto principal de la evaluación. Aquí se describe cómo el ciclo de vida es cubierto por la metodología. Se presentan las etapas de la metodología y qué clase de actividades se deben realizar en cada etapa. Para tratar actividades, se ha utilizado la taxonomía clásica de la Ingeniería de Software: Análisis de Requerimientos, Diseño, Implementación y Prueba.

  i. Las actividades de requerimiento se relacionan con la especificación inicial de las necesidades del cliente. El análisis incluye cualquier actividad relacionada con la descripción de "*lo que el sistema debe hacer*". Las actividades de diseño se orientan a definir cómo el sistema hace lo que debe hacer, y cómo el modelo final del sistema se obtiene. Las actividades de implementación están relacionadas con la generación de código. Finalmente, las actividades de prueba intentan comprobar y validar el sistema bajo construcción.
  ii. Cada una de estas actividades se califican con el término: *Enfocado* si la metodología se orienta completamente a ese tipo de actividades en la etapa bajo consideración, *No Enfocado* si la etapa no considera actividades de esta clase y *Enfocado Parcialmente* en otro caso.
  iii. Además, un aspecto significativo es si la metodología considera la participación del usuario en el proceso de desarrollo y en qué momento ocurre. La participación del usuario es esencial para obtener el producto final deseado; ya que es señalada por diversas metodologías nuevas de la Ingeniería de Software.
  iv. Este framework considera si la metodología confía en el usuario en cada etapa, y se utiliza el término *Desconocido* cuando esta información no está clara o disponible.

- **Herramientas de Soporte y Generación de Código:** Otro aspecto importante es la disponibilidad de herramientas de apoyo de la metodología. La existencia de herramientas se evalúa en lo referente a las actividades en las cuales se utiliza la herramienta. La evaluación indica simplemente si hay o no herramientas para cada tipo de actividad y cómo ésta es apoyada por la herramienta.

2. **Vistas del Modelo:** Esta sección intenta evaluar los diagramas y las técnicas propuestos por la metodología para modelar el sistema. También se consideran las relaciones entre los diagramas o entre diversas vistas del sistema, la evolución del modelo en el ciclo de vida y otros aspectos.

   - **Conceptos y su Representación:** Al modelar el sistema muchos conceptos deben ser utilizados y cada uno de ellos necesita una representación. Este punto presenta una lista de los conceptos usados y en donde (diagrama o modelo) se representan. Esta lista se detalla para cada etapa de la metodología, relacionando los diversos diagramas y conceptos manejados por cada una de ellas. El punto final, *Interacción Humana*, evalúa si las interacciones entre los usuarios y el sistema se consideran específicamente en la metodología.

   - **Relaciones entre los Modelos*:*** La metodología puede proporcionar una forma de derivar una nueva vista del modelo existente, o puede definir las reglas para garantizar la consistencia y la corrección, cuando dos vistas están relacionadas. Este punto intenta destacar estas características.

   - **Entregables:** En este punto se evalúa la cobertura proporcionada por las diversas vistas del modelo, con respecto al ciclo de vida entero, estudiando cuáles son los entregables proporcionados por la metodología en las diversas etapas. Se puede decidir si el modelo satisface todas las etapas propuestas o si algunas de ellas están débilmente cubiertas. En este último caso, los modelos se deben complementar con prototipos o con descripciones escritas en lenguaje formal, estructurado o natural.

3. **Agente:** En esta sección se considera la manera en la cual la metodología define las características del agente. Un punto fundamental en la definición de un *MAS* es la descripción de los componentes individuales del sistema, es decir, de los agentes. Las características consideradas son:

   - **Concepto de Agente:** El concepto de agente propuesto por la metodología es relevante porque introduce el marco conceptual del desarrollo. Muchas características individuales de los agentes se pueden introducir en la definición, si la metodología se piensa para una arquitectura específica de agente por ejemplo *BDI*) o si sólo se definen tipos específicos de agentes, etc.

   - **Atributos del Agente:** Este punto esta relacionado estrechamente con el *Concepto del Agente*; ya que maneja las características intrínsecas que la metodología utiliza: autonomía, sociabilidad, reactividad, pro actividad, inteligencia u otras.

   Además, los objetivos alcanzados por los *MAS* deben ser más que la suma de los objetivos individuales de los agentes, debido a la interacción y a la cooperación de los agentes. Este aspecto es estudiado y evaluado considerando:

- **Tipos de Comunicación:** En este punto la interacción entre agentes se evalúa en un alto nivel de la abstracción. La metodología puede permitir solamente la comunicación entre agentes *(A-A)* (heterogénea u homogénea) u otro tipo de comunicación, tales como la comunicación entre humanos y agentes *(A-H)* o la comunicación entre otros sistemas de software y agentes *(A-O)*.

- **Protocolos de Comunicación:** La comunicación puede seguir protocolos bien conocidos que pueden ser predefinidos en la metodología.

- **Cooperación:** Este criterio muestra los tipos de interacción entre agentes, los cuales se pueden definir utilizando la metodología: negociación, delegación de tareas, etc.

- **Organización del Agente:** Los agentes pueden tener una estructura interna con influencia en la comunicación. Esta organización puede ser jerárquica, par a par, etc.

4. C**aracterísticas Adicionales:** Esta sección trata las extensiones que normalmente proponen las metodologías para tratar aspectos importantes de los *MAS*, tales como ontologías, características de movilidad y características adicionales.

5. **Documentación:** Un aspecto importante a considerar cuando se proponen nuevas metodologías, es cómo éstas se documentan. Para evaluar este punto se toman las siguientes características:

   - **Documentación Disponible:** La documentación proporcionada por los autores (papers, sitio Web, etc.) se evalúa y se califica con: Bueno, Suficiente o Pobre; donde *Pobre* significa que la documentación proporcionada no es bastante para hacer la metodología totalmente comprensible.

   - **Casos de Estudio Presentados:** Evalúa otro aspecto importante al documentar una metodología, esto es, los ejemplos proporcionados. Se evalúa como *Trivial* si hay solamente pequeños ejemplos para demostrar aspectos particulares, como *Parcial* en el caso que los ejemplos son más completos pero no cubren el ciclo de vida completo propuesto o *Completos* si cubren el ciclo de vida entero.

Capítulo 4

Alcance de la comparación

4.1 Técnicas de evaluación

Para la realización de la evaluación de metodologías se utilizarán: *Técnica de Comparación Basadas en Características* y *Técnica Empírica de Evaluación*.

La utilización de la primera se fundamenta en que la mayoría de los trabajos encontrados en la literatura utiliza esta técnica para realizar comparaciones. Para llevar a cabo la evaluación bajo esta técnica se debe elegir al menos un frameworks de evaluación. En el caso concreto de este proyecto se han elegido dos frameworks de evaluación, a saber, el de *Pedro Cuesta et al.* y el de *Arnon Sturm et al*. Esta elección obedece a que son los más completos que se encuentran en la literatura, en cuanto a los criterios que estos consideran y a la documentación proporcionada.

Por otra parte, la utilización de técnicas empíricas de evaluación se fundamenta en el uso de un caso de estudio para evidenciar de mejor manera las principales fortalezas, desventajas, concordancias y diferencias entre las metodologías a evaluar. La elección del caso de estudio será tratada en mayor detalle en la sección 4.3.

4.2 Metodologías a comparar

Para realizar el análisis comparativo de metodologías orientadas a agentes se evaluarán dos metodologías, a saber, *MASE* y *PASSI*. La principal razón de esta elección es que son fuertemente utilizadas para el desarrollo de sistemas multiagente, debido a que ambas:

- Consideran las características de noción débil de un agente (autonomía, habilidad social, relatividad y proactividad). Además, *PASSI* incluye la característica de movilidad en los agentes, a diferencia del resto de las metodologías. A pesar que *MASE* no la integra actualmente, se están realizando trabajos futuros en donde se pretende su inclusión.

- Incorporan un proceso de desarrollo. *MASE* considera las etapas de análisis, diseño y parte de la implementación (generación de código). *PASSI* considera estás mismas etapas más la etapa de prueba.

- Utilizan *UML* como lenguaje de modelado. Esta razón es importante; ya que *UML* es un lenguaje de modelado aceptado ampliamente tanto en el ámbito académico como en la industria del software.

- Incorporan herramientas de soporte que facilitan el proceso de desarrollo de un *MAS*. *MASE* posee *AgentTool* y *PASSI* el *PASSI Tool Kit (PTK)*.

- Son independientes de la arquitectura de agentes, lo que posibilita la selección entre varias opciones.

- Están bien documentadas. Existen muchos papers en la literatura en donde se explican estas metodologías (por ejemplo, para *MASE* [22] y para *PASSI* [23], entre otros). Además, ambas incorporan casos de estudio para mostrar sus características. Por ejemplo, se ha utilizado *MASE* para desarrollar un sistema multiagente para la gestión de reuniones [32] y *PASSI* para desarrollar el caso de estudio *Juul Møller Bokhandel A/S* [33] y sistemas biológicos [34].

4.3 Caso de estudio

Las publicaciones que describen a las metodologías consideradas para la evaluación no utilizan un ejemplo común para mostrar sus características más destacables, sino que eligen uno particular. Sin embargo resulta muy difícil poder comparar sin un ejemplo común. Dentro del marco de este proyecto se considerará como caso de estudio el desarrollo de un sistema de transporte de pasajeros en respuesta a la demanda *(DRTS - Demand Responsive Transport Systems)* basado en el paradigma de agentes, para mostrar las características y la forma de trabajo de las metodologías anteriormente seleccionadas.

4.3.1 Transporte de pasajeros en respuesta a la demanda

Los servicios *DRT* proporcionan un sistema de transporte adicional orientándose hacia las necesidades de diferentes usuarios. El uso de servicios de transporte flexible, donde las rutas, horas de partida, vehículos e incluso los operarios, pueden ser asignados de acuerdo a las condiciones de la demanda, permiten proveer un servicio más orientado al usuario y una estrategia efectiva en costos. La adaptación de los servicios de transporte para hacerlos coincidir con la demanda actual facilita el ahorro de costos a los operarios, gremios y pasajeros.

Los *DRT* pueden ser vistos como un elemento más, dentro de una cadena de servicio intermodal más grande, proporcionando movilidad local y complementándose con otras formas de transporte convencional (por ejemplo buses y tranvías regulares y, trenes regionales). En este contexto, los *DRT* proporcionan un rango de soluciones intermedias de transporte, cubriendo la brecha existente entre los servicios tradicionales de buses públicos y los taxis individuales (ver Figura 4.1).

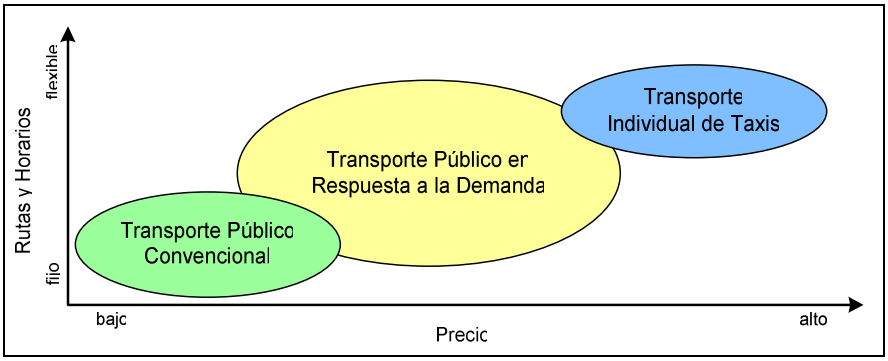

Figura 4.1 Servicios de movilidad.

El ámbito de este transporte intermedio ha estado generalmente orientado a cumplir necesidades de movilidad dispersa, tales como, períodos de baja demanda (por ejemplo en las noches o en fines de semana), áreas de baja densidad poblacional (por ejemplo. zonas rurales), o cuando los usuarios finales se encuentran dispersos entre la población general (por ejemplo discapacitados, ancianos, enfermos, turistas, estudiantes). Pero, los *DRT* no sólo aspiran a solucionar los problemas de movilidad mencionados, ya que, en las áreas urbanas existen períodos de tiempo, sectores de la ciudad o grupos de usuarios, que quedan fuera del alcance del transporte convencional.

4.3.2 Requerimientos funcionales del sistema de transporte

El *DRTS* atiende solicitudes de transporte provenientes desde un conjunto de clientes, las que deben ser cumplidas por una flota de vehículos. El servicio de transporte ofrecido consiste en recoger al cliente en un punto de origen (*lugar de recogida)* y conducirlo a un punto de destino (*lugar de entrega*). Esto se debe realizar en los intervalos o ventanas de tiempo que el cliente establezca para ser recogido en el lugar de recogida (*tiempo de recogida*) y dejado en el lugar de entrega (*tiempo de entrega*). Además, el servicio puede incluir características como el uso compartido o exclusivo del vehículo, lugar para silla de ruedas o algún otro servicio complementario.

El sistema de transporte mencionado considera flotas heterogéneas de vehículos, compuestas por buses, minivans, vehículos para discapacitados, taxis, entre otros. Estos vehículos se caracterizan por distintas propiedades, pero en general tienen aspectos en común como: capacidad limitada de pasajeros, lapsos de tiempo de disponibilidad durante el día, y un área de cobertura geográfica. Pero también, pueden tener características adicionales como: distintos tipos de asientos, baño, aire acondicionado o servicios complementarios como bar, transporte de bicicletas, entre otros. Estas propiedades afectan la comodidad del cliente y por consiguiente su percepción del servicio de transporte recibido.

Para cumplir con servicio de transporte demandado, el sistema debe permitir:

- Atender un conjunto de solicitudes de transporte provenientes de los clientes (las que están geográficamente distribuidas), asignarlas a los vehículos y planificarlas

de acuerdo a las restricciones de tiempo establecidas por el cliente y de la capacidad máxima de pasajeros por vehículo, la cual no puede ser excedida. Es importante mencionar que el sistema no debe considerar restricciones acerca de la capacidad mínima de pasajeros por vehículo.

- Proporcionar al cliente un canal de comunicación con el resto del sistema de transporte. El cliente deberá poder comunicar cualquier eventualidad que pudiera aparecer, por ejemplo, atraso del cliente en el lugar de recogida o cancelación del viaje. Por otra parte, el cliente recibe a través del sistema cualquier información referente a las solicitudes de viajes y a cambios en la planificación original del viaje, tales como atrasos y fallas de los vehículos, cambios relevantes en las condiciones de tráfico, entre otros.

- Capturar y mantener toda la información asociada a las solicitudes de transporte de cada cliente (tiempo de recogida, tiempo de entrega, lugar de recogida, lugar de entrega, tipo de vehículo, uso exclusivo del vehículo, número de pasajeros a transportar, tipos de asientos, aire acondicionado, baño, etc.), y su respectiva función de utilidad (tiempo de espera del cliente, tiempo de exceso de viaje del cliente).

- Capturar y mantener los datos del cliente y sus preferencias acerca de situaciones contingentes tales como atrasos y fallas de los vehículos, desviaciones, congestiones, entre otros. Los requerimientos de transporte y las preferencias de cada cliente podrán utilizarse para que el sistema actué durante todo el proceso en nombre del cliente real (como si fuera un asistente personal de viaje) o simplemente como un mero delegado de sus decisiones. Esto dependerá del grado de autonomía que el cliente le proporcione al sistema.

- Proporcionar al vehículo y su conductor un canal de comunicación con el resto del sistema de transporte. El conductor deberá informar acerca del estado y progreso del vehículo a lo largo del día y cualquier contingencia que pudiera aparecer a lo largo del viaje, por ejemplo, la falla del vehículo, la ausencia del cliente en el lugar de recogida, entre otras. Por otra parte, el conductor recibe a través del sistema cualquier información referente a cambios en la planificación original, tales como transportar un nuevo cliente, atrasos y cancelación de los clientes, cambios relevantes en las condiciones de tráfico, entre otros.

- Proporcionar al operador un canal de comunicación con el resto del sistema de transporte, para capturar y mantener toda la información asociada al tipo de servicio de transporte ofrecido por cada vehículo (tiempo de recogida, tiempo de entrega, dirección de recogida, dirección de entrega, lugares requeridos, tiempo servicio de recogida, tiempo servicio de entrega y máximo tiempo de viaje) y a su respectiva función de utilidad (tiempo total de espera entrega, tiempo total de ocio del vehículo, tiempo total de exceso de viaje y tiempo total de viaje del vehículo).

- Proporcionar información referente al estado de las rutas (congestiones, desvíos) y a la zona geográfica cubierta a los diferentes usuarios del sistema (localización de direcciones y paradas, nombres de calles y distancias entre localizaciones). Para realizar esta labor, el sistema se comunica con un sistema de información de tráfico. Con esta información el sistema puede replanificar la programación inicial de los vehículos adaptándose mejor a nuevas situaciones.

- Asignar rutas de viaje a los distintos vehículos. Dichas rutas deben considerar la distancia más corta entre cada punto visitado por un vehículo, además de considerar el tiempo de traslado desde un punto a otro. El sistema de información de tráfico le entregará al sistema toda información referente a las distancias actuales entre cada para de puntos de detención y su tiempo de traslado correspondiente.

- Capturar y proporcionar información referente a: la descripción del servicio entregado (detalle del viaje, tarifas), los clientes (datos personales), la modalidad de pago (pago a bordo o de antemano) y opciones (tarjeta de crédito, efectivo, boleto). Esta información la enviará a un sistema de pago, el cual se encargará de gestionar el proceso de pago.

- Proporcionar información asociada a las transacciones (tipo y costo de viaje, nombre y dirección del cliente, operador y conductor del vehículo, etc.) al sistema de transacciones, el cual es responsable de almacenar y gestionar esta información.

Capítulo 5

Desarrollo de la comparación

5.1 Metodología PASSI

Para mostrar el desarrollo realizado con la metodología *PASSI* se comenzará primero con la presentación del modelado del sistema y luego se continuará con la aplicación de los frameworks de *Arnon Sturm* y *Pedro Cuesta*.

5.1.1 Modelado del sistema

En esta sección se presentan los productos obtenidos tras el desarrollo de las etapas y fases propuestas por la metodología *PASSI*.

5.1.1.1 Modelo de requerimientos del sistema

Como se mencionó en la sección 2.4.6, este modelo es desarrollado en la etapa de análisis y esta compuesto por las fases: *Descripción del Dominio, Identificación de Agentes, Identificación de Roles* y *Especificación de Tareas.*

En la fase *Descripción del Dominio* se representan los requerimientos del sistema a través de casos de uso. En la fase *Identificación de Agentes*, se agrupan los casos de uso de la fase anterior según su funcionalidad y se asignan a un agente, de esta manera es posible establecer los agentes que integrarán el sistema y las funcionalidades asociadas a cada uno de ellos.

La Figura 5.1 muestra el *Diagrama de Identificación de Agentes* resultante de la fase *Identificación de Agentes*.

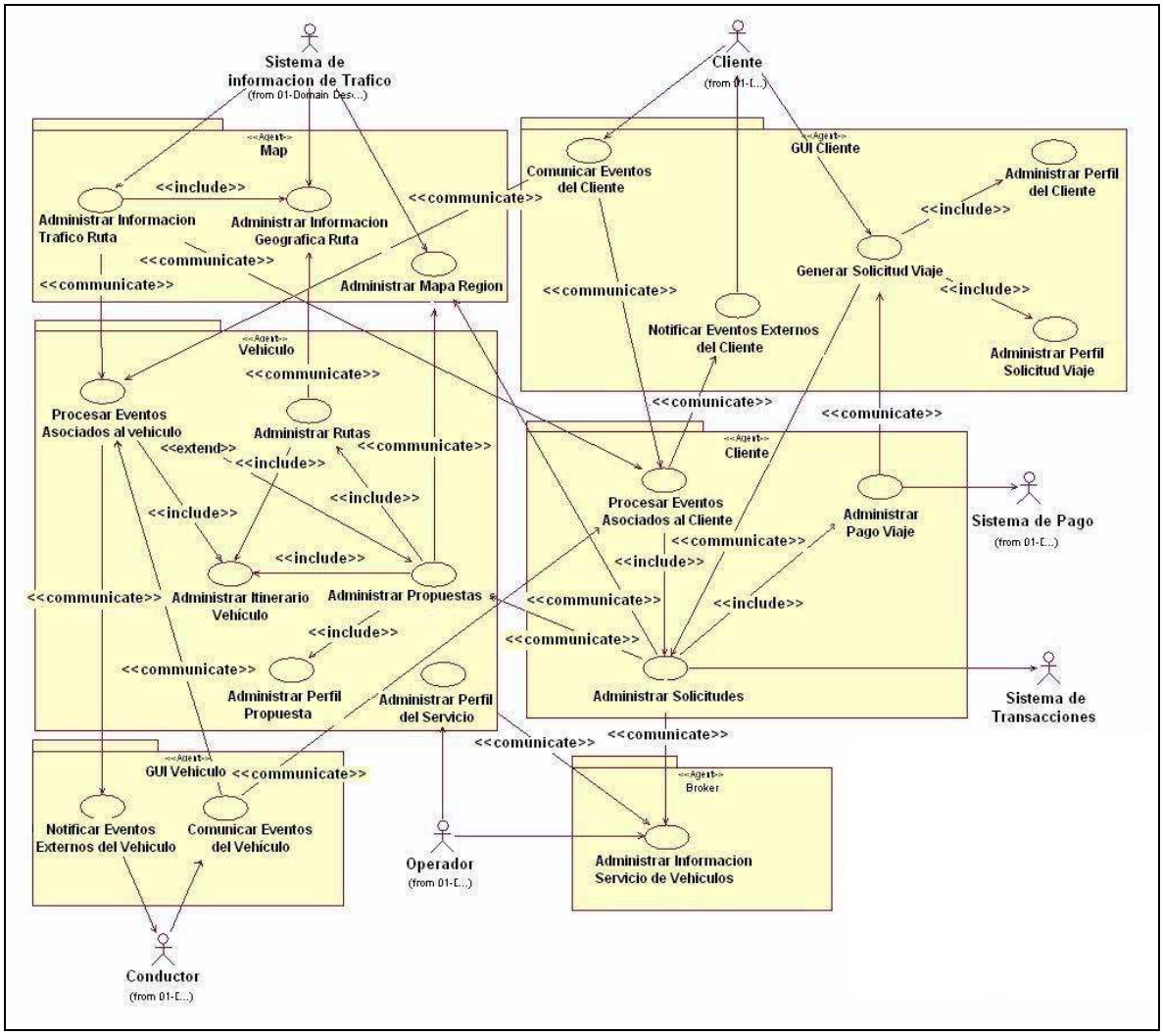

Figura 5.1 Diagrama identificación de agentes.

Como se aprecia en la Figura 5.1, los clientes individuales y sus requisitos son capturados por los agentes *GUI Cliente* y *Cliente*, los que proveen una completa comunicación e interoperabilidad entre el usuario real con el sistema de transporte. El agente *GUI Cliente* proporciona una interfaz de usuario personalizada que permite mantener al usuario informado sobre todo el proceso de solicitud de servicios y los eventos importantes que ocurran en relación a estos, mientras que el agente *Cliente* gestiona el proceso de solicitud del servicio, sus características y toma la decisión requerida. Por otra parte, cada vehículo real es representado por la dupla de agentes *GUI Vehículo* y *Vehículo*, los cuales proporcionan interoperabilidad entre el vehículo que representan y el sistema de transporte al cual pertenecen. El agente *GUI Vehículo* tiene el rol de interfaz, proporcionando al vehículo y su conductor un canal de comunicación con el resto del sistema de transporte, por el cual el conductor es capaz de comunicar a lo largo del viaje cualquier contingencia que pudiera aparecer. Por otro lado, el conductor recibe a través del agente *GUI Vehículo* cualquier información referente a cambios en la planificación original, tales como transportar un nuevo cliente, cancelación de clientes y cambios

relevantes en las condiciones de tráfico, entre otros. Además, el agente *GUI Vehículo* también es responsable de monitorear e informar acerca del estado y progreso del vehículo a lo largo del día. El agente *Vehículo* por su parte, esta a cargo de gestionar la ruta del vehículo y procesar cualquier nueva solicitud para transportar un nuevo cliente. Este agente es responsable de cualquier ajuste necesario en la ruta del vehículo o la programación de sus viajes motivados por alguna eventualidad o cambio. Ambos agentes dependen estrechamente el uno del otro de manera que el primero actúa como una especie de sensor de lo que ocurre en el ambiente y en el vehículo. Mientras el segundo razona basado en esa entrada y toma las decisiones de acuerdo a la planificación, comunicándola de vuelta al vehículo real pasando a través del agente *GUI Vehículo*. El agente *Map* modela la actual región geográfica bajo cobertura y tiene el papel de proveer al resto de los agentes cualquier información relacionada a esta. Para implementar estos propósitos, el agente *Map* provee la información sobre las distancias entre diversos lugares y que ruta deben seguir (ruta más corta). Para proporcionar esa información el agente del mapa modela el área cubierta como sistema de nodos que representan los lugares posibles de parada. Contiene internamente una matriz de la distancias con una representación gráfica de esos nodos. Además, el agente *Map* maneja otra matriz con las rutas mínimas (y por lo tanto las distancias) entre cada par de nodos. Las distancias se almacenan con dos valores: el tiempo de traslado de un nodo a otro y la distancia entre estos (kilómetros). Estas dos maneras de representar distancias son importantes porque reflejar diversos aspectos de caminos y de trayectorias. Para una trayectoria dada y bajo condiciones que cambian (por ejemplo congestión, accidente de tráfico) la medida de la distancia sigue siendo invariante mientras que la medida del tiempo tenderá a aumentar, pues llevará más tiempo llegar al destino a través de ese lugar. El papel principal del *Map* es proporcionar las distancias y las trayectorias más cortas entre los pares de puntos dados. Para llegar a esa información necesita los datos sobre las distancias reales entre cada par de nodos y su localización verdadera (dirección). El agente *Map* se puede conectar con un *Sistema de información tráfico* que pueda proveerle esta información. También, el agente *Map* puede recibir datos actualizados de un agente tráfico u otras fuentes externas de la información del tráfico. Si éste es el caso, el agente mapa necesitará proporcionar la funcionalidad para informar a los agentes correspondientes sobre esos cambios. Por último, el papel principal del agente *Broker* es saber qué servicios de transporte existen y sus características. Puede analizar las características del servicio por requerimiento del vehículo e informarle acerca de ellas. Además, registra/cancela las subscripciones de servicios de vehículos, permitiendo que los vehículos se incorporen o que salgan del sistema libremente.

Posteriormente, en la fase *Identificación de Roles* se identifican los roles que desempeñará cada uno de los agentes que componen la sociedad. Por último, en la fase *Especificación de Tareas* se especifican las tareas que deben ejecutar los agentes, de acuerdo al rol que desempeñen, para alcanzar sus objetivos.

5.1.1.2 Modelo de sociedad de agentes

El *Modelo de Sociedad de Agentes* es un modelo de interacciones sociales y dependencias entre los agentes involucrados en la solución. Desarrollar este modelo involucra las siguientes fases: *Identificación de Roles, Descripción de la Ontología del Dominio, Descripción de la Ontología de Comunicación, Descripción de Roles* y *Descripción de Protocolos.* En la fase *Identificación de Roles* (perteneciente también al modelo anteriormente descrito) se identifican las rutas de comunicación entre los agentes y el contenido de los mensajes intercambiados. Luego, en la fase *Descripción de la Ontología del Dominio* se define el conocimiento del entorno que la sociedad de agentes deben poseer, es decir, se establece un mismo vocabulario para la comunicación. Posteriormente, en la fase *Descripción de la Ontología de Comunicación* se establecen los conocimientos (ontologías del dominio), tipo de lenguaje de comunicación y protocolo para cada comunicación entre los agentes.

A partir de esto, en la fase *Descripción de Roles* se muestran las relaciones y dependencias entre los roles que desempeña cada agente (ver Figura 5.2). Por ejemplo, el agente *Cliente* como *Administrador De solicitudes* necesita consultar al agente *Broker* sobre los servicios que se encuentran disponibles y que encajan con el perfil de la solicitud, a través de la tarea *Recolectar Información del Servicio*. Por su parte, el agente *Broker* como *Administrador de la Información del Servicio de Vehículos* debe realizar la tarea *Matching Servicios Disponibles* para enviar el *Perfil del Servicio Disponible* en caso de que el matching sea favorable.

Por último, en la fase *Descripción de Protocolos* si no se utilizan protocolos estándar, como por ejemplo los protocolos *FIPA*, es necesario definir los protocolos de las interacciones existentes entre cada agente.

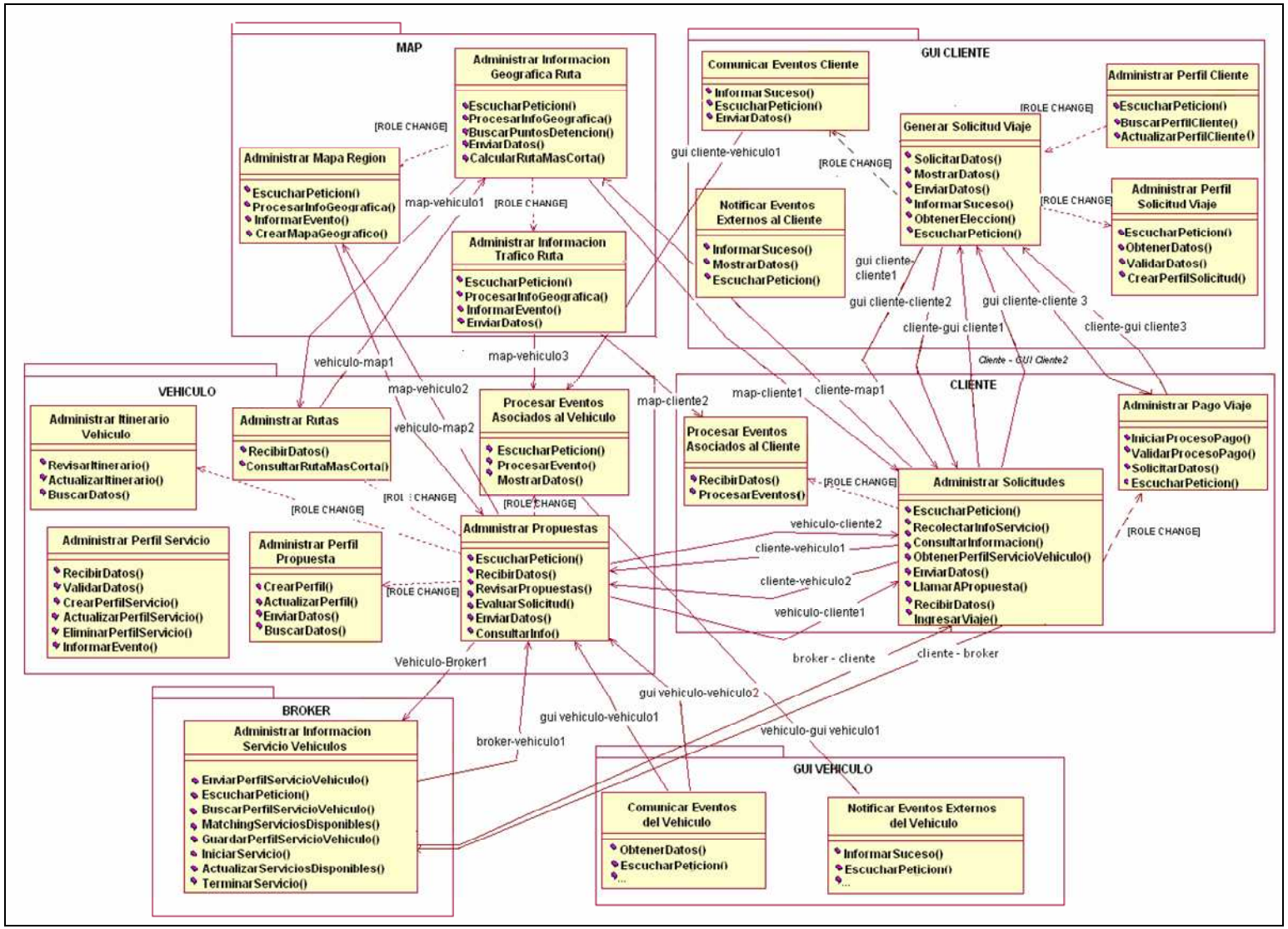

Figura 5.2 Diagrama de descripción de roles.

5.1.1.3 Modelo de implementación de agentes

El *Modelo de Implementación de Agentes* es un modelo típico de arquitectura en términos de clases y métodos. Este modelo está compuesto por las fases: *Definición de la Estructura del Agente, Definición de la Estructura del Sistema Multiagente, Descripción del Comportamiento del Agente* y *Descripción del Comportamiento del Sistema Multiagente.*

En la fase *Definición de la Estructura del Agente* se describe la estructura del agente y todas sus tareas. En la fase *Definición de la Estructura del Sistema Multiagente* (ver Figura 5.3) se representa la estructura del sistema multiagente a través de un diagrama de clases, el cual contiene clases y actores. Cada clase simboliza un agente del sistema.

Como se aprecia en la Figura 5.3, la clase correspondiente al agente *Broker* posee los atributos *eventos* y *servicios vehículos*; los cuales corresponden a las ontologías que el agente conoce y entiende; y por ejemplo, el método *iniciar servicio*, el cual realiza la tarea de registrar a todos los vehículos que estén en condiciones de comenzar con su labor.

Por último, en las fases *Descripción del Comportamiento del Agente* y *Descripción del Comportamiento del Sistema Multiagente*, se muestra el comportamiento de los agentes y del sistema respectivamente, ante la ocurrencia de un determinado evento.

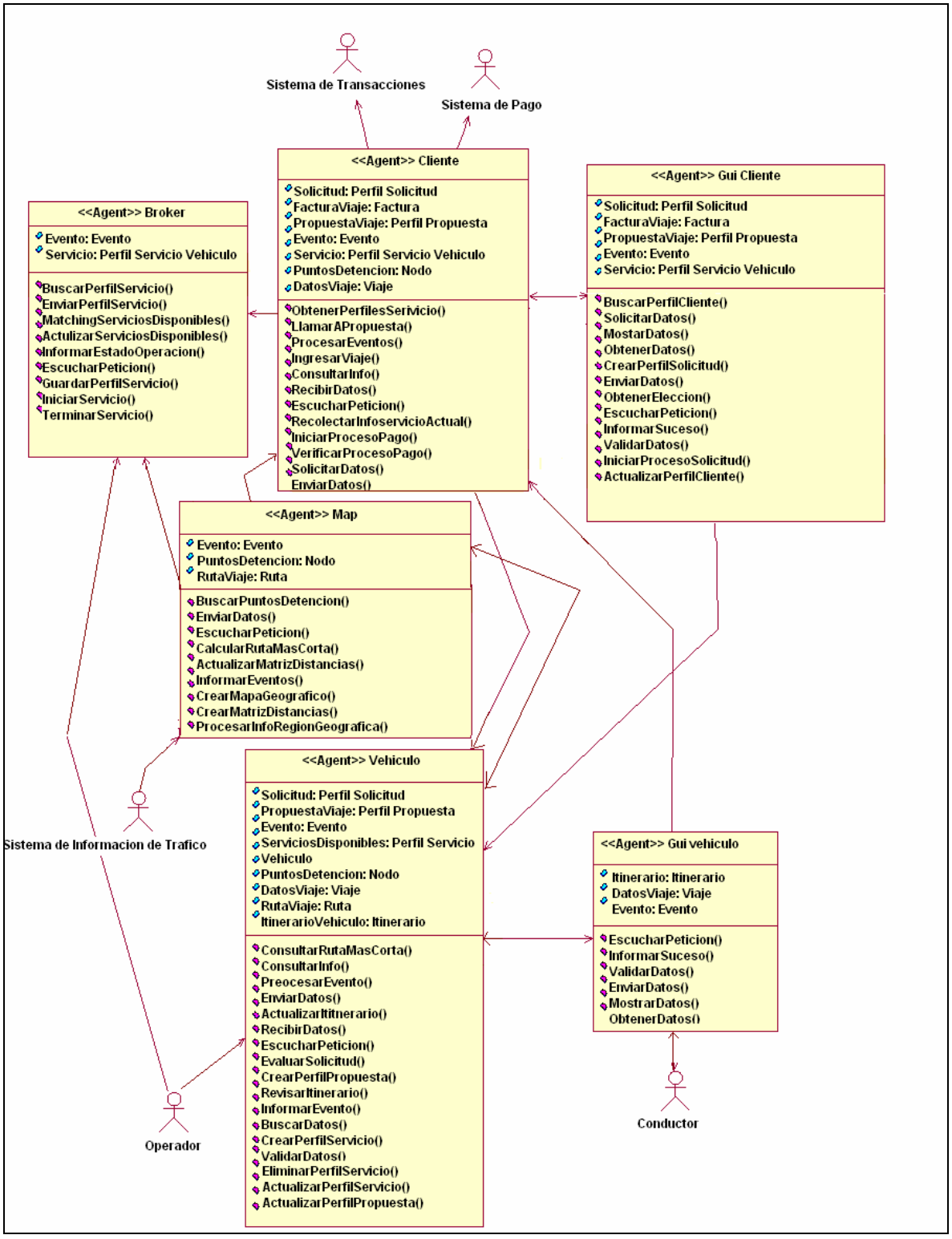

Figura 5.3 Diagrama definición de la estructura del sistema multiagente.

5.1.1.4 Modelo de código

El *Modelo de Código* es un modelo de la solución a nivel de código, el cual requiere de la fase *Generación de Código y Reutilización*. En esta etapa de la metodología el *PTK* genera automáticamente las estructuras (carcasas) de cada agente. Por motivos de alcance del proyecto, esta etapa no se especificará. Sin embargo será contemplada dentro de la evaluación.

5.1.1.5 Modelo de despliegue

El *Modelo de Despliegue* es un modelo de distribución de las partes del sistema, a través de las unidades de hardware de procesado.

Como se puede apreciar en la Figura 5.4, el sistema *DRT* es un sistema distribuido en donde por ejemplo, el agente *GUI Vehículo* puede habitar en un contenedor alojado en un dispositivo móvil utilizado por el conductor vehículo. El agente *GUI Cliente* puede habitar en un contenedor alojado en el computador personal del cliente; mientras que el resto de los agentes pueden estar alojados en el servidor de un centro de operaciones.

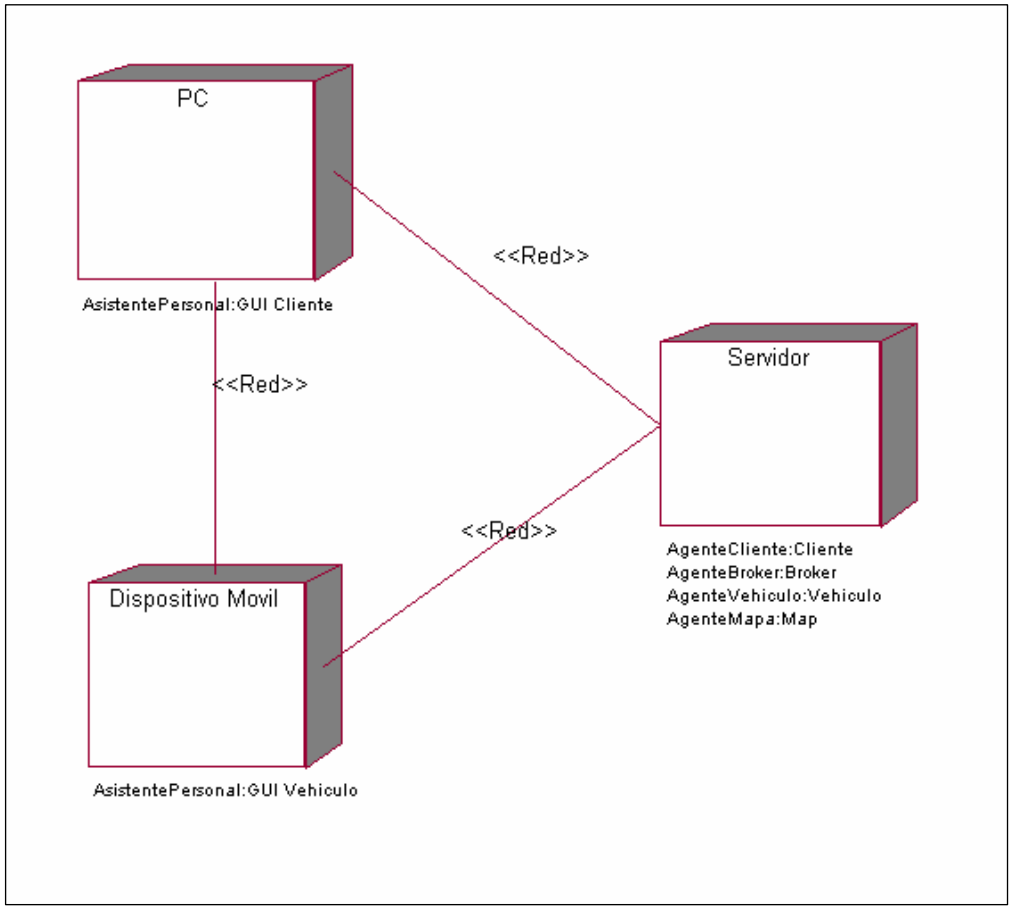

Figura 5.4 Diagrama de despliegue sistema DRT.

5.1.2 Aplicación de los frameworks de evaluación

En esta sección se presenta la aplicación de los frameworks de evaluación de *Arnon Sturm* y *Pedro Cuesta* a la metodología orientada a agentes *PASSI*.

5.1.2.1 Framework de Arnon Sturm

Este framework utiliza cuatro criterios para la evaluación, a saber: *Conceptos y Propiedades, Notación y Técnicas de Modelado, Proceso y Pragmática*; los cuales serán presentados de acuerdo a la definición propuesta por el autor (ver sección 3.2.2).

Conceptos y Propiedades

Tabla 5.1 Evaluación de los conceptos generales de PASSI según Arnon Sturm.

Concepto	Ranking	Descripción
Autonomía	7	Es expresada a través del concepto de rol. Un rol encapsula una funcionalidad determinada la cual es desempeñada por un agente, y por lo tanto, es responsable de ella. Esta funcionalidad es interna y no es afectada por el entorno. Los roles son identificados en la fase *Identificación de Roles* (*Modelo de Requerimientos del Sistema*), a través del uso de diagramas de secuencia (ver Figura A.6), y son descritos en la fase *Descripción de Roles* (*Modelo de Sociedad de Agentes*) (ver Figura 5.2).
Reactividad	7	Es expresada a través de la definición del comportamiento de un agente. En *PASSI* el comportamiento se representa a nivel de agente y del sistema multiagente. La fase *Descripción del Comportamiento de un Agente* (*Modelo de Implementación de un Agente*) desarrolla el comportamiento que un agente debe tener ante la ocurrencia de un evento, el cual está sujeto a restricciones/condiciones. La fase *Descripción del Comportamiento del Sistema Multiagente* (*Modelo de Implementación de un Agente*) permite mostrar todas las interacciones entre las principales clases de agentes y sus clases internas, representadas por tareas (tareas de cooperación entre agentes).
Pro Actividad	4	Es expresada a través de los conceptos de rol y tarea. Un agente desempeña uno o más roles dentro de la sociedad de agentes con la finalidad de alcanzar un objetivo, el cual se verá alcanzado concretamente a través de la ejecución de tareas. Sin embargo, la metodología no permite darle al agente la capacidad de plantearse nuevos objetivos y de realizar las tareas que éste estime conveniente.

Concepto	Ranking	Descripción
Habilidad Social	7	Es expresada a través de los conceptos de comunicación. En *PASSI*, la comunicación se da a través de la interacción entre agentes, referida a un protocolo de interacción y a una pieza de la ontología del dominio (conocimiento intercambiado durante la interacción). La metodología muestra este concepto en sus distintas fases. En la fase *Identificación de Roles* (*Modelo de Requerimientos del Sistema*) se identifican las rutas de comunicación existentes entre los agentes y los mensajes intercambiados entre ellos. En la fase *Descripción de la Ontología de* Comunicación (*Modelo de Sociedad de Agente*) se establecen las comunicaciones (conversaciones) entre agentes, definiendo para cada una de ellas: las tareas participantes, el lenguaje de comunicación, el protocolo de interacción y la ontología utilizada (ver Figuras A.21a y A.21b). En la fase *Descripción de Roles* (*Modelo de Sociedad de Agente*) se muestra la comunicación existente entre los roles que desempeñan los agentes.

Bloques de Construcción Básicos

Tabla 5.2 Evaluación de los bloques de construcción básicos de PASSI según Arnon Sturm.

Bloque	Ranking	Descripción
Agente	4	Un agente (agente *FIPA*) es una entidad de software capaz de acción en un entorno, que puede comunicarse directamente con otros agentes a través del uso de un lenguaje de comunicación, la cual es dirigida por un conjunto de tendencias (bajo la forma de objetivo individual o de una función de la satisfacción que intenta optimizar), que posee recursos propios, que es capaz de percibir su ambiente, que tiene solamente una representación parcial de este ambiente, que posee habilidades y puede ofrecer servicios, cuyo comportamiento tiende hacia la satisfacción de sus objetivos, tomando en cuenta sus habilidades y los recursos disponibles para él y dependiendo de su percepción, representación y comunicación. De esto se desprende que la metodología se dirige a la noción débil de agente (ranking promedio: 6). En cuanto a la noción fuerte de agente, la metodología sólo menciona aspectos de movilidad y racionalidad (ranking promedio: 2).
Creencia	1	Bloque no especificado en la metodología.
Deseo	1	Bloque no especificado en la metodología.
Intención	1	Bloque no especificado en la metodología.

Bloque	Ranking	Descripción
Mensaje	6	Un mensaje corresponde a un acto comunicativo, en el sentido que lo codifica para una transmisión confiable entre los agentes. Un mensaje dado puede representar múltiples actos comunicativos individuales. Los mensajes son descritos en la fase *Descripción del Comportamiento del Sistema Multiagente* perteneciente al *Modelo de Implementación de Agentes*. Es importante señalar que el *PTK* permite utilizar lenguajes de contenido estándar como el *FIPA ACL content lenguaje*.
Organización	1	Bloque no especificado en la metodología.
Protocolo	6	Es un patrón común de conversaciones usado para realizar una cierta tarea. Un protocolo de interacción es asignado y descrito (en el caso de no ser un protocolo FIPA) en el *Modelo de Sociedad de Agente*, fases *Descripción de la Ontología de Comunicación* y *Descripción de Protocolos* (ver Figuras A.21a y A.21b), y es reportado en el *Modelo de Implementación de Agentes*, fase *Descripción del Comportamiento del Sistema Multiagente*. La metodología se refiere como protocolo a patrones del diálogo entre los agentes, y a protocolo de red a los mecanismos de transporte, como por ejemplo *TCP/IP*.
Rol	6	Es una porción del comportamiento social de un agente que es caracterizado por un objetivo y/o proporciona una funcionalidad/servicio.
Servicio	6	La metodología define a un servicio como un bloque singular, coherente de actividad el cuál será contratado por un agente. Un conjunto de servicios se asocia a cada rol de un agente. En la fase *Descripción de Roles* (*Modelo de Sociedad de Agentes*) se representan las dependencias entre servicios, recursos y cambios de roles.
Sociedad	5	Una sociedad es una colección de agentes y organizaciones que colaboran para promover sus objetivos individuales. La metodología dirige este bloque a través del *Modelo de Sociedad de Agentes*, el cual está compuesto por las fases: *Descripción de la Ontología de Dominio*, *Descripción de la Ontología de Comunicación*, *Descripción de Roles* y *Descripción de Protocolos*. Es importante destacar que la sociedad de agentes propuesta por la metodología no está dirigida por normas o reglas.
Tarea	6	Una tarea especifica las acciones que los agentes deben realizar para alcanzar sus objetivos. Son identificadas en el *Modelo de Requerimientos del Sistema* (fase *Especificación de Tareas*) y en el *Modelo de Implementación de Agentes* (fase *Descripción de la Estructura del Sistema Multiagente*), y posteriormente utilizadas en el *Modelo de Sociedad de Agente* (fases *Descripción de la Ontología de Comunicación* y *Descripción de Roles*) y en el *Modelo de Implementación de Agentes* (fase *Descripción del Comportamiento del Sistema Multiagente*).

Notación y Técnicas de Modelado

Tabla 5.3 Evaluación de la notación y técnicas de modelado de PASSI según *Arnon Sturm*.

Propiedades	Ranking	Descripción
Accesibilidad	6	El modelado es básicamente fácil de entender; ya que utiliza *UML* como lenguaje de modelado. Sin embargo, la metodología requiere el conocimiento de conceptos relacionados al área de agente para su aplicación.
Capacidad de Análisis	7	Esta propiedad la realiza automáticamente el *PTK*.
Administración de la Complejidad	7	Esta propiedad se puede apreciar en las múltiples vistas que propone la metodología. A medida que se va avanzando en el desarrollo, el nivel de detalle crece, pero siempre se mantiene una vista global del sistema. Por ejemplo, en la fase *Identificación de Agentes* (*Modelo de Requerimientos del Sistema*) se identifican los agentes que compondrán el sistema y las funcionalidades que deberán alcanzar (ver Figura 5.1). Luego, en la fase *Descripción de Roles* (*Modelo de Sociedad de Agentes*) dichas funcionalidades se ven representadas como roles. La fase muestra cuales son los roles desempeñados por cada agente y como es la comunicación entre estos, además de las tareas asociadas a cada uno. Las tareas ejecutadas por cada agente son detalladas en la fase *Especificación de Tareas* perteneciente al *Modelo de Requerimientos del Sistema* (ver Figura A.17).
Ejecutabilidad y Testeabilidad	2	Si bien el *PTK* no incluye una funcionalidad para poder testear una cierta porción del sistema, la metodología sugiere a grandes rasgos la inclusión de pruebas unitarias y del sistema.
Expresividad y Aplicabilidad a Múltiples Dominios	5	En las diversas fases de la metodología se pueden apreciar claramente los conceptos relacionados con: ontologías (ver Figuras A.20a, A.20b, A.21a y A.21b), flujo de datos dentro del sistema (ver Figura A.6), agentes (ver Figura 5.1 y Figura A.25) y arquitectura del sistema (ver Figura 5.3). Sin embargo, la interacción con sistemas externos, y la definición de interfaces de usuario no son especificadas.
Modularidad	4	La metodología define que el proceso de desarrollo es iterativo. Sin embargo, el *PTK* presenta problemas de consistencia cuando se realizan cambios en los modelos de etapas previas, lo cual obliga a los desarrolladores a llevar un control de versiones exhaustivo.
Exactitud	7	Cada modelo es especificado de manera no ambigua.

Proceso

Tabla 5.4 Evaluación del proceso de desarrollo de PASSI según Arnon Sturm.

Etapas	Ranking	Descripción
Reunión de Requerimientos	1	En *PASSI* se asume que los requerimientos del sistema ya han sido detallados y se comienza el modelado a partir de estos.
Análisis	5	La etapa de análisis está definida en el *Modelo de Requerimientos del Sistema*. Las actividades realizadas en esta etapa son: *Descripción del Dominio*, *Identificación de Agentes*, *Identificación de Roles* y *Especificación de Tareas*. Los entregables generados en esta etapa corresponden a los diagramas de las actividades anteriormente nombradas, a saber: diagrama de descripción del dominio, diagrama de identificación de agentes, diagrama de identificación de roles y diagrama de especificación de tareas. La validación de esta etapa la realiza completamente el *PTK*. La metodología no proporciona guías de aseguramiento de la calidad ni administración de proyectos en esta etapa.
Diseño	5	La etapa de diseño está definida en el *Modelo de Sociedad de Agentes*. Las actividades realizadas en esta etapa son: *Descripción de la Ontología de Dominio*, *Descripción de la Ontología de Comunicación*, *Descripción de Roles y Descripción de Protocolos.* Los entregables generados en esta etapa corresponden a los diagramas de las actividades anteriormente nombradas, a saber: diagrama de descripción de la ontología de dominio, diagrama de descripción de la ontología de comunicación, diagrama de descripción de roles y diagrama de descripción de protocolo de interacción (en caso de que no sean protocolos *FIPA*). La validación de esta etapa la realiza completamente el *PTK*. La metodología no proporciona guías de aseguramiento de la calidad ni administración de proyectos en esta etapa.
Implementación	5	La etapa de implementación está definida en el *Modelo de Implementación de Agentes* y *Modelo de Código*. Las actividades realizadas en esta etapa son: *Definición de la Estructura del MAS*, *Descripción del Comportamiento del MAS*, *Definición de la Estructura del Agente* y *Descripción del Comportamiento del Agente*. Los entregables generados en esta etapa corresponden a los diagramas de las actividades anteriormente nombradas, a saber: diagrama de definición de la estructura del *MAS*, diagrama de descripción del comportamiento del *MAS*, diagrama de definición de la estructura del agente y diagrama de descripción del comportamiento del agente. La validación de esta etapa la realiza completamente el *PTK*. La metodología no proporciona guías de aseguramiento de la calidad ni administración de proyectos en esta etapa.
Prueba	2	La metodología sólo sugiere a grandes rasgos la inclusión de pruebas unitarias y del sistema.

Pragmática

Tabla 5.5 Evaluación de la pragmática de PASSI según Arnon Sturm.

Aspectos	Ranking	Descripción
Recursos	5	Existe una gran cantidad de papers donde se explica cada una de las etapas de la metodología, pero siempre de manera resumida, aunque no por eso dejan de ser muy claros y útiles. Además, como se mencionó anteriormente, la metodología cuenta con el *PTK* como herramienta case para el modelado y la generación automática de código.
Habilidad Requerida	5	Se requiere tener conocimientos básicos del área de agentes y sistemas multiagente, además de saber *UML*.
Aplicación de Lenguajes	6	Es independiente del lenguaje de programación. Sin embargo, se infiere que es más adecuado utilizar lenguajes que ayuden de mejor manera a codificar sistemas distribuidos (por ejemplo, lenguajes orientados a objetos).
Dominio de Aplicación	6	*PASSI* se presenta como metodología de propósito general. En la literatura podemos encontrar ejemplos de desarrollos en diversos dominios de aplicación (robótica, biología celular, sistemas de información tradicionales, entre otros). Además, se ha podido comprobar que la metodología es aplicable al desarrollo del sistema *DRT*.
Escalabilidad	6	A través del *PTK* se pueden utilizar funcionalidades provistas por *Rational Rose* como por ejemplo, la exportación/ importación de proyectos (en formato *JAR*).

5.1.2.2 Framework de Pedro Cuesta

Este framework utiliza cinco criterios para la evaluación, a saber: *Proceso de Desarrollo, Vistas del Modelo, Agente, Características Adicionales de Modelado y Documentación;* los cuales serán presentados de acuerdo a la definición propuesta por el autor (ver sección 3.2.3).

Proceso de Desarrollo

Tabla 5.6 Evaluación del proceso de desarrollo de PASSI según Pedro Cuesta.

Aspectos a Considerar	Explicación	Evaluación
Dominio de Aplicación	La metodología es de propósito general. Se construyen sistemas heterogéneos.	Dominio Independiente
Áreas de Aplicación	Las áreas de aplicación van desde sistemas de información (sistemas de transporte, biblioteca, impresión, etc.) hasta sistemas biológicos.	----------
	Robótica, Biología Celular y Sistemas de Información Tradicionales (Biblioteca).	Documentada
Sistemas Abiertos	La metodología no está orientada al desarrollo de sistemas abiertos.	No
Tipo de Ciclo de Vida	A pesar que se indica que es iterativo, existen algunos problemas al momento de realizar iteraciones. Esto es debido a que la metodología se apoya el proceso de desarrollo a través de una herramienta *CASE* (*PTK*), la cual no permite realizar cambios a los modelos una vez que se ha pasado a una nueva fase.	**Iterativo**
Etapa 1 Actividades de	**Análisis** Requerimientos Análisis Diseño Implementación Prueba Implicación del Usuario	**Bien definida** Parcialmente Enfocado Enfocado Parcialmente Enfocado No Enfocado No Enfocado Desconocido
Etapa 2 Actividades de	**Diseño** Requerimientos Análisis Diseño Implementación	**Bien definida** No Enfocado Parcialmente Enfocado Enfocado No Enfocado

Aspectos a Considerar	Explicación	Evaluación
Etapa 2	**Diseño (continuación)**	**Bien definida**
Actividades de	Prueba	No Enfocado
	Implicación del Usuario	Desconocido
Etapa 3	**Implementación**	**Bien Definida**
Actividades de	Requerimientos	No Enfocado
	Análisis	No Enfocado
	Diseño	No Enfocado
	Implementación	Enfocado
	Prueba	Parcialmente Enfocado
	Implicación del Usuario	Desconocido
Etapa 4	**Prueba**	**Parcialmente Definida**
Actividades de	Requerimientos	No Enfocado
	Análisis	No Enfocado
	Diseño	No Enfocado
	Implementación	Parcialmente Enfocado
	Prueba	Parcialmente Enfocado
	Implicación del Usuario	Desconocido
Herramientas de Soporte	**PASSI Tool Kit (*PTK*)**	Apoya todas las etapas del proceso de desarrollo propuesto por la metodología.
Apoya Actividades de	Requerimientos	Parcialmente Apoyado
	Análisis	Sí
	Diseño	Sí
	Implementación: Generación de Código	Sí
	Orientado al desarrollo de una herramienta	No[1]
	Prueba	Desconocido

[1] La metodología PASSI, al igual que PASSI Tool Kit, es independiente de cualquier arquitectura de agente, lenguaje de programación o framework de comunicación.

Vistas del Modelo

Tabla 5.7 Evaluación de las vistas del modelo de PASSI según Pedro Cuesta.

Conceptos y Representaciones para:	Concepto	Diagrama para cada Concepto
Proceso de Desarrollo	Agente	Identificación de Agentes, Identificación de Roles, Especificación de Tareas, Descripción Ontología de Comunicación, Descripción de Roles, Descripción Estructura del Sistema Multiagente, Descripción Comportamiento del Sistema Multiagente.
	Rol	Identificación de Roles, Descripción Ontología de Comunicación, Descripción de Roles.
	Tarea	Especificación de Tareas, Descripción Ontología de Comunicación, Descripción de Roles, Descripción Estructura del Sistema Multiagente, Descripción Comportamiento del Sistema Multiagente.
	Requerimiento	Descripción del Dominio, Identificación de Agentes.
	Objetivo	No explícitamente diseñado.
	Escenario	Identificación de Roles.
	Comunicación	Identificación de Roles, Descripción Ontología de Comunicación, Descripción de Roles.
	Mensaje	Descripción Comportamiento del Sistema Multiagente.
	Protocolo de Interacción de Agente	Descripción Ontología de Comunicación, Descripción de Protocolos, Descripción Comportamiento del Sistema Multiagente.
	Performative	Descripción de Protocolos, Descripción Comportamiento del Sistema Multiagente.
	Ontología (concepto, acción, y predicado)	Descripción Ontología del Dominio, Descripción Ontología de Comunicación, Descripción Comportamiento del Sistema Multiagente.
	Agente Plataforma FIPA	Descripción Estructura Agente, Descripción Estructura del Sistema Multiagente, Descripción Comportamiento del Agente, Reutilización de Código, Producción de Código, Configuración del Despliegue.
	Tarea Plataforma FIPA, Servicio	Descripción de Roles, Descripción Estructura Agente y del Sistema Multiagente, Descripción Comportamiento del Agente, Reutilización de Código, Producción Código.

Conceptos y Representaciones para:	Concepto	Diagrama para cada Concepto
Análisis	Requerimiento	Descripción del Dominio, Identificación de Agentes.
	Objetivo	No explícitamente diseñado.
	Caso de Uso	Descripción del Dominio.
	Agente	Identificación de Agentes, Identificación de Roles, Especificación de Tareas.
	Rol, Comunicación, Escenario	Identificación de Roles.
	Tarea	Especificación de Tareas.
Diseño	Ontología (concepto, acción, y predicado)	Descripción Ontología del Dominio, Descripción Ontología de Comunicación.
	Comunicación, Agente, Rol, Tarea	Descripción Ontología de Comunicación, Descripción de Roles.
	Protocolo de Interacción	Descripción Ontología de Comunicación, Descripción de Protocolos.
	Performative	Descripción de Protocolos.
	Servicio, Recurso	Descripción de Roles.
Implementación	Mensaje, Protocolo de Interacción, Performative, Ontología (concepto, acción y predicado).	Descripción Comportamiento del Sistema Multiagente.
	Agente	Descripción Estructura del Sistema Multiagente, Descripción Comportamiento del Sistema Multiagente.
	Agente Plataforma FIPA	Descripción Estructura Agente, Descripción Comportamiento del Agente, Descripción Comportamiento del Sistema Multiagente, Reutilización de Código, Producción de Código, Configuración del Despliegue.
	Tarea Plataforma IPA	Descripción Estructura Agente, Descripción Comportamiento del Agente, Descripción Comportamiento del Sistema Multiagente, Reutilización de Código, Producción de Código.

Conceptos y Representaciones para:	Concepto	Diagrama para cada Concepto
Prueba	----------	No explícitamente definido.
Interacción Humana	----------	No explícitamente definido.
Relación entre Modelos	Todos los modelos están relacionados. El *PTK* verifica su consistencia automáticamente.	Bien definida
Entregables	Todos los diagramas previamente definidos son posibles entregables, pero no se menciona si se deben complementar con prototipos o con descripciones escritas en lenguaje formal.	No explícitamente definido.

Agente

Tabla 5.8 Evaluación del concepto de agentes en PASSI según Pedro Cuesta.

Concepto	Definición	Restricciones en la Arquitectura de Agentes o Tipo
Agente	Un agente (agente *FIPA*) es una entidad de software: la cual es capaz de acción en un entorno, la cual puede comunicarse directamente con otros agentes típicamente usando un lenguaje de comunicación, la cual es dirigida por un conjunto de tendencias (bajo la forma de objetivo individual o de una función de satisfacción/supervivencia que intenta optimizar), que posee recursos propios, que es capaz de percibir su ambiente (pero a un grado limitado), que tiene solamente una representación parcial de este ambiente (y quizás ninguna), que posee habilidades y puede ofrecer servicios, que puede reproducirse a sí misma- cuyo comportamiento tiende hacia la satisfacción de sus objetivos, tomando en cuenta los recursos y las habilidades disponible para él y dependiendo de su percepción, representación y comunicación que recibe.	Es posible diseñar una arquitectura propia o utilizar arquitecturas predefinidas (como por ejemplo BDI). Asimismo, un diseñador puede utilizar componentes predefinidos o desarrollarlos. Los componentes consisten en un conjunto de atributos, métodos, y, si es complejo, puede tener una sub arquitectura.
Atributos de Agente	Autonomía Habilidad Social Reactividad Pro actividad	Sí Sí Sí Sí

Concepto	Definición	Restricciones en la Arquitectura de Agentes o Tipo
Atributos de Agente	Inteligencia ▪ Veracidad ▪ Benevolencia ▪ Racionalidad	Desconocido Desconocido Sí
Tipos de Comunicación	No se establece explícitamente la comunicación entre el MAS desarrollado y otros sistemas, ni tampoco con los usuarios.	Agente-Agente
Protocolos de Comunicación	El diseñador puede definir su propio protocolo de comunicación o utilizar uno ya establecido (como por ejemplo el protocolo FIPA), pero siempre la decisión es del usuario.	Definido por el usuario
Cooperación	Expresada a través de la comunicación (relaciones) entre agentes.	Sí
Organización de Agentes	----------	No

Características Adicionales de Modelado

Tabla 5.9 Evaluación de las consideraciones adicionales de modelado en PASSI según Pedro Cuesta.

Características	Explicación	Evaluación
Aspectos Ontológicos	Considerados ampliamente en el modelo de sociedad de agentes (descripción de la ontología de dominio y descripción de la ontología de comunicación). Además, el *PTK* permite exportar ontologías a RDF.	Sí
Movilidad	Presente en el modelo de despliegue, pero considerada de manera muy ligera.	Sí

Documentación

Tabla 5.10 Evaluación de la documentación de PASSI según Pedro Cuesta.

Aspecto	Explicación	Evaluación
Documentación Disponible	Papers Libros Tutoriales	Buena Pobre Suficiente
Casos de Estudio Presentados	Sistema de información biológico [34]. Sistema de información para una biblioteca [35]. Desarrollo de aplicaciones robóticas [36].	Completo Completo Parcial

5.2 Metodología MASE

Para mostrar el desarrollo realizado con la metodología *MASE* se comenzará primero con la presentación del modelado del sistema y luego se continuará con la aplicación de los frameworks de *Arnon Sturm* y *Pedro Cuesta*.

5.2.1 Modelado del sistema

En esta sección se presentan los productos obtenidos tras el desarrollo de las etapas y fases propuestas por la metodología *MASE*.

5.2.1.1 Capturar objetivos

Esta fase toma lo más importante de la especificación inicial del sistema y la transforma en un conjunto estructurado de objetivos del sistema. Los objetivos son utilizados para encapsular los requerimientos porque contienen lo que el sistema trata de alcanzar.

Las Figuras 5.5a, 5.5b y 5.5c muestran los objetivos que satisfacen los requerimientos de transporte gestionados por el sistema *DRT*. La Figura 5.5a muestra los objetivos del sistema que gestionan información relacionada con los clientes y las solicitudes, la Figura 5.5b muestra el objetivo del sistema que gestiona información relacionada con los vehículos, y por último, la Figura 5.5c muestra el objetivo del sistema que gestiona información relacionada con las rutas.

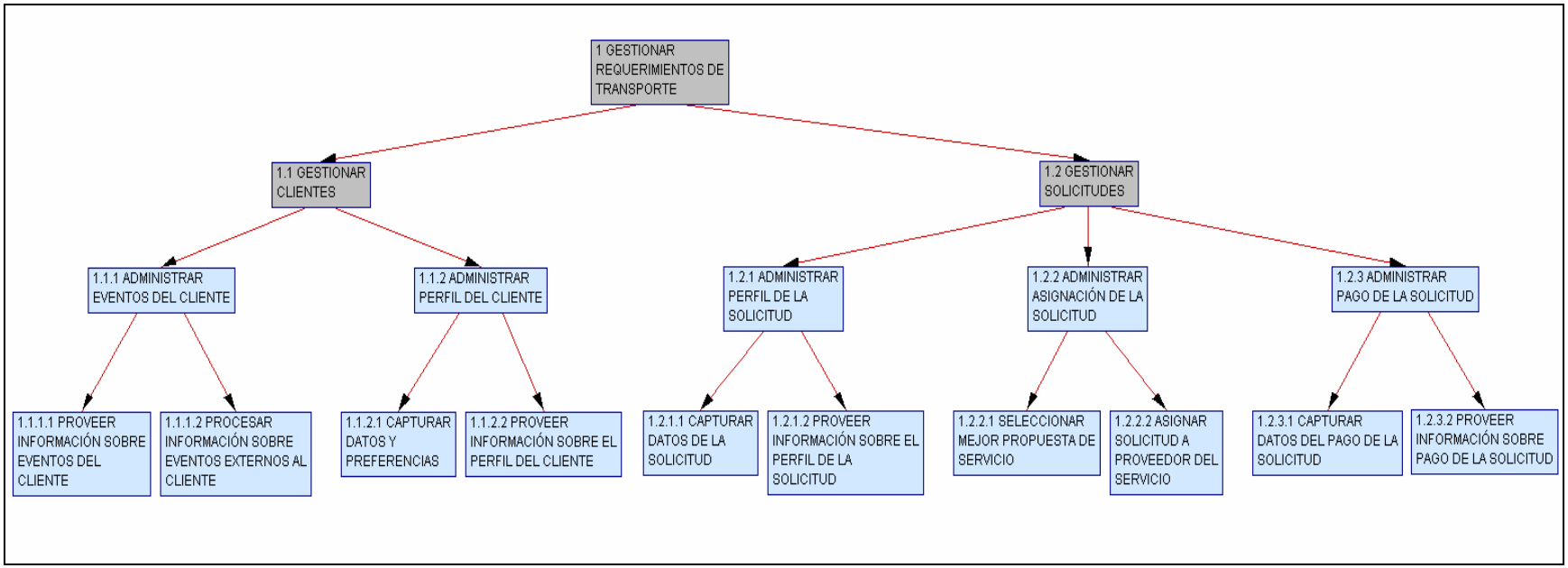

Figura 5.5a Objetivos 1.1 y 1.2 del diagrama de jerarquía de objetivos.

La Figura 5.5a muestra los objetivos que gestionan información relacionada con los clientes (objetivo *1.1 Gestionar Cliente*) y las solicitudes de servicio de transporte (objetivo *1.2 Gestionar Solicitudes*). Ambos objetivos son particionados, esto quiere decir que no serán implementados como roles en las fases posteriores, pero sí lo serán sus sub-objetivos.

El objetivo *Gestionar Cliente* debe ser capaz de mantener informados a los clientes sobre los eventos que ocurran durante el proceso de petición de solicitudes (sub-objetivo *Administrar Eventos del Cliente*), y de administrar toda la información referente a sus datos personales y preferencias (sub-objetivo *Administrar Perfil del Cliente*). Por otra parte, el objetivo *Gestionar Solicitudes* debe ser capaz de proveer y mantener información referente a la solicitud del servicio de transporte (sub-objetivo *Administrar Perfil de la Solicitud*), de administrar el proceso de selección y asignación del proveedor del servicio de transporte (sub-objetivo *Administrar Asignación de la Solicitud*), y de proveer y mantener información referente al pago de la solicitud (sub-objetivo *Administrar Pago Solicitud*).

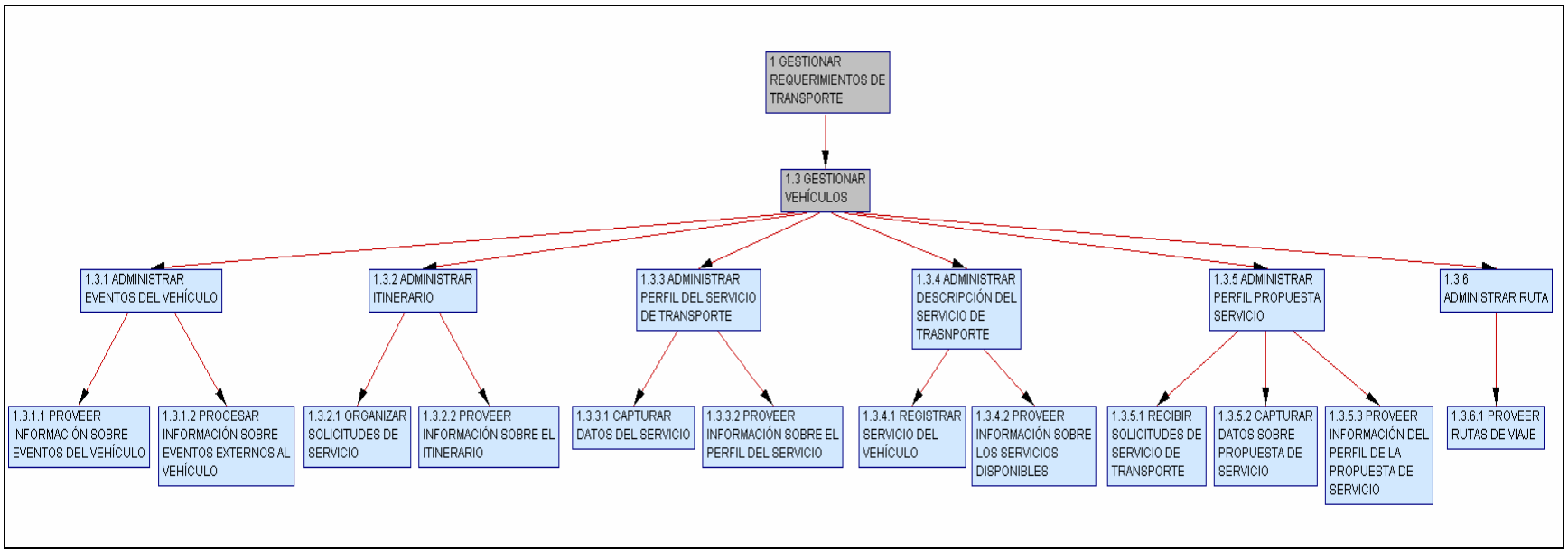

Figura 5.5b Objetivo 1.3 del diagrama de jerarquía de objetivos.

La Figura 5.5b muestra el objetivo encargado de gestionar información relacionada con los vehículos (objetivo *1.3 Gestionar Vehículos*). Al igual que los objetivos anteriores, este objetivo es particionado, lo cual indica que sólo sus sub-objetivos serán implementados en las fases posteriores.

El objetivo *Gestionar Vehículos* debe ser capaz de mantener informados a los conductores de los vehículos sobre los eventos que ocurran durante el proceso de solicitud del servicio y/o el viaje (sub-objetivo *Administrar Eventos del Vehículo*), de proveer y mantener información sobre el itinerario del vehículo que presta el servicio (sub-objetivo *Administrar Itinerario*), de proveer y mantener información referente al perfil del servicio de transporte (sub-objetivo *Administrar Perfil del Servicio de Transporte*), de registrar y proveer información sobre la descripción del servicio (sub-objetivo *Administrar Descripción del Servicio de Transporte*), de recibir, enviar y mantener información referente al perfil de la propuesta del servicio (sub-objetivo *Administrar Perfil Propuesta Servicio*), y de proveer información referente a la ruta (sub-objetivo *Administrar Ruta*).

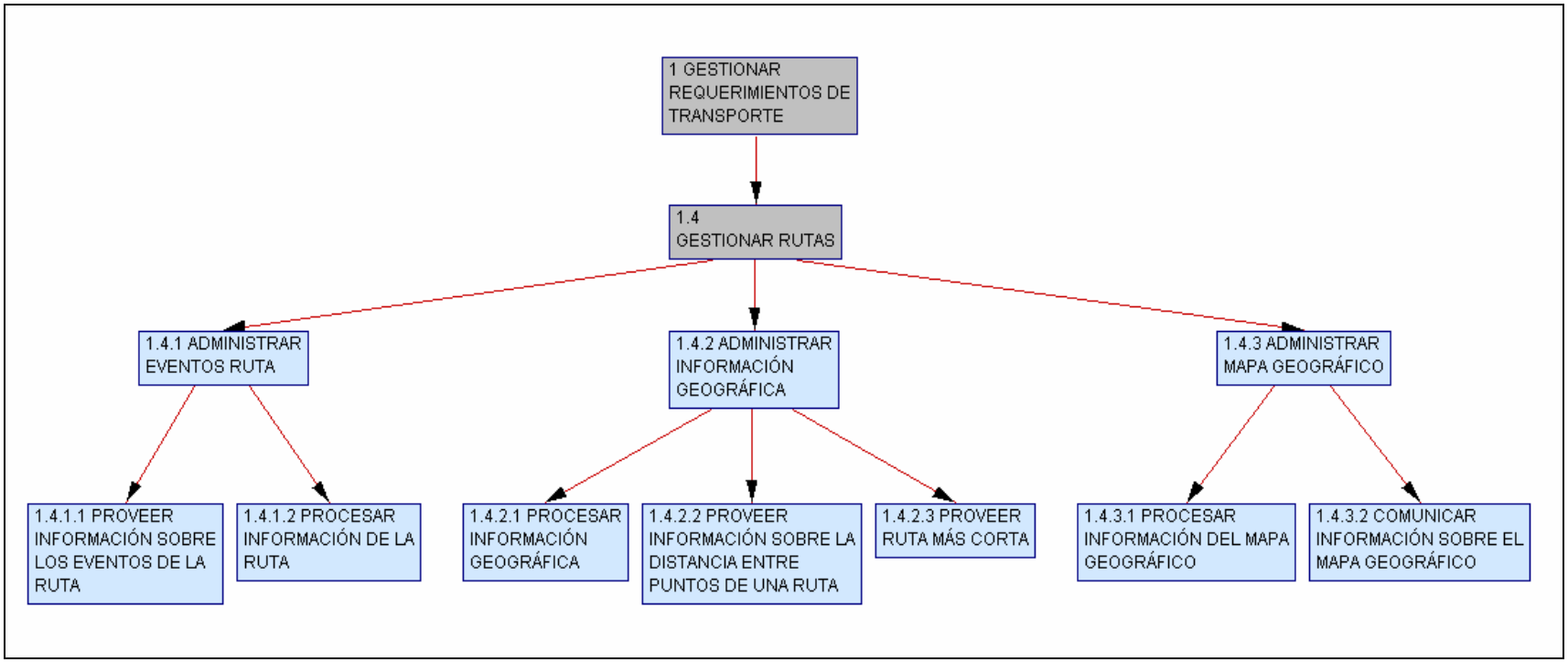

Figura 5.5c Objetivo 1.4 del diagrama de jerarquía de objetivos.

La Figura 5.5c muestra el objetivo encargado de gestionar información relacionada con las rutas (objetivo *1.4 Gestionar Rutas*). Al igual que los objetivos anteriores, este objetivo es particionado, lo cual indica que sólo sus sub-objetivos serán implementados en las fases posteriores.

El objetivo *Gestionar Rutas* debe ser capaz de proveer y procesar información referente a los eventos que afectan a una ruta (sub objetivo *Administrar Eventos Ruta*), de procesar información geográfica sobre la ruta, de proveer información sobre las distancias entre puntos de una ruta y proveer la ruta más corta entre dos puntos (sub-objetivo *Administrar Información Geográfica*), y por último, procesar y comunicar información sobre el mapa geográfico (sub-objetivo *Administrar Mapa Geográfico*).

5.2.1.2 Transformar objetivos a roles

En esta fase los objetivos definidos en la fase anterior son transformados en roles y sus tareas asociadas. Cada uno de los objetivos están asociados a un rol y cada uno de los roles es ejecutado o representado por una clase de agentes.

En la Figura 5.6 se puede apreciar que para alcanzar el sub-objetivo *Administrar Eventos Clientes*, se crea el rol *Administrador de Eventos Cliente*, el cual debe realizar las tareas de *Comunicar Evento Cliente*, *Notificar Evento Externo Cliente* y *Procesar Eventos*.

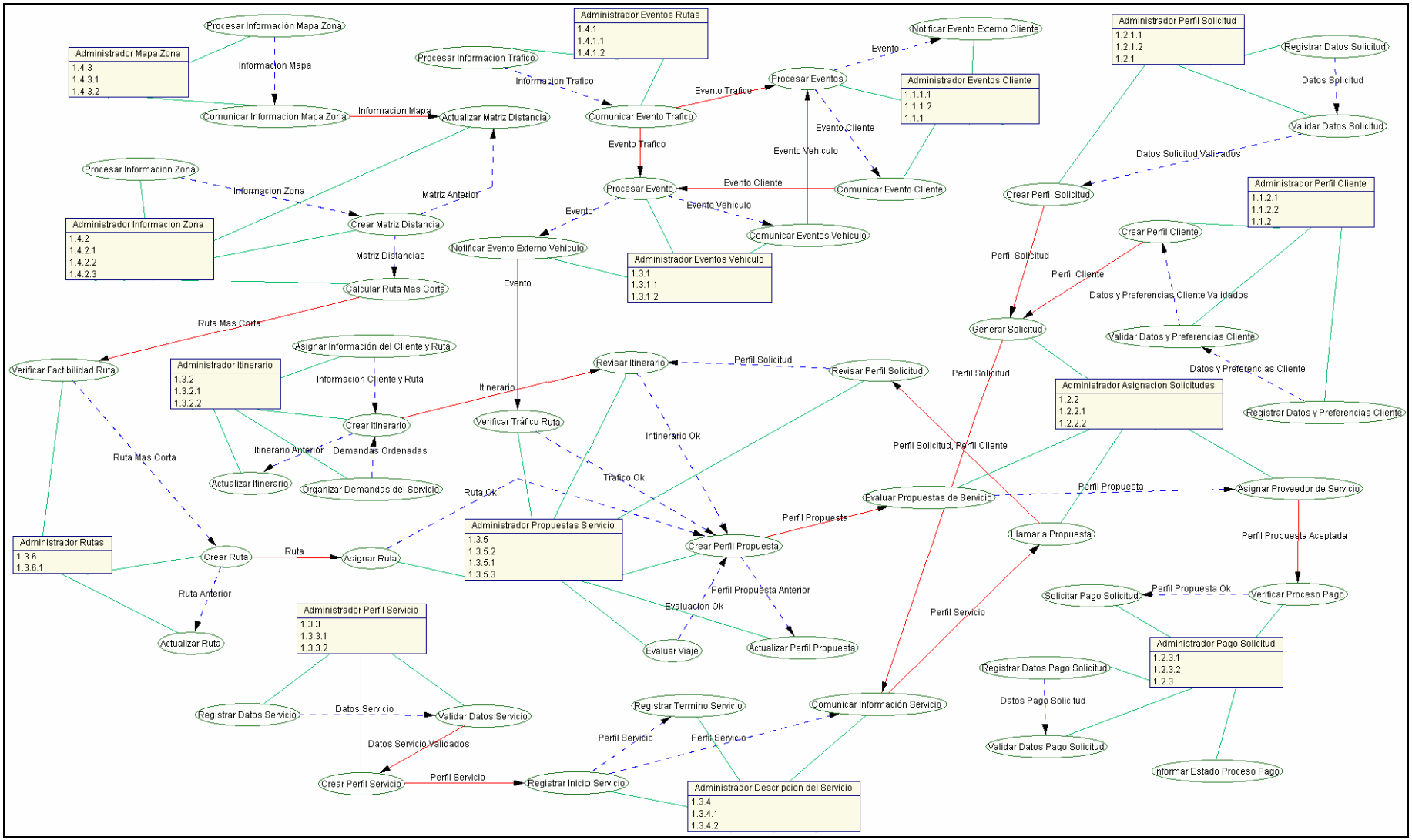

Figura 5.6 Diagrama de roles.

5.2.1.3 Crear clases de agente

En esta fase se crean las clases de los agentes pertenecientes a la solución. De esta fase se obtiene el *Diagrama de Clases de Agentes* (ver Figura 5.7) que es muy similar al *Diagrama de Objetos*, aunque las relaciones en el primero se interpretan como conversaciones.

Como se muestra en la Figura 5.7, el agente *GUI Cliente* mantiene diversas conversaciones con el agente *Cliente*. Primero, existe una conversación que representa la petición de solicitud de transporte de parte del cliente (conversación *solicitud de viaje*). Una vez que ocurre esto, el agente *Cliente* conversa con el agente *Broker* para pedirle el listado de servicios disponibles que se ajustan a la solicitud del cliente. Luego de que el *Broker* envía la lista de servicios disponibles que cumplen con el perfil de la solicitud, el agente *Cliente* comienza el proceso de negociación con el agente *Vehículo*. El agente *Cliente* informa al agente *GUI Cliente*, para que éste informe al cliente todas las propuestas de servicio. Una vez que el cliente le comunica su decisión al *GUI Cliente*, éste le informa al *Cliente* el cual cierra la negociación con el respectivo agente *Vehículo*. Por último, el *Cliente* le envía la confirmación de viaje al agente *GUI Cliente*.

Por otra parte, es posible ver también en el diagrama los roles que desempeñan cada uno de los agentes que componen el sistema. Por ejemplo, el agente *Cliente* cumple los roles de *Administrador Asignación Solicitud* y *Administrador Pago Solicitud*.

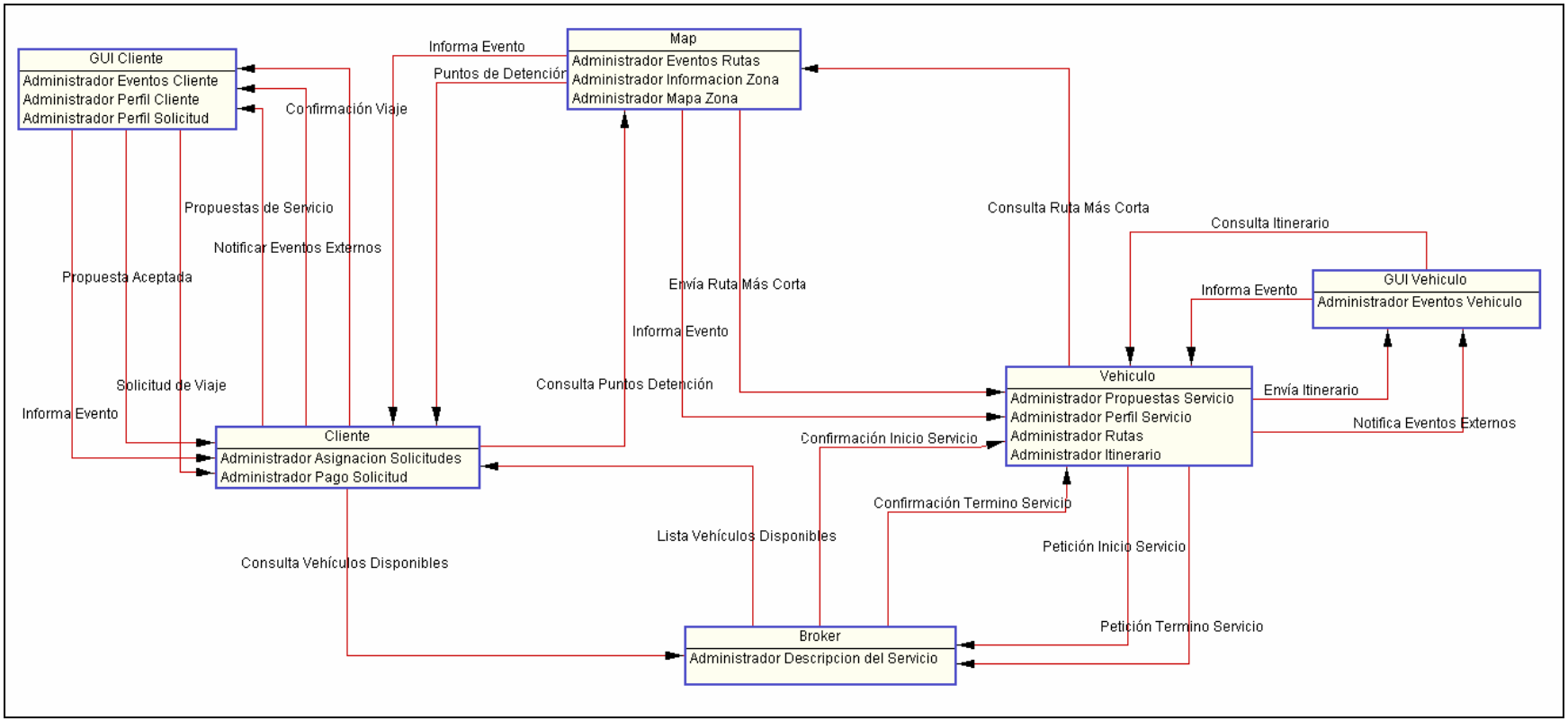

Figura 5.7 Diagrama de clases de agente.

5.2.1.4 Ensamblado de clases de agentes

En esta fase se define la arquitectura de los agentes mediante componentes. Es importante destacar que existe un diagrama de arquitectura para cada agente, pero no existe uno que muestre la arquitectura del sistema completo.

La Figura 5.8 muestra la arquitectura del agente *Broker*. En ella se aprecia que el agente posee tres componentes: los componentes *Registrar Inicio Servicio* y *Registrar Termino Servicio*, los cuales se comunican con el componente *Comunicar Información Servicio* para efectuar el inicio/término del servicio de transporte respectivamente.

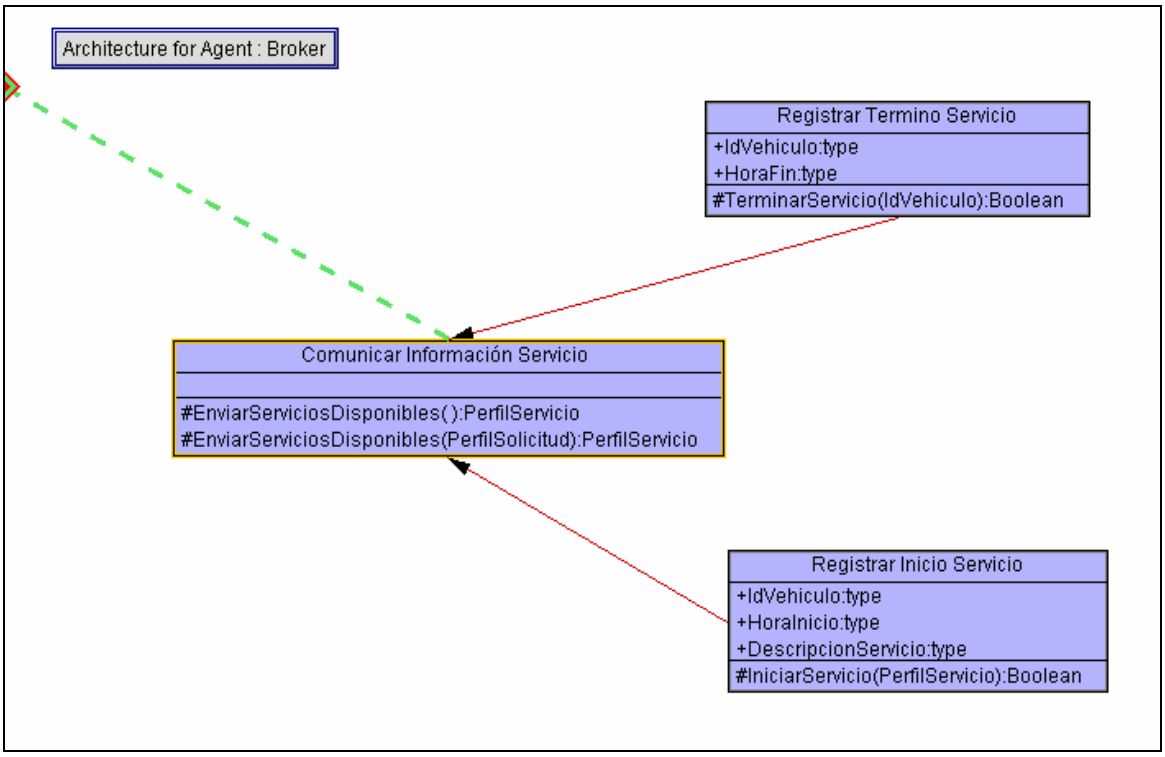

Figura 5.8 Diagrama de arquitectura agente Broker.

5.2.1.5 Despliegue del sistema

La fase *Despliegue del Sistema* es un modelo de distribución de las partes del sistema, a través de las unidades de hardware de procesado.

Como se puede apreciar en la Figura 5.9, el sistema *DRT* es un sistema distribuido en donde, por ejemplo, los agentes *GUI Cliente* y *GUI Vehículo* puede habitar en entornos distintos del resto de los agentes.

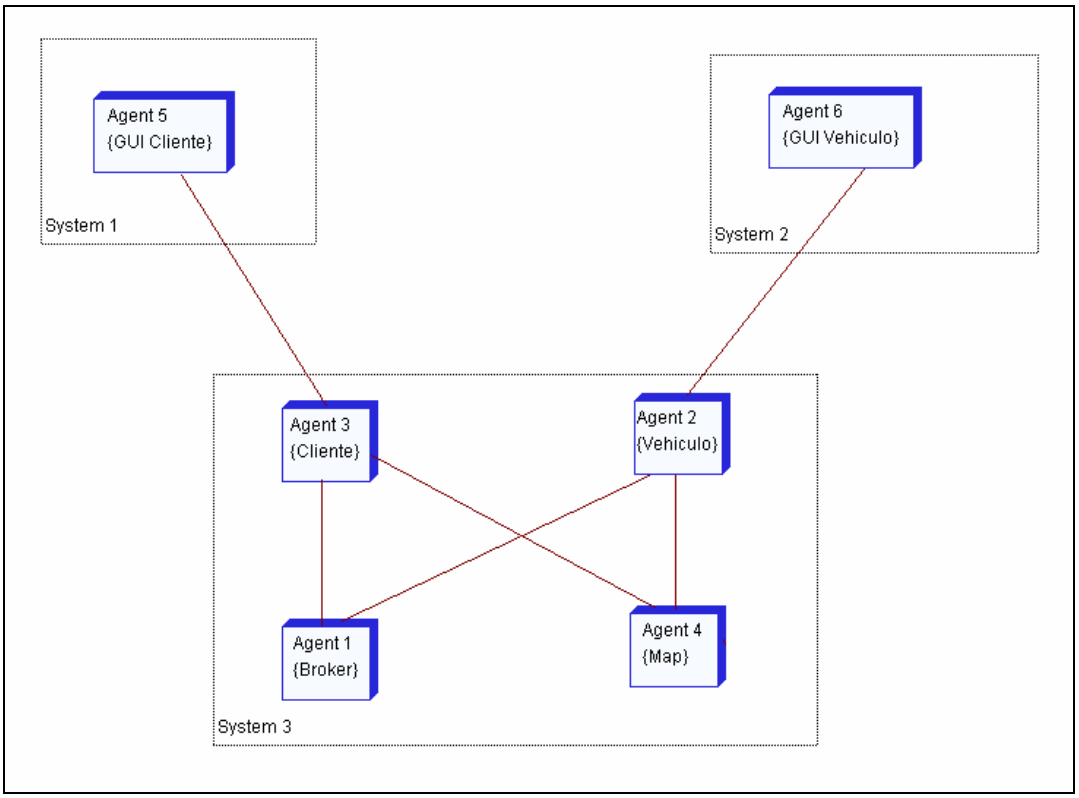

Figura 5.9 Diagrama de despliegue sistema DRT.

5.2.2 Aplicación de frameworks de evaluación

En esta sección se presenta la aplicación de los frameworks de evaluación de *Pedro Cuesta* y *Arnon Sturm* a la metodología orientada a agentes *MASE*.

5.2.2.1 Framework Arnon Sturm

Este framework utiliza cuatro criterios para la evaluación, a saber: *Conceptos y Propiedades*, *Notación y Técnicas de Modelado*, *Proceso y Pragmática*; los cuales serán presentados de acuerdo a la definición propuesta por el autor (ver sección 3.2.2).

Conceptos y Propiedades

Tabla 5.11 Evaluación de los conceptos generales de MASE según Arnon Sturm.

Concepto	Ranking	Descripción
Autonomía	7	Es expresada a través del concepto de rol. Los roles representan los objetivos del sistema durante la etapa de diseño. Cada objetivo es asociado a un rol y cada rol es desempeñado por un agente, el cual tiene la responsabilidad de cumplirlo. Por lo tanto, cada agente será capaz de tomar las acciones y decisiones que estime pertinente para alcanzar los objetivos que le fueron encomendados, sin la necesidad de intervención de otros agentes o personas. Los roles son identificados en la fase *Transformar los Objetivos a Roles*, la cual genera el *Diagrama de Roles* (ver Figura 5.6); y son utilizados en la fase *Aplicar Casos de Uso*, la cual genera varios *Diagramas de Secuencia* (ver Figura A.33).
Reactividad	5	Es expresada a través del comportamiento de un agente. El comportamiento en *MASE* se representa a nivel de conversaciones entre agentes, el cual es definido en la fase *Construir Conversaciones*. Esta fase genera un *Diagrama de Comunicación de Clases* para cada agente inmerso en la conversación (ver Figuras A.37 y A.38). Un diagrama de comunicación de clases es un autómata de estado finito que define el estado de la conversación entre dos clases de agentes. Cuando el agente recibe un mensaje (con argumentos y condiciones), transita la conversación a un nuevo estado y ejecuta cualquier acción requerida tanto por la transición como por el estado.
Pro Actividad	4	Se puede pensar que esta característica es expresada a través de los conceptos de rol y tarea en la fase *Transformar Objetivos a Roles en el Diagrama de Roles* (ver Figura 5.6); ya que un agente desempeña uno o más roles dentro de la sociedad de agentes para alcanzar un objetivo, el cual se verá concretamente alcanzado a través de la ejecución de tareas. Sin embargo, la metodología no permite darle al agente la capacidad de plantearse nuevos objetivos y de definir las tareas que estime conveniente.

Concepto	Ranking	Descripción
Habilidad Social	5	Es expresada a través del concepto de comunicación, el cual involucra los conceptos de mensaje, protocolo de comunicación y conversación. La comunicación se ve reflejada en distintas fases. La fase *Transformar Objetivos a Roles* muestra a través del *Diagrama de Roles* (ver Figura 5.6), los protocolos de comunicación existentes entre las tareas que debe realizar cada rol. Se hace distinción entre protocolos internos (comunicación entre tareas del mismo rol) y externos (comunicación entre tareas de roles distintos). En los *Diagramas de Secuencia* (ver Figura A.33) pertenecientes a la fase *Aplicar Casos de Uso*, se determinan el conjunto mínimo de mensajes que deben intercambiar los roles. Si un mensaje es traspasado entre dos roles, entonces debe existir la correspondiente comunicación entre ellos. Por último, en el *Diagrama de Conversación de Clases* (ver Figuras A.37 y A.38) de la fase *Construir Conversaciones*, se define el estado de la conversación de las dos clases de agentes participantes, entendiendo por conversación a una secuencia de mensajes entre dos (o más) agentes, los cuales apoyan un objetivo particular. Sólo queda agregar que la metodología no entrega guías ni aspectos concretos sobre el conocimiento común que los agentes deben tener sobre su entorno.

Bloques de Construcción Básicos

Tabla 5.12 Evaluación de los bloques de construcción básicos de MASE según Arnon Sturm.

Bloque	Ranking	Descripción
Agente	4	En *MASE* un agente es una abstracción útil para resolver problemas en dominios específicos. Según esto, un agente vendría caracterizado por lo siguiente: son entidades que están distribuidas, autónomas (dirigidas por objetivos) y sociales (comparten información con otros agentes de forma interactiva, conformando sistemas multiagente). De esto se desprende que la metodología se dirige la noción débil de agente (ranking promedio: 5). Pero en cuanto a la noción fuerte de agente, la metodología sólo menciona aspectos de movilidad en sus extensiones (ranking promedio: 2).
Creencia	2	La metodología incorpora esta propiedad de manera muy general. En la fase *Ensamblar Agentes* se pueden utilizar plantillas para definir la arquitectura interna de las clases de agentes. Si se escoge la plantilla correspondiente a la arquitectura *BDI*, se deben definir las creencias, deseos e intenciones.
Deseo	2	Igual que propiedad anterior.
Intención	2	Igual que propiedad anterior.

Bloque	Ranking	Descripción
Mensaje	5	En *MASE* un mensaje es una unidad individual de comunicación entre dos o más agentes. Es importante destacar que *AgentTool* no permite especificar el tipo de lenguaje de contenido a utilizar en los mensajes.
Norma	1	Bloque no especificado en la metodología.
Organización	1	Bloque no especificado en la metodología.
Protocolo	5	Los protocolos definen las diferentes maneras en que una clase de agente interactúa con otra. Sin embargo, estos son definidos de manera general. Además, *AgentTool* no permite seleccionar protocolos estándar como los protocolos *FIPA*.
Rol	7	Los roles son la base de la metodología. Específicamente son utilizados para construir las clases de agentes y contienen objetivos del sistema definidos en la etapa de análisis. En otras palabras, los roles son un puente entre lo que el sistema trata de alcanzar (etapa de análisis - objetivos) y cómo conseguirlo (etapa de diseño - clases de agente).
Servicio	1	Bloque no especificado en la metodología.
Sociedad	1	Bloque no especificado en la metodología.
Tarea	6	En *MASE* una tarea es una descripción detallada de cómo un rol cumple un objetivo. Consiste en una máquina de estados. Una tarea incluye la comunicación con otros roles y el estado de variables locales.

Notación y Técnicas de Modelado

Tabla 5.13 Evaluación de la notación y técnicas de modelado de MASE según Arnon Sturm.

Propiedades	Ranking	Descripción
Accesibilidad	5	Si bien el modelado es básicamente fácil de entender (porque se utiliza *UML* como lenguaje de modelado) no es tan simple de realizar. En la fase *Capturar Objetivos* no queda tan claro cuales son los objetivos que serán implementados como roles en fases posteriores. Además, es necesario indicar que se requiere conocimiento previo de los conceptos relacionados al área de agente para entender la metodología.
Capacidad de Análisis	4	*AgentTool* sólo verifica la consistencia entre las conversaciones. Sin embargo, verifica la eliminación correcta de algún elemento presente en los modelos y sus implicancias.

Propiedades	Ranking	Descripción
Administración de la Complejidad	5	Esta propiedad se puede apreciar en las múltiples vistas que propone la metodología. A medida que se va avanzando en el desarrollo, el nivel de detalle crece. Por ejemplo, en el *Diagrama de Roles* (ver Figura 5.6) se muestran las tareas necesarias que deben realizar los roles y la comunicación que existe entre ellas; mientras que en el *Diagrama de Clases* (ver Figura A.36) se muestra en detalle como se ejecuta una tarea, a través de diagramas de estado. Sin embargo, cabe señalar que la metodología pierde la vista global del sistema a medida en que se avanza en las etapas del proceso de desarrollo.
Ejecutabilidad y Testeabilidad	1	La metodología no entrega guías ni aspectos concretos sobre la ejecución de simulaciones o de la generación de prototipos del sistema. Además, tampoco se dirige a la inclusión de pruebas del sistema.
Expresividad y Aplicabilidad a Múltiples Dominios	4	En cuanto a la definición de la arquitectura de las clases de agente, la metodología la realiza a través del uso de plantillas (por ejemplo plantilla de la arquitectura *BDI*). Cada plantilla tiene un conjunto específico de componentes, los cuales están unidos con conectores internos y externos. En cuanto a otros aspectos, como por ejemplo la definición de ontologías, interfaces de usuario y actividades concurrentes, la metodología no los especifica ni entrega ningún tipo de guías.
Modularidad	6	El proceso de desarrollo permite especificar el sistema de manera iterativa; ya que se pueden añadir nuevos requerimientos sin afectar las especificaciones existentes.
Exactitud	5	En este punto la metodología presenta algunos problemas; ya que por ejemplo, la especificación del *Diagrama de Jerarquía de Objetivos* (ver Figuras 5.5a, 5.5b y 5.5c) no es muy clara. En éste se permite que no todos los objetivos sean implementados como roles en las fases siguientes, pero no queda claramente establecido si deben ser del mismo nivel. Por otra parte, la relación entre casos de uso y objetivos del sistema no es directa, simplemente se habla que los casos de uso deben ser definidos a partir de los requerimientos iniciales y que estos aportan información adicional.

Proceso

Tabla 5.14 Evaluación del proceso de desarrollo de MASE según Arnon Sturm.

Etapas	Ranking	Descripción
Reunión de Requerimientos	1	*MASE* no está dirigida a esta propiedad; ya que el modelado del sistema se comienza a partir de los requerimientos; los cuales han sido detallados previamente.
Análisis	4	La etapa de análisis está definida en las fases: *Capturar Objetivos*, *Transformar Objetivos a Roles* y *Aplicar los Casos de Uso*. Los entregables generados corresponden a los siguientes diagramas: *Diagrama de Jerarquía de Objetivos* (ver Figuras 5.5a, 5.5b y 5.5c), *Casos de Uso, Diagrama de Roles* (ver Figura 5.6) *y Diagramas de Secuencia* (ver Figura A.36). La metodología en esta etapa no proporciona validación ni guías de aseguramiento de la calidad y administración de proyectos.
Diseño	5	La etapa de diseño está definida en las fases: *Creación de Clases de Agentes*, *Construir Conversaciones*, *Ensamblar Clases de Agente* y *Despliegue del Sistema*. Los entregables generados corresponden a los siguientes diagramas: *Diagrama de Clases de Agentes* (ver Figura A.36), *Diagrama de Conversaciones de Clases* (ver Figuras A.37 y A.38) y *Diagrama de Despliegue del Sistema* (ver Figura 5.9). La metodología proporciona algún tipo de validación en esta etapa (ver bloque *Capacidad de Análisis*). Sin embargo, no proporciona guías de aseguramiento de la calidad y administración de proyectos.
Implementación	2	La metodología se dirige a esta etapa de manera muy general. La metodología ayuda a la creación de todos los objetos de diseño y guía el diseño del sistema para que se produzca la generación automática de código. Sin embargo, *AgentTool* no permite generar código automáticamente, pero su inclusión se presenta como trabajo futuro.
Prueba	1	La metodología no entrega guías ni aspectos concretos sobre esta etapa.

Pragmática

Tabla 5.15 Evaluación de la pragmática de MASE según Arnon Sturm.

Aspectos	Ranking	Descripción
Recursos	5	La cantidad de trabajos que explican la metodología es suficiente. Sin embargo hacen falta más ejemplos de modelados de sistema. Además, como se mencionó anteriormente, la metodología cuenta con *AgentTool* como herramienta case para el modelado del sistema.
Habilidad Requerida	5	Se requiere tener conocimientos básicos del área de agentes y sistemas multiagente, además de saber *UML*.
Aplicación de Lenguajes	6	La metodología es independiente del lenguaje de programación y de la arquitectura de agentes. En este último punto, provee cinco opciones de plantillas de arquitecturas. Las arquitecturas incluyen creencias-deseos-intenciones (*BDI*), reactividad, planificación y bases de conocimiento.
Dominio de Aplicación	5	*MASE* es una metodología de propósito general. En la literatura podemos encontrar aplicaciones a sistemas de información (por ejemplo un sistema inteligente de reuniones electrónicas). Además, se ha podido comprobar que la metodología es aplicable al desarrollo del sistema *DRT*.
Escalabilidad	5	*AgentTool* hace uso de otras herramientas como *VisualAge* para exportar/ importar proyectos, comprimiéndolos en formato *JAR*.

5.2.2.2 Framework Pedro Cuesta

Este framework utiliza cinco criterios para la evaluación, a saber: *Proceso de Desarrollo, Vistas del Modelo, Agente, Características Adicionales de Modelado y Documentación*; los cuales serán presentados de acuerdo a la definición propuesta por el autor (ver sección 3.2.3).

Proceso de Desarrollo

Tabla 5.16 Evaluación del proceso de desarrollo de MASE según Pedro Cuesta.

Aspectos a Considerar	Explicación	Evaluación
Dominio de Aplicación	Construcción de sistemas multiagentes heterogéneos.	Dominio independiente
Áreas de Aplicación	El rango de las áreas de aplicación va desde los sistemas de información tradicionales hasta sistemas más sofisticados.	----------
	Integración de base de datos heterogéneas.	Documentada
Sistemas Abiertos	----------	No
Tipo de Ciclo de Vida	Se pretende que el analista o diseñador se mueva entre las etapas libremente, tal que cada etapa sucesiva agregue más detalle, y eventualmente se produzca un completo y consistente sistema.	**Iterativo**
Etapa 1 Actividades de	**Análisis** Requerimientos Análisis Diseño Implementación Prueba Implicación del Usuario	**Bien definido** Parcialmente Enfocado Enfocado Parcialmente Enfocado No Enfocado No Enfocado Desconocido
Etapa 2 Actividades de	**Diseño** Requerimientos Análisis Diseño Implementación Prueba Implicación del Usuario	**Bien definido** No Enfocado Parcialmente Enfocado Enfocado No Enfocado No Enfocado Desconocido

Aspectos a Considerar	Explicación	Evaluación
Etapa 3	**Implementación**	**No Definida**
Actividades de	Requerimientos	No definida
	Análisis	No definida
	Diseño	No definida
	Implementación	No definida
	Prueba	No definida
	Implicación del Usuario	No definida
Etapa 4	**Prueba**	**No Definida**
Actividades de	Requerimientos	No definida
	Análisis	No definida
	Diseño	No definida
	Implementación	No definida
	Prueba	No definida
	Implicación del Usuario	No definida
Herramientas de Soporte	**AgentTool**	Apoya todas las etapas del proceso de desarrollo propuesto por la metodología.
Apoya Actividades de	Requerimientos	Parcialmente Apoyado
	Análisis	Sí
	Diseño	Sí
	Implementación: Generación de Código	Sí
	Prueba	Parcialmente Apoyado
	Orientado al Desarrollo de Herramientas	No; es independiente de la arquitectura y lenguaje de programación.

Vistas del Modelo

Tabla 5.17 Evaluación de vistas del modelo de MASE según Pedro Cuesta.

Conceptos y Representaciones para:	Concepto	Diagrama para cada Concepto
Proceso de Desarrollo	Objetivo	Diagrama de Jerarquía de Objetivos.
	Casos de Uso	Casos de Uso.
	Rol	Diagramas de Secuencia, Diagrama de Roles.
	Evento	Diagramas de Secuencia, Diagrama de Roles, Diagrama de Tareas.
	Tarea	Diagrama de Tareas.
	Mensaje (performative), Actividad	Diagrama de Clases de Agente, Diagrama de Despliegue.
	Clase Agente	Diagrama de Clases de Agente.
	Conversación	Diagrama de Comunicación de Clases.
	Arquitectura de Agente	Diagrama de Arquitectura de Agente.
	Agente (instancia)	Diagrama de Despliegue.
Análisis	Objetivo	Diagrama de Jerarquía de Objetivos.
	Casos de uso (entidad, ruta de comunicación)	Casos de uso.
	Rol, Evento	Diagrama de Secuencia.
	Rol, Tarea	Diagrama de Roles.
	Tarea, mensaje (performative), actividad	Diagrama de Tareas.
Diseño	Clases de Agente, protocolo de interacción.	Diagrama de Clases de Agentes.
	Conversaciones	Diagrama de Comunicación de Clases.

Conceptos y Representaciones para:	Concepto	Diagrama para cada Concepto
Diseño (continuación)	Arquitectura de agentes Agentes (instancia), Clases de Agentes	Diagrama de Arquitectura de Agentes. Diagrama de Despliegue.
Implementación	No considerado	No explícitamente definido.
Prueba	No considerado	No explícitamente definido.
Interacción Humana	No hay un modelo explícito, sin embargo se sugiere que un rol específico podría crearse para encapsular la interfaz de usuario.	No explícitamente definido.
Relación entre Modelos	La información acerca de las relaciones entre modelos es suficiente.	Bien definida
Entregables	Todos los diagramas previamente definidos son posibles entregables, pero no se menciona si se deben complementar con prototipos o con descripciones escritas en lenguaje formal.	No explícitamente definido.

Agente

Tabla 5.18 Evaluación del concepto de agentes en MASE según Pedro Cuesta.

Concepto	Definición	Restricciones en la Arquitectura
Agente	Los agentes en *MASE* son una especialización de los objetos, una conveniente abstracción, que puede o no poseer inteligencia, y se coordinan a través de conversaciones. Además, actúan de forma pro activa para cumplir objetivos tanto individuales como los de todo el sistema.	Los diseñadores tienen la posibilidad de elegir entre diseñar su propia arquitectura o utilizar una predefinida, como por ejemplo una arquitectura BDI. Del mismo modo, un diseñador puede utilizar componentes predefinidas o desarrollarlas por completo, entendiendo por componentes a una serie de atributos y métodos. Si la componente es compleja, puede tener una sub-arquitectura.
Atributos de Agente	Autonomía Habilidad Social	Sí Sí

Concepto	Definición	Restricciones en la Arquitectura
Atributos de Agente	Reactividad Pro actividad Inteligencia ▪ Veracidad ▪ Benevolencia ▪ Racionalidad	Sí Sí No No No
Tipos de Comunicación	No se establece explícitamente la comunicación entre e MAS desarrollado y otros sistemas, y con los usuarios.	Agente-Agente
Protocolos de Comunicación	El diseñador puede definir su propio protocolo de comunicación.	Definido por el usuario
Cooperación	Expresada a través de la comunicación (relaciones) entre agentes.	Sí
Organización de Agentes	----------	No

Características Adicionales de Modelado

Tabla 5.19 Evaluación consideraciones adicionales de modelado en MASE según Pedro Cuesta.

Características	Explicación	Evaluación
Aspectos Ontológicos	Sólo una extensión de la metodología propone la integración de ontologías en ingeniería de sistemas multiagente.	No
Movilidad	La extensión a la metodología propone diseño y especificación de movilidad.	Sí
Otros	----------	---

Documentación

Tabla 5.20 Evaluación documentación MASE según Pedro Cuesta.

Aspecto	Explicación	Evaluación
Documentación Disponible	Papers Libros Tutoriales	Suficiente Pobre Suficiente
Casos de Estudio Presentados	Sistema Electrónico Inteligente de Recolección y Decisión (EGADS) [37]. Sistema Inteligente de Reuniones Electrónicas [38].	Completo Completo

Capítulo 6

Comparación de metodologías

6.1 Framework Sturm

En esta sección se presenta la comparación de las metodologías *PASSI* y *MASE* a través de la aplicación del framework de *Arnon Sturm*. La comparación se realizó siguiendo el ejemplo presentado en [29].

6.1.1 Conceptos y propiedades

Este criterio ha sido evaluado en dos partes. La primera se refiere a la evaluación de los conceptos generales de agentes y sistemas multiagentes, y la segunda a los bloques de construcción básicos que incorporan las metodologías.

En la siguiente tabla se presentan los resultados obtenidos (ranking) tras la evaluación de los *Conceptos Generales de Agentes y Sistemas Multiagente* de ambas metodologías.

Tabla 6.1 Comparación de conceptos generales de agentes y sistemas multiagente.

Concepto	PASSI	MASE
Autonomía	7	7
Reactividad	7	5
Pro Actividad	4	4
Habilidad Social	7	5
Total	**6**	**5**

Como se aprecia en la Tabla 6.1, el nivel de **autonomía** dirigido por ambas metodologías es alto (ranking para ambas metodologías: 7). El concepto de autonomía es comúnmente interpretado como una de las propiedades clave de los agentes porque marca la diferencia entre agentes y objetos**.** De acuerdo a la investigación realizada en esta evaluación, se deduce que ambas metodologías reconocen esta importancia. Esto se debe a que expresan este concepto a través de bloques de construcción que facilitan su descripción. Por ejemplo, en ambas metodologías se utilizan los roles para encapsular los objetivos (funcionalidades) del sistema, los cuales son desempeñados por los agentes y por lo tanto, son responsables de su cumplimiento. La responsabilidad implica autonomía; ya que cada

agente deberá realizar las acciones (tareas) que estime conveniente para alcanzar los objetivos que le fueron asignados.

En cuanto al concepto de **reactividad**, esta expresado en ambas metodologías por el comportamiento de los agentes. El nivel de reactividad dirigido por las metodologías difiere (ranking *PASSI*: 7; ranking *MASE*: 5). En *PASSI* el comportamiento se representa a nivel de agente y a nivel del sistema multiagente. De esta manera, se puede obtener una vista global del comportamiento, tanto de los agentes como del sistema multiagente, pero con el suficiente detalle para su comprensión. Por ejemplo, el *Diagrama de Descripción del Comportamiento de un Agente* permite mostrar los estados por los que pasa cada agente o las diferentes actividades que realizan (dependiendo de la representación que se utilice) y las respectivas transiciones entre estos, originadas por la ocurrencia de un determinado evento; mientras que el *Diagrama de Descripción del Comportamiento del Sistema Multiagente* permite mostrar todas las interacciones entre las principales clases de agentes y sus clases internas, representadas por tareas (tareas de cooperación entre agentes). También es posible ver el comportamiento a nivel de conversaciones entre agentes; ya que en el *Diagrama de Descripción de la Ontología de Comunicación* (ver Figuras A.21a y A.21b) se puede crear un diagrama de estado para cada conversación en la que participa un agente. En cambio, en *MASE*, el comportamiento se representa sólo a nivel de las conversaciones entre agentes (ver Figuras A.37 y A.38), lo cual puede dificultar su comprensión. Por una parte se pierde la vista global del comportamiento del sistema y, por otra, se pierde también la vista global del comportamiento de un agente; ya que se muestra el comportamiento de un agente en cada una de sus interacciones, lo que significa que si un agente tiene *n* conversaciones, existirán 2*n* diagramas de comunicación de clase (para cada conversación, un diagrama para el que inicia la conversación y otro diagrama para el receptor). Otra de las causas de la diferencia en el ranking, es que el modelado de este concepto es más flexible en *PASSI*; ya que permite representar el comportamiento de los agentes y el comportamiento del sistema multiagente de la manera más apropiada para el diseñador (por ejemplo a través de diagramas de flujo, diagramas de estados o descripciones textuales semi-formales); mientras que en *MASE* el modelado se representa sólo a través de diagramas de estados.

Por otra parte, el concepto de **pro actividad** está expresado en ambas metodologías a través de los conceptos rol y tarea. El nivel de pro actividad dirigido por ambas metodologías es medio (ranking 4), esto debido a que los roles tienen asociados objetivos del sistema, los cuales se ven concretamente alcanzados a través de la ejecución de tareas. Sin embargo, las metodologías no entregan guías ni aspectos concretos sobre como construir agentes con la capacidad para plantearse nuevos objetivos, desempeñar nuevos roles y definir las tareas o acciones que estimen pertinentes para alcanzarlos.

Por último, el concepto de **habilidad social** es dirigido en distinto niveles por las metodologías (ranking *PASSI*: 7; ranking *MASE*: 5). Esto se debe a que *PASSI*, a diferencia de *MASE*, incorpora más elementos que permiten demostrar las habilidades de comunicación que poseen los agentes, como por ejemplo las ontologías. Las ontologías en *PASSI* se presentan de dos formas. La primera se encuentra en la fase *Descripción de la Ontología del Dominio*, en donde se representa el conocimiento en común que los agentes deben entender sobre su entorno. Esto permite que todos los agentes entiendan lo mismo

cuando se habla de un término en particular, disminuyendo así la ambigüedad en la comunicación. Para el caso de estudio presentado en la sección 4.3, un ejemplo de ontología es el *Evento Tráfico*, el cual representa la ocurrencia de congestión o desvío de vehículos en alguna ruta de viaje (ver Figuras A.20a y A.20b). La segunda forma de presentar el concepto de ontología se encuentra en la fase *Descripción de la Ontología de Comunicación*, en donde se muestran las interacciones permitidas entre los agentes y se especifica el protocolo de interacción utilizado en la comunicación, el lenguaje de contenido de los mensajes intercambiados y la ontología involucrada (ver Figuras A.21a y A.21b).

Los resultados de la segunda parte de la evaluación son presentados en la siguiente tabla.

Tabla 6.2 Comparación de bloques de construcción básicos.

Bloque	**PASSI**	**MASE**
Agente	4	4
Creencia	1	2
Deseo	1	2
Intención	1	2
Mensaje	6	5
Norma	1	1
Organización	1	1
Protocolo	6	5
Rol	6	7
Servicio	6	1
Sociedad	6	1
Tarea	6	6
Total	**4**	**3**

Como se aprecia en la Tabla 6.2, el nivel en que las metodologías dirigen el concepto de **agente** es medio (ranking para ambas metodologías: 4); ya que ambas se dirigen a éste pero carecen de algunos aspectos principales. En *MASE* un agente es considerado como una abstracción útil para resolver problemas en dominios específicos. Éste puede o no tener inteligencia, es autónomo, está orientado a objetivos y se comunica

con sus pares a través de mensajes. Mientras que en *PASSI* se reconoce que un agente posee ciertas características (autonomía, reactividad, pro actividad, habilidad social, racionalidad y movilidad) que lo distinguen de otras entidades (por ejemplo los objetos). De esto se desprende que ambas metodologías se dirigen a la noción débil de agentes (autonomía, reactividad, pro actividad y habilidad social) (ranking promedio *PASSI*: 6; ranking promedio *MASE*: 5). Sin embargo, en cuanto a la noción fuerte de agentes (movilidad, veracidad, benevolencia y racionalidad) se aprecia que en general las metodologías se dirigen hacia esa propiedad pero no entregan detalles (ranking promedio *PASSI*: 2; ranking promedio *MASE*: 2).

Para el caso de las **creencias, deseos e intenciones**, se puede apreciar que las metodologías se dirigen a ellas en distintos niveles (ranking promedio *PASSI*: 1; ranking promedio *MASE*: 2). *MASE* incorpora estos bloques de manera muy general en la fase *Ensamblar Agentes*. En ella se pueden utilizar plantillas para definir la arquitectura interna de las clases de agentes. Dentro de estas se encuentra la de la *Arquitectura BDI*, la cual incluye la definición de creencias, deseos e intenciones, mientras que *PASSI* no está dirigida a ninguno de estos bloques.

En cuanto a los **mensajes**, existe una pequeña diferencia en el nivel en que las metodologías se dirigen a este bloque (ranking *PASSI*: 6; ranking *MASE*: 5). La diferencia radica en que *PASSI* permite indicar los performatives de los mensajes en la fase *Descripción del Comportamiento del Sistema Multiagente*; mientras que en *MASE* esto no es posible. Además, la herramienta *CASE* de *PASSI* (*PTK*), a diferencia de la herramienta *CASE* de *MASE* (*AgentTool*), permite utilizar lenguajes de contenido estándar como el *FIPA ACL Content Lenguaje*.

En cuanto a los bloques **norma y organización**, se puede apreciar que ambas metodologías no están dirigidas a ninguno de ellos (ranking para ambas metodologías: 1).

Por otra parte, existe una pequeña diferencia en el nivel en que las metodologías se dirigen a los **protocolos de interacción** (ranking *PASSI*: 6; ranking *MASE*: 5). Ambas metodologías permiten definir protocolos de interacción, pero además *PASSI* (a diferencia de *MASE*) permite utilizar protocolos estándar como por ejemplo los protocolos *FIPA*.

Para el caso de los **roles**, estos son dirigidos a alto nivel en ambas metodologías (ranking *PASSI*: 6; ranking *MASSE*: 7). La diferencia en la puntuación se debe a que *MASE se* orienta de mayor manera a la presencia de los roles en el proceso de desarrollo del *MAS*. Los roles son la base de esta metodología y son utilizados como un puente entre lo que el sistema trata de alcanzar (etapa de análisis - objetivos) y cómo conseguirlo (etapa de diseño - clases de agente).

En cuanto al bloque **servicio**, existe una gran diferencia en el nivel en que las metodologías se dirigen a él (ranking *PASSI*: 6; ranking *MASSE*: 1). Como se puede apreciar, *PASSI* dirige este bloque sin mayores problemas. En la fase *Descripción de Roles* y *Modelo de Sociedad de Agentes* (ver Figura 5.2) se representan las dependencias entre servicios, recursos y cambios de roles; mientras que *MASE* no se dirige a esta propiedad.

Lo mismo ocurre con el bloque **sociedad** (ranking *PASSI*: 5; ranking *MASSE*: 1). *MASE* no esta dirigida a esta propiedad, mientras que *PASSI* lo hace a través del *Modelo de Sociedad de Agentes*, el cual está compuesto por las siguientes fases:

- *Descripción de la Ontología de Dominio* (ver Figuras A.20a y A.20b), en donde se representa el dominio del sistema en términos de conceptos, predicados, acciones y relaciones entre ellos.

- *Descripción de la Ontología de Comunicación* (ver Figuras A.21a y A.21b), en donde el centro de atención es la comunicación existente entre los agentes, la cual es explicada en términos de elementos ontológicos como lenguaje de contenido y protocolo.

- *Descripción de Roles* (ver Figura 5.2), en la cual se detallan los distintos roles desempeñados por los agentes en la sociedad y las tareas involucradas en cada rol.

- *Descripción de Protocolos*, la cual especifica los protocolos utilizados en las interacciones de la sociedad de agentes. Estos pueden ser definidos por el diseñador o utilizar protocolos estándares como los protocolos *FIPA*. Un aspecto importante a destacar es que la metodología no entrega guías o aspectos concretos sobre normas o reglas que rijan a la sociedad de agentes.

Por último, el bloque **tarea** es dirigido a alto nivel en ambas metodologías (ranking para ambas metodologías: 6), y esto se debe a que es definido de manera clara y precisa. En ambas metodologías se utilizan para describir como los roles cumplen con sus objetivos. En *PASSI* se pueden identificar principalmente en el *Modelo de Requerimientos del Sistema* (fase *Especificación de Tareas*) y en el *Modelo de Sociedad de Agente* (fase *Descripción de Roles*), mientras que en *MASE* se identifican en la fase *Transformar Objetivos a Roles* (*Diagrama de Rol* y *Diagrama de Tareas*).

Como conclusión general de la evaluación de los *Conceptos Generales y Bloques de Construcción Básicos de Agentes y Sistemas Multiagentes*, se tiene que ambas metodologías se dirigen a este criterio, pero carecen de aspectos principales (ranking promedio para ambas metodologías: 4). En cuanto a los *Agentes*, falta desarrollar características tales como veracidad, benevolencia, racionalidad y movilidad. Además, se deben incluir en su implementación aspectos como creencias, deseos, intenciones y veracidad. En cuanto a la *Sociedad de Agentes*, falta establecer normas o reglas que la rijan e incorporar más aspectos de la organización de esta.

6.1.2 Notación y técnica de modelado

Los resultados obtenidos tras la evaluación de la *Notación y Técnica de Modelado* utilizadas por las metodologías son presentados en la siguiente tabla.

Tabla 6.3 Comparación de notación y técnica de modelado.

Propiedades	PASSI	MASE
Accesibilidad	6	5
Capacidad de Análisis	7	4
Administración de la Complejidad	7	5
Ejecutabilidad y Testeabilidad	2	1
Expresividad y Aplicabilidad	5	4
Modularidad	4	6
Exactitud	7	5
Total	**5**	**4**

Como se puede apreciar en la Tabla 6.3, existe una pequeña diferencia en el nivel de **accesibilidad** de ambas metodologías (ranking *PASSI*: 6; ranking *MASE*: 5). El modelado bajo ambas metodologías es fácil de entender porque utilizan *UML* como lenguaje de modelado. Sin embargo, al momento de llevarlo a cabo *PASSI* dirige de mejor manera esta propiedad; ya que los diversos modelos propuestos son claros y precisos, no así en *MASE*. El desarrollo de la fase *Capturar Objetivos* no es completamente claro, se deben identificar los objetivos que serán implementados como roles en las fases posteriores, pero no se establece si deben pertenecer al mismo nivel de la jerarquía. Cabe destacar que para la aplicación de ambas metodologías se requiere el conocimiento de conceptos relacionados al área de agente. Además, las metodologías no están orientadas a explicar conceptos sino a su aplicación, pero ayudan a madurarlos.

En cuanto a la **capacidad de análisis** de las metodologías, esto es, revisar la consistencia interna de los modelos o identificar interrelaciones entre operaciones que parecen sin conexión; estas difieren significativamente (ranking *PASSI*: 7; ranking *MASE*: 4). La principal diferencia se encuentra en que el *PTK* verifica la consistencia de todos los

modelos al pasar de una fase a otra, mientras que *AgentTool* sólo verifica que las conversaciones del *Diagrama de Despliegue del Sistema* existan en el *Diagrama de Clases de Agentes* y que estas sean consistentes.

En cuanto a la **administración de la complejidad** en las metodologías, el nivel difiere (ranking *PASSI*: 7; ranking *MASE*: 5). En las distintas vistas que proponen las metodologías se puede apreciar que a medida que se va avanzando en el desarrollo, el nivel de detalle crece. Esto lo logra con mayor éxito *PASSI*, ya que en todos los modelos se evidencia de manera clara el propósito de cada vista del sistema. Además, a medida que se avanza en las etapas del proceso de desarrollo se agrega más detalle, pero siempre se mantiene una visión global del sistema (ver Figuras 5.1, A.21a, A.21b y 5.3). Por el contrario, en *MASE* esto se pierde; ya que a medida que se avanza en las etapas del proceso se centra más en la agregación de detalles (ver Figuras 5.5a, 5.5b, 5.5c, 5.6 y 5.8).

Para el caso de la **ejecutabilidad y testeabilidad,** las metodologías no desarrollan este aspecto (ranking *PASSI*: 2; ranking *MASE*: 1). Ninguna permite ejecutar una simulación o generar un prototipo del sistema, y sólo *PASSI* menciona que se deben realizar pruebas en la fase de código y despliegue del sistema, pero no se especifica cómo.

Por otra parte, el nivel de **expresividad y aplicabilidad** en las metodologías es medio (ranking *PASSI*: 5; ranking *MASE*: 4). Tanto *PASSI* como *MASE* incorporan en sus diversas fases los conceptos relacionados con el flujo de datos dentro del sistema, con agentes y con la arquitectura del sistema; pero sólo *PASSI* incorpora el concepto de ontología (ver Figuras A.20a, A.20b, A.21a y A.21b). Sin embargo, existen otros aspectos, por ejemplo la definición de interfaces de usuario y actividades concurrentes, que no son especificados por las metodologías.

En cuanto a la **modularidad**, existen pequeñas diferencias entre ambas metodologías (ranking *PASSI*: 4; ranking *MASE*: 5). Para ambos casos los procesos de desarrollo permiten especificar el sistema de manera iterativa, pero en *MASE* esto se logra de mejor manera, ya que *AgentTool* permite añadir nuevos requerimientos sin afectar las especificaciones existentes. En cambio, el *PASSI Tool Kit* no permite la realización de cambios en los diagramas una vez que se ha pasado a la siguiente fase, lo cual dificulta la agilidad del desarrollo. Al realizar cambios se pierden todos los diagramas elaborados teniéndose que rehacer todo el trabajo.

Por último, el nivel de **exactitud** de las metodologías difiere (ranking *PASSI*: 7; ranking *MASE*: 5). Las diferencias en la evaluación se deben a que en *PASSI* cada modelo es especificado de manera no ambigua, mientras que en *MASE* la especificación del *Diagrama de Jerarquía de Objetivos* no es muy clara (ver Figuras 5.5a, 5.5b y 5.5c). En este diagrama los objetivos del sistema son ordenados por niveles de acuerdo a su grado de importancia y detalle, para que el diseñador pueda identificar cuales son los objetivos que se implementarán como roles en las fases posteriores. El problema radica en que no queda explícitamente establecido si estos objetivos deben ser del mismo nivel. Además, la relación entre casos de uso y objetivos del sistema no es directa, simplemente se habla de que los casos de uso tienen que ser definidos a partir de los requerimientos iniciales y que estos aportan información adicional.

Como conclusión general de la evaluación del criterio *Notación y Técnica de Modelado* se tiene que *PASSI* se dirige a él de mejor manera que *MASE* (ranking promedio *PASSI*: 5; ranking promedio *MASE*: 4); ya que administra de mejor manera la complejidad y posee una mayor exactitud en los modelos que propone. Por otra parte, es importante destacar la deficiencia de ambas metodologías en cuanto a la ejecutabilidad y testeabilidad de las especificaciones que se desarrollan.

6.1.3 Proceso

Los resultados obtenidos tras la evaluación de las dos metodologías con respecto a sus *Procesos de Desarrollo* son presentados en la siguiente tabla.

Tabla 6.4 Comparación de procesos de desarrollo.

Etapas	PASSI	MASE
Reunión de Requerimientos	1	1
Análisis	5	4
Diseño	5	5
Implementación	5	2
Prueba	2	1
Total	**4**	**3**

Como se puede apreciar en la Tabla 6.4, las metodologías no se dirigen a la etapa de **reunión de requerimientos** (ranking para ambas metodologías: 1). Esto se debe principalmente a que el modelado del sistema se comienza a partir de los requerimientos, los cuales han sido detallados previamente.

El nivel en que ambas metodologías se dirigen a la etapa de **análisis** es medio, sólo se aprecia una pequeña diferencia (ranking *PASSI*: 5; ranking *MASE*: 4). Ambas metodologías presentan en esta etapa una serie de actividades (apoyadas por ejemplos de su desarrollo) que ayudan a alcanzar la comprensión de los requerimientos. *MASE* presenta las fases: *Capturar Objetivos*, *Transformar Objetivos a Roles* y *Aplicar los Casos de Uso*, mientras que PASSI presenta las fases: *Descripción del Dominio*, *Identificación de Agentes*, *Identificación de Roles* y *Especificación de Tareas* (definidas en el *Modelo de Requerimientos del Sistema*). Sin embargo, *PASSI* logra que el desarrollo de esta etapa sea más fácil de comprender (ver sección 6.1.2, propiedad de accesibilidad). Por otra parte, es importante mencionar que ambas metodologías no proporcionan guías de aseguramiento de la calidad ni administración de proyectos en esta etapa.

Por otra parte, se puede apreciar que el nivel en que las metodologías se dirigen a la etapa de **diseño** es medio (ranking para ambas metodologías: 5). Ambas metodologías presentan en esta etapa una serie de actividades, apoyadas por ejemplos de su desarrollo. *MASE* presenta las fases: *Creación de Clases de Agentes*, *Construir Conversaciones*, *Ensamblar Clases de Agente* y *Despliegue del Sistema*, mientras que *PASSI* presenta las fases: *Descripción de la Ontología de Dominio, Descripción de la Ontología de Comunicación, Descripción de Roles y Descripción de Protocolos* (definidas en el *Modelo de Sociedad de Agentes*). Al igual que en la etapa anterior, ambas metodologías no proporcionan guías de aseguramiento de la calidad ni administración de proyectos.

En cuanto a la etapa de **implementación**, se puede apreciar que existen grandes diferencias en el nivel en que las metodologías se dirigen a ella (ranking *PASSI*: 5; ranking *MASE*: 2). Esto se debe a que *MASE*, a diferencia de *PASSI*, se dirige a esta etapa de manera muy general. La metodología ayuda a la creación de todos los objetos de diseño y guía el diseño del sistema para que se produzca la generación de código, pero no se explicita ninguna actividad. Por el contrario, *PASSI* define esta etapa en el *Modelo de Implementación de Agentes* y *Modelo de Código*. Las actividades realizadas en esta etapa son: *Definición de la Estructura del MAS, Descripción del Comportamiento del MAS, Definición de la Estructura del Agente* y *Descripción del Comportamiento del Agente*. Además, el *PTK* permite la generación automática de código, por lo menos de las estructuras generales (clases de agentes y sus comportamientos). Sin embargo, cabe mencionar que *PASSI* no proporciona guías de aseguramiento de la calidad ni administración de proyectos en esta etapa.

Por último, se puede apreciar que la etapa de **prueba** no es dirigida mayormente por las metodologías (ranking *PASSI*: 2; ranking *MASE*: 1). *MASE* no está dirigida a esta etapa y *PASSI* sugiere a grandes rasgos la inclusión de pruebas unitarias y pruebas del sistema en las fases de *Código* y *Despliegue del Sistema*.

Como conclusión general de la evaluación del criterio *Proceso* se tiene que *PASSI* se dirige a él de mejor manera que *MASE* (ranking promedio *PASSI*: 4; ranking promedio *MASE*: 3), ya que incluye la etapa de implementación en su proceso de desarrollo. Sin embargo, ambas metodologías no incluyen mayormente la etapa de prueba y guías sobre aseguramiento de la calidad y administración de proyectos.

6.1.4 Pragmática

En la siguiente tabla se presentan los resultados obtenidos tras la evaluación del criterio *Pragmática*, para ambas metodologías.

Tabla 6.5 Comparación de pragmática.

Aspectos	PASSI	MASE
Recursos	5	4
Habilidad Requerida	5	5
Aplicación de Lenguajes	6	6
Dominio de Aplicación	6	5
Escalabilidad	6	6
Total	**6**	**5**

Como se puede observar en la Tabla 6.5, los **recursos** dispuestos por las metodologías alcanzan un nivel medio (ranking *PASSI*: 5; ranking *MASE*: 4). Ambas metodologías entregan documentación en donde se describen las fases de cada etapa y cuentan con herramientas *CASE* que facilitan el proceso de desarrollo y su respectiva documentación. Sin embargo, se puede encontrar un mayor número casos de estudio (ejemplos de aplicación), modelados bajo la metodología *PASSI*.

En cuanto a la **habilidad requerida** para aplicar las metodologías, el nivel alcanzado es medio (ranking para ambas metodologías: 5). Se requiere tener conocimientos básicos del área de agentes y sistemas multiagente, además de saber *UML*.

Para el caso de la **aplicación de lenguajes**, el nivel alcanzado es medio (ranking para ambas metodologías: 5). Ambas metodologías son independientes del lenguaje de programación y de la arquitectura de agentes.

Por otra parte, el **dominio de aplicación** de las metodologías alcanza un nivel alto (ranking *PASSI*: 6; ranking *MASE*: 5). Ambas metodologías son de propósito general. Sin embargo, en la literatura se encuentran más ejemplos de aplicación bajo diversos dominios (robótica, biología celular, sistemas de información tradicionales, etc.) utilizando *PASSI*. Además, es importante destacar que se ha comprobado que ambas metodologías son aplicables al desarrollo del sistema *DRT*.

Por último, en cuanto a la **escalabilidad** de las metodologías se aprecia que el nivel alcanzado es alto (ranking para ambas metodologías: 6). A través del *PTK* se pueden utilizar funcionalidades provistas por *Rational Rose* como por ejemplo, la exportación/

importación de proyectos (en formato *JAR*). Por su parte, *AgentTool* también permite estas funcionalidades a través de otras herramientas como *VisualAge*.

Como conclusión general de la evaluación del criterio *Pragmática* se observa que ambas metodologías se dirigen a él de buena manera (ranking promedio *PASSI*: 6; ranking promedio *MASE*: 5), sin embargo *PASSI* posee una pragmática más desarrollada que *MASE*, principalmente por la cantidad de recursos entregados y la diversidad de casos de estudio en distintos dominios de aplicación documentados.

6.2 Framework Cuesta

En esta sección se presenta la comparación de las metodologías *PASSI* y *MASE* a través de la aplicación del framework de *Pedro Cuesta*.

6.2.1 Proceso de desarrollo

En la siguiente tabla se presentan los resultados obtenidos tras la evaluación del criterio *Proceso de Desarrollo* para ambas metodologías.

Tabla 6.6 Comparación proceso de desarrollo.

Aspectos a Considerar	PASSI	MASE
Dominio de Aplicación	Dominio Independiente	Dominio independiente
Áreas de Aplicación	Robótica (Documentada) Biología Celular (Documentada) Sistemas de Información Tradicionales (Documentada)	Integración de base de datos heterogéneas (Documentada).
Sistemas Abiertos	No	No
Tipo de Ciclo de Vida	Iterativo	Iterativo
Requerimientos	No enfocada	No enfocada
Análisis	Enfocada	Enfocada
Diseño	Enfocada	Enfocada
Implementación	Enfocada	No enfocada
Prueba	Parcialmente Enfocada	No enfocada
Herramientas de Soporte	PASSI Tool Kit	AgentTool

Como se aprecia en la Tabla 6.6, ambas metodologías son de propósito general para el desarrollo de sistemas multiagentes heterogéneos, en otras palabras, su **dominio de aplicación** es independiente. Además, ninguna de las metodologías está pensada para ser aplicada a sistemas abiertos. En *MASE* se asume que los sistemas que utilizan la metodología para su desarrollo son cerrados.

Respecto a las **áreas de aplicación**, para el caso de *PASSI* se encuentran diversos ejemplos en la literatura. Estos abarcan distintos dominios como por ejemplo, el desarrollo de un sistema de información biológico [34], de un sistema de información para una biblioteca [35] y de un sistema de robótica [36]. En cambio *MASE* sólo presenta el desarrollo de los casos de estudio de un sistema electrónico inteligente de recolección y decisión (EGADS) [37], y de un sistema inteligente de reuniones electrónicas [38]. Sin embargo, es importante destacar que se ha comprobado que ambas metodologías son aplicables al caso de estudio propuesto (desarrollo de un sistema *DRT*).

Por otra parte, el **ciclo de vida** de ambas metodologías es iterativo. En *MASE* se logra que el analista o diseñador se mueva entre las etapas y sus diversas fases libremente, de tal modo que en cada iteración se agrega detalle hasta obtener el diseño de un sistema completo y coherente. En cambio en *PASSI* esto no se alcanza con tanto éxito; ya que el proceso de desarrollo es apoyado por la herramienta *CASE PASSI Tool Kit*, la cual no permite la realización de cambios en los diagramas una vez que se ha pasado a la siguiente fase. Al realizar cambios se pierden los diagramas elaborados, teniéndose que rehacer todo el trabajo.

En cuanto al proceso de desarrollo mismo, las metodologías no incorporan actividades de **requerimientos**. Sin embargo se dirigen a las etapas de **análisis y diseño** sin mayores problemas, realizando una serie de actividades que ayudan a alcanzar sus propósitos. En *MASE*, la etapa de análisis propone las fases de *Capturar Objetivos*, *Transformar Objetivos a Roles* y *Aplicar Casos de Uso*, con el objetivo de elaborar un conjunto de roles, cuyas tareas describen lo que el sistema debe hacer para satisfacer todos sus requerimientos. La etapa de diseño propone las fases de *Crear Clases de Agentes*, *Construir Conversaciones*, *Ensamblar Clases de Agente* y *Despliegue del Sistema*, con el fin de definir la organización general del sistema a través de la transformación de los roles y las tareas definidas en el análisis, en clases de agentes y conversaciones. En cambio en *PASSI*, la etapa de análisis propone las fases *Descripción del Dominio*, *Identificación de Agentes*, *Identificación de Roles* y *Especificación de Tareas*; y la etapa de diseño propone las fases *Descripción de la Ontología de Dominio*, *Descripción de la Ontología de Comunicación*, *Descripción de Roles y Descripción de Protocolos.* A modo general, se puede concluir que las etapas de análisis y diseño en ambas metodologías están relacionadas; ya que poseen actividades que incorporan aspectos compartidos. Por ejemplo, en *PASSI* la etapa de análisis y diseño está relacionada a través de la fase *Identificación de Roles*; ya que ayuda a identificar los roles que desempeñarán los agentes (concerniente a la etapa análisis) y las interacciones que existen entre ellos (concerniente a la etapa de diseño); mientras que en *MASE* la conexión es posible verla en las fase *Transformar Objetivos a Roles* y *Crear Clases de Agentes.* Por otra parte, las principales diferencias entre las metodologías radican en la completitud de sus procesos de desarrollo. *PASSI*, a diferencia de *MASE*, desarrolla las etapas de **implementación y prueba** (aunque esta

última a grandes rasgos).Además, cabe agregar que ninguna de las etapas propuestas por las metodologías incorpora actividades que integren la participación de los usuarios.

Por último, otro aspecto a considerar es que ambas metodologías utilizan **herramientas de soporte** que apoyan cada una de las fases propuestas. *PASSI* posee *PASSI Tool Kit* (*PTK*), la cual permite la verificación y generación de cada uno de los modelos propuestos por la metodología y la generación de código. *MASE* posee *AgentTool*, la cual permite la generación de cada uno de los modelos propuestos por la metodología y la verificación de las conversaciones entre los agentes.

6.2.2 Vistas del modelo

Los resultados obtenidos tras la evaluación del criterio *Vistas del Modelo* de ambas metodologías se presentan en la siguiente tabla.

Tabla 6.7 Comparación vistas del modelo.

Conceptos y Representaciones:	**PASSI**	**MASE**
Proceso de Desarrollo	Agente, rol, tarea, requerimiento, objetivo, escenario, mensaje, ontología, protocolo, comunicación, performative, concepto, acción, predicado, agente plataforma *FIPA*, tarea plataforma *FIPA*, servicio y recurso.	Objetivo, casos de uso, rol, evento, tarea, mensaje, actividad, clase agente, conversación, protocolo, arquitectura de agente y agente (instancia).
Requerimientos	No considerado	No considerado
Análisis	Requerimiento, objetivo, caso de uso, agente, rol, comunicación, escenario y tarea.	Objetivo, casos de uso (entidad, ruta de comunicación), rol, evento, rol, tarea, mensaje, actividad.
Diseño	Ontología, concepto, acción, predicado, comunicación, agente, rol, tarea, protocolo de interacción, performative, servicio y recurso.	Clases de agente, conversaciones (protocolo de interacción), arquitectura de agentes y agentes (instancia).
Implementación	Mensaje, protocolo de interacción de agente, performative, ontología, concepto, acción, predicado, agente, agente plataforma *FIPA*, tarea plataforma *FIPA*.	No considerado
Prueba	No especificado	No considerado
Interacción Humana	No explícitamente definido.	No explícitamente definido.
Relación entre Modelos	Bien definida	Bien definida
Entregables	No explícitamente definido	No explícitamente definido

Como se aprecia en la Tabla 6.7, en general las metodologías incorporan los mismos conceptos en sus **procesos de desarrollo**. La principal diferencia está en el concepto de ontología, el cual es considerado sólo en *PASSI*.

En cuanto a la etapa de **requerimientos**, ambas metodologías no incorporan conceptos y representaciones puesto que no la desarrollan.

Para la etapa de **análisis**, ambas metodologías incorporan los conceptos de objetivo, caso de uso, rol y tarea. En *PASSI*, las representaciones de estos conceptos se encuentran en los diagramas de *Casos de Uso, Identificación de Roles y Especificación de Tareas* respectivamente. En *MASE*, dichas representaciones se encuentran en los diagramas de *Jerarquía de Objetivos, Casos de Uso, Secuencia, Roles* y *Tareas*. En general se puede apreciar que ambas metodologías incluyen conceptos y representaciones similares.

Por otra parte, en la etapa de **diseño** ambas metodologías incorporan los conceptos de protocolo de interacción y agente. En *PASSI*, el concepto de protocolo es representado en los diagramas de *Descripción de la Ontología de Comunicación* y *Descripción de Protocolos*, mientras que el concepto de agente es representado en los diagramas de *Descripción de la Ontología de Comunicación y Descripción de Roles*. En *MASE* el concepto de protocolo es representado en el diagrama de *Clases de Agentes*; y el concepto de agente es representado en el diagrama de *Clases de Agentes* (como clase) y en el diagrama de *Despliegue* (como instancia). La principal diferencia en esta etapa es que sólo *PASSI* incorpora el concepto de ontología.

En cuanto a la etapa de **implementación**, sólo *PASSI* incorpora conceptos y sus respectivas representaciones, puesto que *MASE* no desarrolla esta etapa. En *PASSI* el diagrama *Descripción Comportamiento del Sistema Multiagente* representa los conceptos mensaje, protocolo de interacción, performative y ontología (concepto, predicado y acción). En cuanto al concepto de agente, se hace una diferencia entre agente y agente *FIPA*. El primero es representado en los diagramas *Descripción Estructura del Sistema Multiagente* y *Descripción Comportamiento del Sistema Multiagente.* El segundo es representado en los diagramas *Descripción Estructura Agente, Descripción Comportamiento del Agente, Descripción Comportamiento del Sistema Multiagente, Reutilización de Código, Producción de Código* y *Configuración del Despliegue*.

La etapa de **prueba** no esta especificada por ninguna de las metodologías. *MASE* no desarrollada esta etapa y *PASSI*, a pesar que la considera, no entrega ninguna guía al respecto.

Respecto de la **interacción humana**, las metodologías no poseen un modelo explicito que muestre su inclusión. Sin embargo, *MASE* sugiere que un rol específico podría crearse para encapsular la interfaz de usuario.

En cuanto a la **relación entre los modelos**, en ambas metodologías está bien definida; ya que los modelos propuestos permiten ver como los requerimientos iniciales se transforman en el sistema mismo.

Por último, en ambas metodologías todos los diagramas previamente definidos son posibles **entregables**, pero no se menciona si se deben complementar con prototipos o con descripciones escritas en lenguaje formal.

6.2.3 Agente

La siguiente tabla muestra los resultados obtenidos tras la evaluación del concepto *Agente* y otros aspectos relacionados.

Tabla 6.8 Comparación concepto agente.

Concepto	PASSI	MASE
Agente	Entidad de software con capacidad de razonamiento.	Abstracción conveniente que puede o no poseer inteligencia.
Autonomía	Sí	Sí
Habilidad Social	Sí	Sí
Reactividad	Sí	Sí
Pro actividad	Sí	Sí
Inteligencia		
Veracidad	No	No
Benevolencia	No	No
Racionalidad	Sí	No
Tipos de Comunicación	Agente-Agente	Agente-Agente
Protocolos	Definido por el usuario	Definido por el usuario
Cooperación	Sí	Sí
Organización	No	No

Como se puede apreciar en la Tabla 6.8, en ambas metodologías los agentes poseen las características de **autonomía, reactividad, pro actividad y habilidad social**.

Respecto de otras capacidades de los agentes como la **veracidad y benevolencia**, estas no son consideradas por las metodologías.

Por otra parte, la principal diferencia observada en la evaluación es que en *MASE* un agente es considerado como una abstracción útil para resolver problemas en dominios específicos, que puede o no tener inteligencia, mientras que en *PASSI* se reconoce que un agente posee **racionalidad**.

Para el caso del **tipo de comunicación,** las metodologías sólo soportan la comunicación entre agentes. No establecen explícitamente la comunicación entre el *MAS* desarrollado y otros sistemas, ni la relación entre agentes y usuarios.

En cuanto a la **cooperación**, este aspecto se aprecia en las relaciones que establecen los agentes del sistema; ya que el propósito de su comunicación es conseguir algo (información, servicio, etc.) que se necesita para alcanzar un objetivo concreto.

Por último, respecto de la **organización** de los agentes del sistema, las metodologías no entregan guías ni aspectos concretos sobre esto.

6.2.4 Características adicionales de modelado

Los resultados obtenidos tras la evaluación de las características adicionales de modelado de ambas metodologías se presentan en la siguiente tabla.

Tabla 6.9 Comparación características adicionales de modelado.

Características	PASSI	MASE
Aspectos Ontológicos	Sí	No
Movilidad	Sí	Sí
Otros	No	No

Como se aprecia en la Tabla 6.9, sólo *PASSI* incorpora aspectos **ontológicos** (fases *Descripción de la Ontología del Dominio* y *Descripción de la Ontología de Comunicación*). Sin embargo, en una extensión de *MASE* se propone la integración de ontologías.

En cuanto a la **movilidad** de los agentes, tanto en *PASSI* como en *MASE* se contempla, pero no se especifica como implementarla. De hecho, se podría pensar que *MASE* es la más cercana a este aspecto; ya que *AgentTool* ofrece en su menú la opción de crear componentes con movilidad, pero en realidad la herramienta no permite realizar esta acción. De todas formas, *MASE* no se queda atrás; ya que se han presentado extensiones para el diseño y la especificación de la movilidad [39].

6.2.5 Documentación

La siguiente tabla muestra los resultados obtenidos tras la evaluación de la documentación que entregan las metodologías.

Tabla 6.10 Comparación de documentación.

Aspecto	PASSI	MASE
Documentación Disponible	Papers (buena) Libros (pobre) Tutoriales (suficiente)	Papers (suficiente) Libros (pobre) Tutoriales (suficiente)
Casos de Estudio Presentados	Completo	Parcial

Como se puede apreciar en la Tabla 6.10, la **documentación** que ofrecen las metodologías es variada. La principal documentación disponible son papers y existe una cantidad suficiente de tutoriales que ayudan a comprender cada una de las metodologías y sus respectivas herramientas de soporte. Sin embargo, la cantidad de libros publicados es baja.

Además, como se mencionó anteriormente, en la literatura se encuentran diversos **casos de estudio** aplicando *PASSI*, como por ejemplo el desarrollo de: un sistema de información biológico [34], de un sistema de información para una biblioteca [35] y de un sistema de robótica [36]. En cambio *MASE* sólo presenta el desarrollo de los casos de estudio de un sistema electrónico inteligente de recolección y decisión (EGADS) [37] y de un sistema inteligente de reuniones electrónicas [38].

Capítulo 7

Conclusiones

7.1 Referentes al proyecto

En este documento se presentó lo correspondiente al desarrollo total del proyecto, esto es, la evaluación de las metodologías orientadas a agente *PASSI* y *MASE* a través de la utilización de los frameworks de evaluación de *Pedro Cuesta* y *Arnon Sturm*. Además, dicha evaluación fue apoyada por la aplicación de ambas metodologías al caso de estudio de un sistema de transporte de pasajeros en respuesta a la demanda (*DRTS*). Por lo tanto, se puede concluir que lo objetivos del proyecto, tanto el general como los específicos, fueron alcanzados.

Por otra parte, se puede concluir que el desarrollo del caso de estudio propuesto ayudó de gran manera a la evaluación, y posterior comparación de las metodologías orientadas a agente *MASE* y *PASSI*; ya que éste permitió conocer a fondo la aplicación de cada una de las metodologías. No hubiese bastado la mera aplicación de los frameworks de evaluación, puesto que estos sirven para realizar evaluaciones a nivel general.

7.2 Referentes al trabajo realizado

Después de aplicar ambas metodologías al caso de estudio propuesto (sistema de transporte de pasajeros en respuesta a la demanda) se ha llegado a las siguientes conclusiones:

- *PASSI* evidencia de mejor manera la relación entre los modelos que propone; ya que por ejemplo, en la etapa de análisis (*Modelo de Requerimientos del Sistema*) los requerimientos iniciales del sistema son claramente trasables entre los distintos diagramas generados. En el diagrama *Descripción del Dominio* (ver Figura A.2) se representan los requerimientos iniciales del sistema a través de casos de uso, luego en el diagrama *Identificación de Agentes* (ver Figura A.3) se identifican los agentes que compondrán el sistema y a cada uno se les asigna casos de uso (funcionalidades) hasta cubrirlos todos. Posteriormente, en el diagrama *Identificación de Roles* (ver Figura A.5) las funcionalidades anteriormente identificadas son encapsuladas en roles, los cuales serán desempeñados por los agentes. Por último, en el diagrama de *Especificación de Tareas* (ver Figura A.14) se especifican las tareas que cada agente deberá ejecutar para alcanzar la funcionalidad deseada. En cambio en *MASE*, en esta misma etapa, el seguimiento de las funcionalidades del sistema no es tan evidente. En el diagrama de *Jerarquía de Objetivos* (ver Figuras 5.5a, 5.5b y 5.5c) son identificados los objetivos (funcionalidades) del sistema. Luego, se deben

identificar los casos de uso a partir de los objetivos, pero estos son definidos de forma textual y en otra vista perdiéndose la conexión. Posteriormente, en el *Diagrama de Roles* (ver Figura 5.6) se identifican los roles que desempeñarán los agentes y se les asigna un objetivo especifico, pero luego hay que volver a los casos de uso y crear un *Diagrama de Secuencia* para cada uno de ellos para identificar las comunicaciones que existen entre los roles y los mensajes mínimos que intercambian (ver Figuras A.33, A.34 y A.35).

- Sólo *PASSI* permitió identificar a los actores que se relacionan con el sistema. Sin embargo, ninguna de las metodologías permitió mostrar la comunicación entre el *MAS* y otros sistemas. Este aspecto es relevante para el caso de estudio; ya que gran parte de la información utilizada por el *DRTS* es provista por sistemas externos. Por ejemplo, el *Sistema de Información de Tráfico* provee información sobre el estado de las rutas (congestiones, desvíos) y de la zona geográfica cubierta (localización de direcciones y paradas, nombres de calles, distancias entre localizaciones y su tiempo de traslado correspondiente). Pero también, el *DRTS* entrega información a otros sistemas, por ejemplo el *Sistema de Pago*. Este sistema es el encargado de gestionar el proceso de pago de los servicios de transportes, para lo cual necesita recibir de parte del *DRTS* información referente a: la descripción del servicio entregado (detalle del viaje, tarifas), los clientes (datos personales), la modalidad de pago (pago a bordo o de antemano) y opciones de pago (tarjeta de crédito, efectivo o boleto). Por otra parte, las metodologías tampoco consideran la interacción entre los usuarios (clientes, conductores y operadores) y el *MAS*. No hay que olvidar que el sistema está integrado por dos tipos de agentes que se relacionan estrechamente con los usuarios, el agente *GUI Cliente* y el agente *GUI Vehículo*.

- La organización de las vistas del sistema en *PASSI* ayuda a comprender de mejor manera el *MAS* que se desea construir. Por ejemplo, cuando se llega a la etapa de diseño en *MASE*, fase *Crear Clases de Agente* (ver Figura 5.7), recién se tiene la primera visión de cuales son los agentes que compondrán el sistema; mientras que en *PASSI* esto ocurre en la etapa de análisis en la fase *Identificación de Agentes* (ver Figura 5.1). Para una persona que desee construir un sistema multiagente sin experiencia previa, es más indicado que se le muestre lo antes posible cual es la composición del sistema, es decir, cuales son los agentes que integrarán el sistema; ya que le permite visualizar el sistema sin necesidad de pensar en su implementación (código).

- Las vistas del sistema propuestas por ambas metodologías son similares. Sin embargo, si pensamos en cuales muestran de mejor manera un mismo nivel de abstracción, tenemos que pensar en las propuestas por *PASSI*. Por ejemplo, para mostrar como es la comunicación entre los agentes en *PASSI*, se utiliza el concepto de ontología del dominio (concepto, predicado y acción) representado en el diagrama *Descripción de la Ontología del Dominio* (ver Figuras A.20a y A.20b), el cual establece el conocimiento del entorno que la sociedad de agentes deben poseer, es decir, establecer un mismo vocabulario para la comunicación.

Además, se utiliza también el concepto de ontología de comunicación (protocolos, lenguajes de comunicación y ontología del dominio) representado en el diagrama *Descripción de la Ontología de Comunicación* (ver Figuras A.21a y A.21b), el cual apunta a establecer cómo se llevará a cabo la comunicación. Cuales son los conocimientos (ontologías del dominio) que un agente debe poseer; ya que no necesariamente debe conocer la ontología completa; el tipo de lenguaje de comunicación para el intercambio de mensajes, y el protocolo a seguir. En cambio en *MASE*, la vista que muestra la comunicación entre agentes, diagrama *Clases de Agentes* (ver Figura 5.7), sólo identifica los agentes involucrados y los protocolos de interacción para las conversaciones que estos sostienen.

7.3 Referentes al trabajo futuro

Es necesario desarrollar otros casos de estudio bajo las metodologías evaluadas, de manera que sea posible ver su comportamiento en distintos dominios de aplicación y que su aplicación sirva como guía para el desarrollo de nuevos sistemas multiagentes.

Por otra parte, el desarrollo de este proyecto ha demostrado que la utilización de ambos frameworks de evaluación es necesaria para una comparación más completa, por esta razón queda pendiente tomar los mejores aspectos de ambos frameworks y realizarle mejoras. Por ejemplo, es necesario establecer claramente la necesidad de una previa aplicación de la metodología a evaluar a un caso de estudio; ya que es posible que un framework sea aplicado sin conocer a fondo la metodología. Además, es necesario también agregar más detalle a los criterios que evalúan los conceptos, el proceso de desarrollo y las vistas de una metodología, de manera que sea posible realizar comparaciones más profundas. Por ejemplo, la evaluación de los conceptos que utiliza la metodología se puede realizar a través de la evaluación de su meta-modelo, para lo cual es necesario establecer antes un meta-modelo genérico. Para la evaluación del criterio de proceso de desarrollo se pueden establecer para cada etapa, actividades que generalmente son realizadas en un proceso de desarrollo de un *MAS*, analizar si la metodología las incluye o no y analizar cómo lo hace o cómo las sustituye si corresponde. Por último, se debe especificar que la aplicación de los frameworks de evaluación sea iterativa, ya que de esta manera se puede disminuir el grado de subjetividad de esta técnica de comparación.

Capítulo 8

Referencias

[1] Wooldridge M.: Agents and Software Engineering, in AI*IA Notizie XI (3), pages 31-37, 1998.

[2] Wooldridge M. and Ciancarini P.: Agent-Oriented Software Engineering: The State of Art, in AI Springer-verlag Lecture Notes, vol. 1957, January 2001.

[3] Cuesta P., Gómez A., González J. and Rodríguez F.: A Framework for Evaluation of Agent-Oriented Methodologies, in Actas del Taller De Agentes Inteligentes en el Tercer Milenio (CAEPIA'2003), Noviembre 2003.

[4] Yu E., Cysneiros L.: Agent-Oriented Methodologies-Towards a Challenge Exemplar, 4th International Workshop on Agent-Oriented Information Systems (AOIS'02), May 2002.

[5] Cernuzzi L., Rossi G.: On the Evaluation of Agent Oriented Methodologies, in Proc. of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, November 2002.

[6] Kumar M.: Contrast and Comparison of Five Major Agent Oriented Software Engineering Methodologies (AOSE), http://students.jmc.ksu.edu/grad/madhukar/www/professional/aosepaper.pdf, 2002.

[7] Shehory O., Sturm A.: Evaluation of Modelling Techniques for Agent-Based Systems, Agents 2001, pages 624-631, 2001.

[8] Wooldridge M. and Jennings N.: Intelligent Agents: Theory and Practice, in The Knowledge Engineering Review, vol. 10(2), pages 115-152, 1995.

[9] Bussman S. and Müller H.: A Communication Architecture for Cooperating Agents, in Computational Artificial Intelligence, vol. 12(1), pages 37-54, 1993.

[10] Jennings R., Sycara K., Wooldridge M.: A Roadmap of Agent Research and Development, in Autonomous Agents and Multiagent Systems, vol. 1, pages 275-306, 1998.

[11] Henderson-Sellers B. and Giorgini P.: Agent-oriented Methodologies, Hershey, PA: Idea Group, 2005.

[12] Jacobson I., Booch G. and Rumbaugh J.: El Proceso Unificado de Desarrollo de Software, Addison Wesley, 1999.

[13] Tansley D. and Hayball C.: Knowledge Based Systems Analysis and Design a KADS developer's handbook, ed. Prentice Hall, 1993.

[14] Iglesias, C.: Definición de una Metodología para el Desarrollo de Sistemas Multi-Agente, Tesis Doctoral, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, 1998.

[15] Iglesias, C., Mercedes Garijo, M., Gonzalez, J., and Velasco, J.: Analysis and Design of Multiagent Systems using MASCommonKADS, in Intelligent Agents IV LNAI, vol. 1365, ed. SpringerVerlag, 1998.

[16] Pressman, R.: Software Engineering: A Practitioner's Approach, in Software Engineering and Technology, ed. McGraw-Hill, 1982.

[17] Caire G. et. al.: Agent Oriented Analysis using MESSAGE/UML, in Actas de Conferencia, Springer Verlag LNCS 2222, pages 119-135, 2001.

[18] Lyytinen K., Rossi M.: METAEDIT: A Fully Configurable Multi-User and Multi-tool CASE and CAME Environment, in Actas de Conferencia, Springer Verlag GNS 1080, 1999.

[19] Gómez J.: Modelado de Sistemas Multi-Agente, Tesis, Facultad de Informática Universidad Complutense, 2002.

[20] Bresciani P., Giorgini P., Giunchiglia F. and Mylopoulos J.: Tropos: An Agent Oriented Software Development Methodology, in Journal of Autonomous Agents and Multi-Agent Systems, vol.8, pages 203-236, 2004.

[21] Wooldridge M., Jennings N., and Kinny D.: The Gaia Methodology for Agent-Oriented Analysis and Design, in Journal of Autonomous Agents and Multi-Agent Systems, pages 285-312, 2000.

[22] DeLoach S.: Analysis and Design using MASE and Agentool, in 12th Midwest Artificial Intelligence and Cognitive Science Conferece (MAICS), 2001.

[23] Burrafato P., Cossentino M.: Designing a Multi-Agent Solution for a Bookstore with the PASI Methodology, 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), 2002.

[24] Cossentino M. and Potts C.: A Case Tool Supported Methodology for the Design of Multi-Agent Systems, in The 2002 International Conference on Software Engineering Research and Practice (SERP'02), 2002.

[25] Antón A., and Potts C.: The Use of Goals to Surface Requirements for Evolving Systems, in International Conference on Software Engineering (ICSE '98), pages 157-166, 1998.

[26] DeLoach S.: Multiagent Systems Engineering of Organization-based Multiagent Systems, 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05), 2005.

[27] Potts, C.: ScenIC: A Strategy for Inquiry-Driven Requirements Determination, in IEEE Fourth International Symposium on Requirements Engineering (RE'99), 1999.

[28] Foundation for Intelligent Physical Agents (FIPA): FIPA Abstract Architecture Specification (Refinements), Document FIPA PC00094, http://www.fipa.org/specs/ fipa00094/PC00094.html, 2001.

[29] Siau K., and Rossi M.: Evaluation of Information Modeling Methods – A Review, in 31 Annual Hawaii International Conference on System Science, pages 314-322, 1998.

[30] Peña J.: Guidelines, Techniques and Modelling Artefacts at the Analysis Stage of AOSE Methodologies to Deal with complexity, in AOSE Technical Forum Group, 2004.

[31] Sturm A., and Sheory O.: A Framework for Evaluating Agent-Oriented Methodologies, in Workshop on Agent-Oriented Information System (AOIS), 2003.

[32] Cox M., Kerkez B., Srinivas C., Edwin G., and Archer W.: Toward Agent-Based Mixed-Initiative Interfaces, in The 2000 International Conference on Artificial Intelligence, 2000.

[33] Espen Andersen (Norwegian School of Management): Juul Møller Bokhandel A/S, http://www.espen.com/papers/jme.pdf, 1997.

[34] Corradini F., Merelli E., and Vita M.: Cell-MAS: ATP Production from Carbohydrate Oxydation, in Computational Science and Its Applications – ICCSA 2005, vol. 3481/2005, 2005.

[35] Burrafato P., Cossentino M.: Designing a Multi-Agent Solution for a Bookstore with the PASI Methodology, in The Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), 2002.

[36] Chella A., Cossentino M., Sabatucci L. and Seidita V.: Agil PASSI: An Agil Process for Designing Agents, in The International Journal of Computer Systems Science & Engineering, Special issue on "Software Engineering for Multi-Agent Systems", 2006.

[37] Wood M.: Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems, in Thesis Air Force Institute of Technology, AFIT/GCS/ENG/00M-26, 2000.

[38] Dileo J., Jacobs T., and DeLoach S.: Integrating Ontologies Into Multiagent Systems Engineering, in Proceedings of Agent Oriented Information Systems (AOIS - 2002), 2002.

[39] Self A. and DeLoach S.: Designing and Specifying Mobility Within The Multiagente Systems Engineering Methodology, in Special Track on Agents, Interactions, Mobility, and Systems (AIMS) at The 18th ACM Symposium on Applied Computing (SAC 2003), 2003.

Anexo

Modelado del sistema DRT

En este capítulo se presentan los diagramas generados al realizar el modelado del *sistema DRT* bajo las metodologías orientadas a agentes *PASSI* y *MASE*.

A.1 Modelado del sistema con PASSI

A.1.1 Modelo de requerimientos del sistema

Descripción del dominio

El primer paso durante esta fase es crear el *Diagrama de Contexto*. Este diagrama no está incluido dentro del proceso de *PASSI*, sin embargo debe ser creado cuando se utiliza el *PTK* para generar los diagramas. El diagrama de contexto permite identificar y mostrar los actores que deberán interactuar con el sistema. Estos pueden ser personas, otros sistemas, o bien, otros agentes externos a los involucrados en el desarrollo. La Figura A.1 muestra el *Diagrama de contexto* para el *Sistema de transporte de pasajeros (DRTS)*.

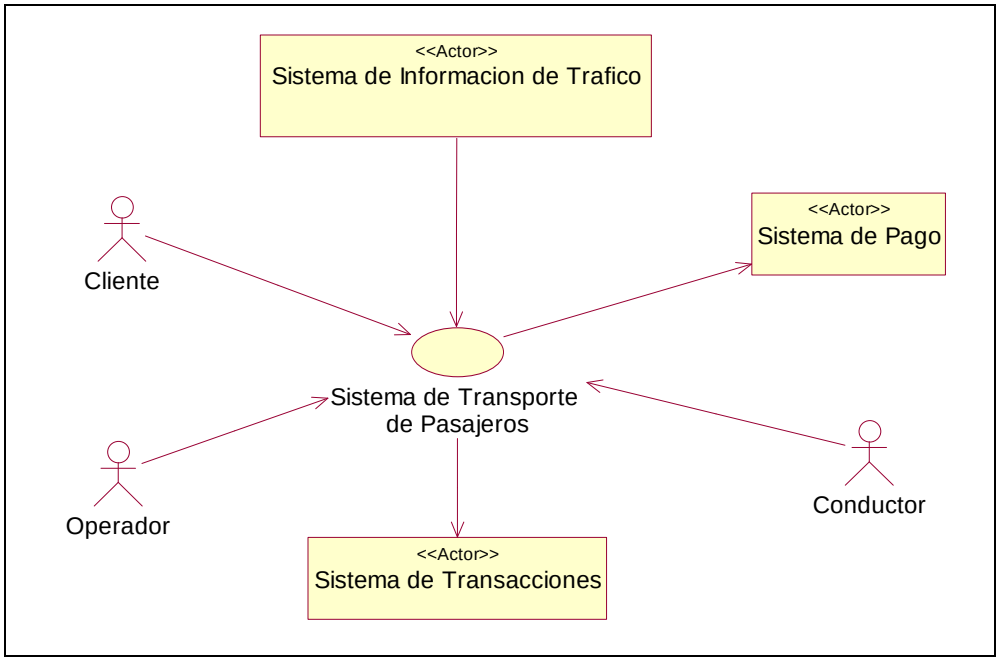

Figura A.1 Diagrama de contexto del sistema de transporte de pasajeros.

Como se aprecia en la Figura A.1, el sistema interactúa con seis actores: cliente, conductor, operador, sistema de información de tráfico, sistema de pago y sistema de transacciones. De acuerdo a los requerimientos funcionales del sistema (ver sección 4.3.2), podemos indicar lo siguiente:

- **Cliente:** Representa a los usuarios que solicitan el servicio de transporte. Estos usuarios deberán estar conectados en forma permanente con el sistema con la finalidad de ser oportunamente informados de cualquier cambio o imprevisto que se presente en la planificación del viaje, como por ejemplo atrasos, fallas de los vehículos, cambios relevantes en las condiciones de tráfico, entre otros.

- **Conductor:** Representa a los individuos que conducen los distintos tipos de vehículos que prestan servicio. Deberán estar conectados en forma permanente con el sistema de manera que puedan comunicar a la empresa sobre eventualidades, como por ejemplo, transportar un nuevo cliente, atrasos, cancelación de viajes, cambios relevantes en las condiciones del tráfico, entre otros.

- **Operador:** Representa al individuo que trabaja en las unidades de despacho de los vehículos. Es el encargado de registrar y cancelar los vehículos cuando éstos inician y terminan su horario de servicio.

- **Sistema de información de tráfico:** Representa al sistema encargado de proveer a los usuarios del sistema la información sobre el estado de las rutas (congestiones y desvíos) y la zona geográfica cubierta. La información proporcionada por el sistema puede ser localización de direcciones y paradas, nombres de calles, distancias entre localizaciones y su tiempo de traslado correspondiente.

- **Sistema de pago:** Representa al sistema encargado de gestionar el proceso de pago de los servicios de transportes. Para esto recibirá del sistema *DRT* información sobre: la descripción del servicio entregado (detalle del viaje y tarifas), los clientes (datos personales), la modalidad de pago (pago a bordo o de antemano) y las opciones (tarjeta de crédito, efectivo y boleto).

- **Sistema de transacciones:** Representa al sistema responsable de almacenar y gestionar esta información. Para esto recibirá del sistema *DRT* información asociada a las transacciones (tipo y costo de viaje, nombre y dirección del cliente, operador y conductor del vehículo, etc.)

Una vez que se ha creado este diagrama, se puede comenzar con la creación del *Diagrama de Descripción de Dominio (D.D)* (ver Figura A.2). Este es el primer diagrama dentro de la etapa de modelado de requerimientos de *PASSI*. En él, mediante casos de uso, se permite identificar y mostrar en términos generales las funcionalidades básicas que tendrá el sistema sin entrar en detalle respecto a cómo serán distribuidas entre los distintos agentes.

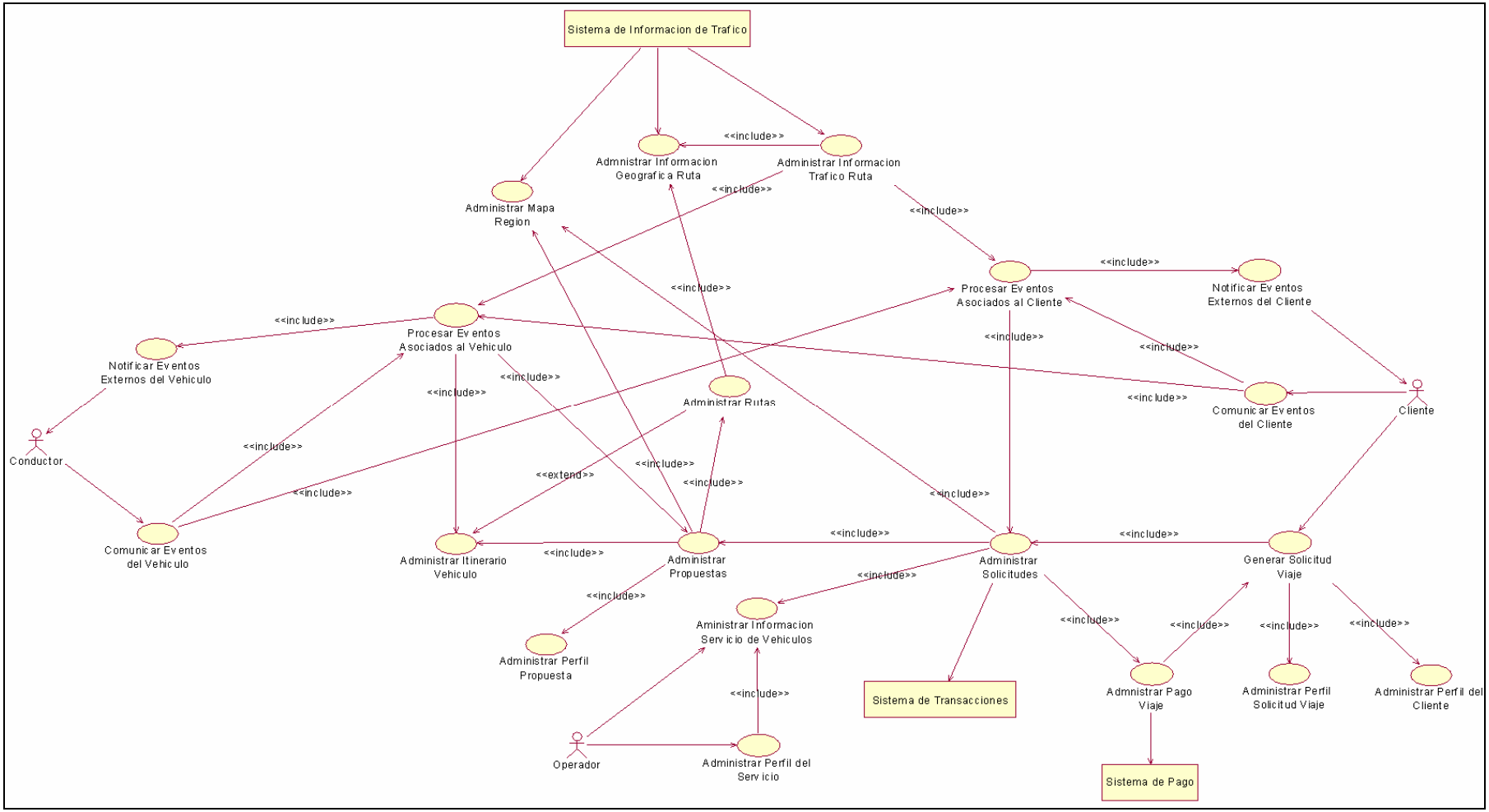

Figura A.2 Diagrama de descripción del dominio.

Para alcanzar una mayor comprensión del modelado del sistema, a continuación se describirán algunas de las funcionalidades identificadas:

- **Administrar mapa región:** Mantiene una mapa geográfico de la región en la que los vehículos circulan. Los clientes pueden consultar el mapa para conocer las paradas o puntos de detención que cada vehículo ofrece.

- **Administrar información geográfica ruta:** Procesa la información entregada por el *Sistema de Información de Tráfico* acerca de las actuales distancias entre un par de puntos y su respectiva localización (dirección). Además, mantiene una matriz con las distancias (en kilómetros) y el tiempo de traslado de un punto a otro. Esta información la utiliza para calcular la ruta más corta entre un par de puntos cuando un par de ellos esté congestionado y/o se produzca un desvió, y cuando se entregue una propuesta de servicio.

- **Administrar información tráfico ruta:** Procesa información de las rutas (congestión y desvíos) para mantener informados a los vehículos y clientes sobre el estado del servicio de transporte.

- **Administrar información servicio de vehículos:** Registra y cancela los servicios que ofrece un vehículo, recibe consultas acerca sobre los servicios de los vehículos y genera una lista con los vehículos que encajan con un determinado perfil de servicio

- **Administrar solicitudes:** Procesa las solicitudes de los servicios de transporte, pide una lista con los vehículos que puedan satisfacer el servicio solicitado por el cliente, realiza llamados a propuestas a los vehículos que encajan con el servicio solicitado, recibe y evalúa propuestas de servicios en respuesta al llamado realizado y administra el proceso de pago de servicios.

- **Administrar propuestas:** Recibe información acerca del servicio demandado (lugar y hora de recogida/entrega y cantidad de pasajeros a transportar), revisa el itinerario de los vehículos para ver si el vehículo está disponible en un horario determinado, valida si es posible llevar una cantidad determinada de pasajeros, valida si los lugares de recogida y entrega son factibles de incluir en la ruta planificada y valida si es posible transportar exclusivamente al cliente en caso de que lo quiera. Además, solicita propuestas de nuevas rutas de viaje (añadir lugares de recogida y entrega a la ruta actual de un vehículo), verifica las condiciones de tráfico en la ruta, consulta el mapa para conocer los puntos de detención y calcula el valor del viaje. Una vez que posee toda la información necesaria, evalúa la conveniencia de responder a la solicitud de viaje. Si se adjudica una propuesta, informa a *Administrar Itinerario* sobre la nueva demanda y entrega información sobre esta (nueva ruta y datos de la demanda).

- **Administrar itinerario:** Recibe desde *Procesar Eventos Asociados al Vehículo* información sobre eventos externos del vehículo como por ejemplo cancelación de viaje por parte del cliente, atraso del cliente, o congestión/desvío en la ruta. Por otra parte, recibe desde *Administrar Propuestas*: la nueva ruta del vehículo, información sobre el servicio demandado e información sobre el cliente. Además, organiza las demandas, calcula cuantos viajes debe realizar el vehículo en base a las demandas, posee la ruta de cada viaje y la actualiza una vez que ya ha visitado un punto.

- **Administrar rutas:** Crea rutas de viaje, es decir, la ruta más corta entre los diversos puntos que un vehículo debe visitar. Para esto consulta a *Administrar Mapa Región* sobre la ruta más corta entre dos puntos.

Identificación de agentes

Esta fase permite la creación del segundo diagrama dentro de la etapa de modelo de requerimientos del sistema. En él se agrupan las funcionalidades obtenidas del diagrama anterior y son asignadas a un determinado agente, permitiendo establecer los agentes que integrarán el sistema y las funcionalidades asociadas a cada uno de ellos. Cabe destacar, que las relaciones entre casos de uso internos de un agente mantienen sus estereotipos originales, tales como <<*extend*>> o <<*include*>>, pero las relaciones entre casos de uso pertenecientes a distintos agentes cambian al estereotipo <<*communicate*>>, por considerarse la forma natural en la cual los agentes interactúan.

La Figura A.3 muestra el *Diagrama de Identificación de Agentes* resultante de la fase *Identificación de Agentes*. En él podemos ver un sistema compuesto por seis tipos de agentes, a saber: *GUI Cliente, GUI Vehículo, Vehículo, Cliente, Map* y *Broker*. Los agentes *GUI Cliente* y *GUI Vehículo* son los encargados de proveer una completa comunicación e interoperabilidad entre los clientes/conductores y el sistema de transporte. El agente *Vehículo* por su parte, esta a cargo de asignar rutas a los vehículos, de programar sus viajes y de crear propuestas de servicio; mientras que el agente *Cliente* gestiona el proceso de solicitud del servicio y selecciona las propuestas de servicio más convenientes para el cliente. El agente *Map* modela la actual región geográfica bajo cobertura y tiene el papel de proveer al resto de los agentes cualquier información relacionada a esta, y por último, el agente *Broker* es el encargado de conocer los servicios de transporte disponibles y sus características.

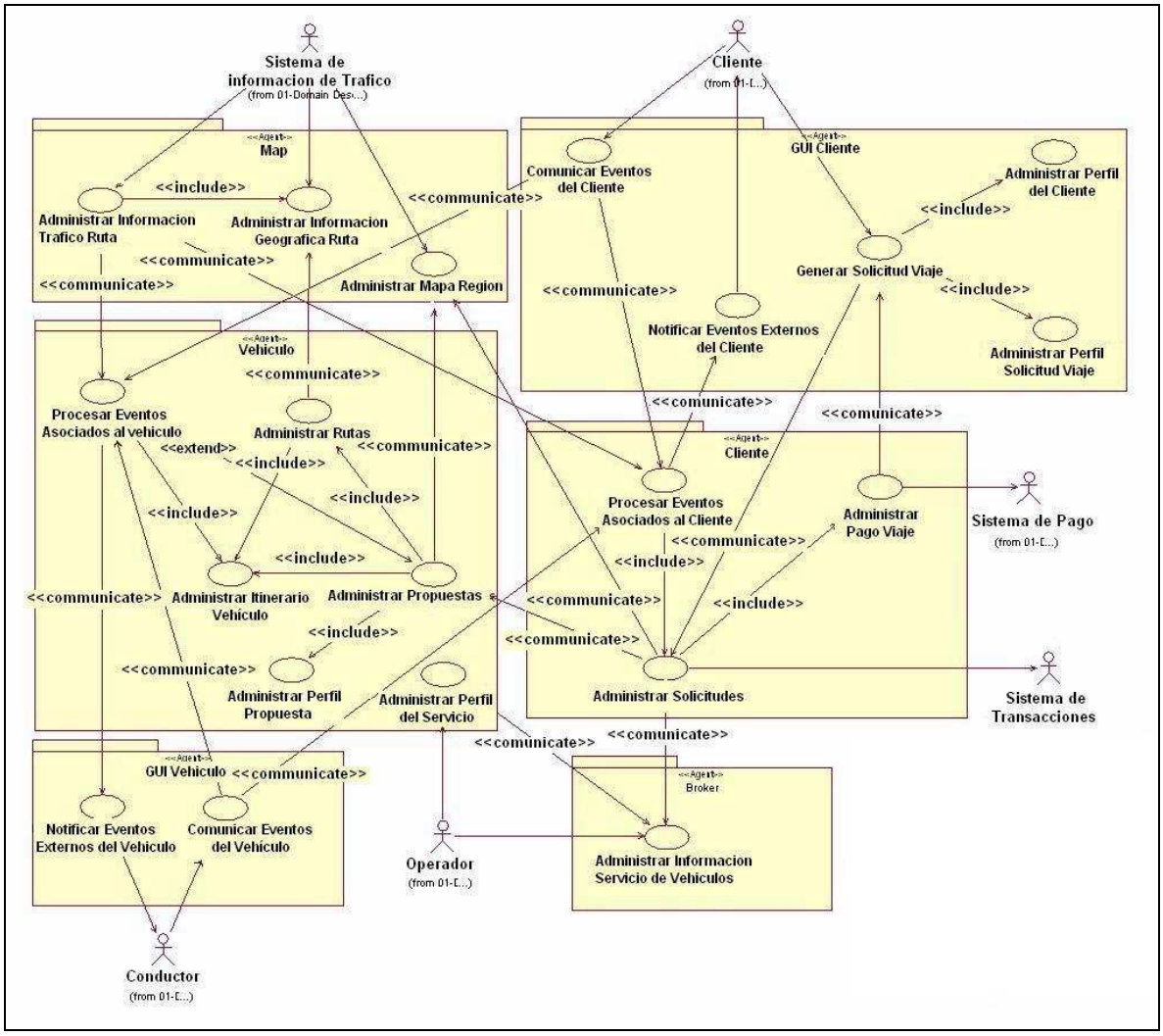

Figura A.3 Diagrama de identificación de agentes.

Identificación de roles

Esta fase permite la creación del tercer diagrama dentro del modelado de requerimientos de *PASSI*. En él se representan los distintos roles que un agente puede tomar y las comunicaciones entre ellos, considerando que eventualmente durante una misma interacción un agente puede tomar varios roles. Para la representación de las comunicaciones se utilizan diagramas de secuencia de UML con ciertas variaciones, las que se refieren a la forma de nombrar los objetos participantes, ya que, en este caso se nombran de la forma <Nombre Rol>:<Nombre Agente>.

A continuación se muestra la descripción de los escenarios más representativos del sistema.

- **R.Id 1 El cliente realiza una solicitud de viaje:** El Cliente solicita un viaje mediante una consulta previa de los vehículos en servicio en ese momento y de la cobertura geográfica de la empresa de transporte. Una vez elegido el tipo vehiculo según las características de su perfil y los puntos de inicio y destino de su viaje, se crea la solicitud para realizar el viaje, la cual deberá también incluir el grado de importancia que asigna el cliente a las características del vehículo (incluidas en el perfil del vehículo), además de la hora y la fecha del viaje, y su propio perfil como cliente. Como respuesta a lo solicitado, el cliente recibe un listado con las propuestas para la realización de su viaje (las que incluyen entre otros datos la tarifa que se cobrará por el servicio). Luego, el cliente elige una de las alternativas y formaliza la solicitud, recibiendo la correspondiente confirmación de la operación si se ha programado satisfactoriamente esa opción en el itinerario del vehículo correspondiente. Las Figuras A.4a, A.4b y A.4c muestran el diagrama de este escenario.

- **R.Id 2 El cliente comunica la cancelación o retraso del viaje:** El cliente informa un atraso o la cancelación de su viaje, para lo cual debe elegir de su listado de viajes pendientes aquel viaje atrasado o a cancelar, y luego enviar un aviso sobre esta situación. En el itinerario del vehículo se registra el atraso o se cancela el viaje y se notifica dicha acción al conductor del vehículo y al agente *Vehículo*. La Figura A.5 muestra el diagrama de este escenario.

- **R.Id 3 El conductor comunica la cancelación, retraso o falla del vehículo:** El conductor del vehículo se ha retrasado o ha cancelado su servicio o ha sufrido alguna falla del vehículo. En caso de ocurrencia de los eventos falla o cancelación del servicio envía el correspondiente aviso para cancelar su perfil de servicio, recibiendo la respectiva confirmación de la operación. Luego envía la cancelación de los viajes programados en su itinerario actual, las cuales deberán ser posteriormente comunicadas a los clientes afectados. Para el caso de retraso del vehículo, el conductor envía el correspondiente aviso, recibiendo la respectiva confirmación de la operación. El sistema evalúa las solicitudes afectadas y eventualmente podría sugerir ya sea una nueva solicitud o una solución alternativa (por ejemplo una nueva ruta). La Figura A.6 muestra el diagrama de este escenario.

- **R.Id 4 El sistema de información de tráfico comunica la ocurrencia de un evento en la ruta:** Una vez procesados los eventos, que pueden ser congestión o desvío de la ruta de viaje, son enviados a los vehículos los puntos de cobertura de la empresa de transporte que se ven afectados, junto con el margen de tiempo que esos puntos permanecerán bloqueados. Además, el conductor debe recibir los cambios que esos eventos hayan producido en su itinerario actual, los cuales también deberán ser informados a los clientes respectivos. Las Figuras A.7a, A.7b, A.7c y A.7d muestran el diagrama de este escenario.

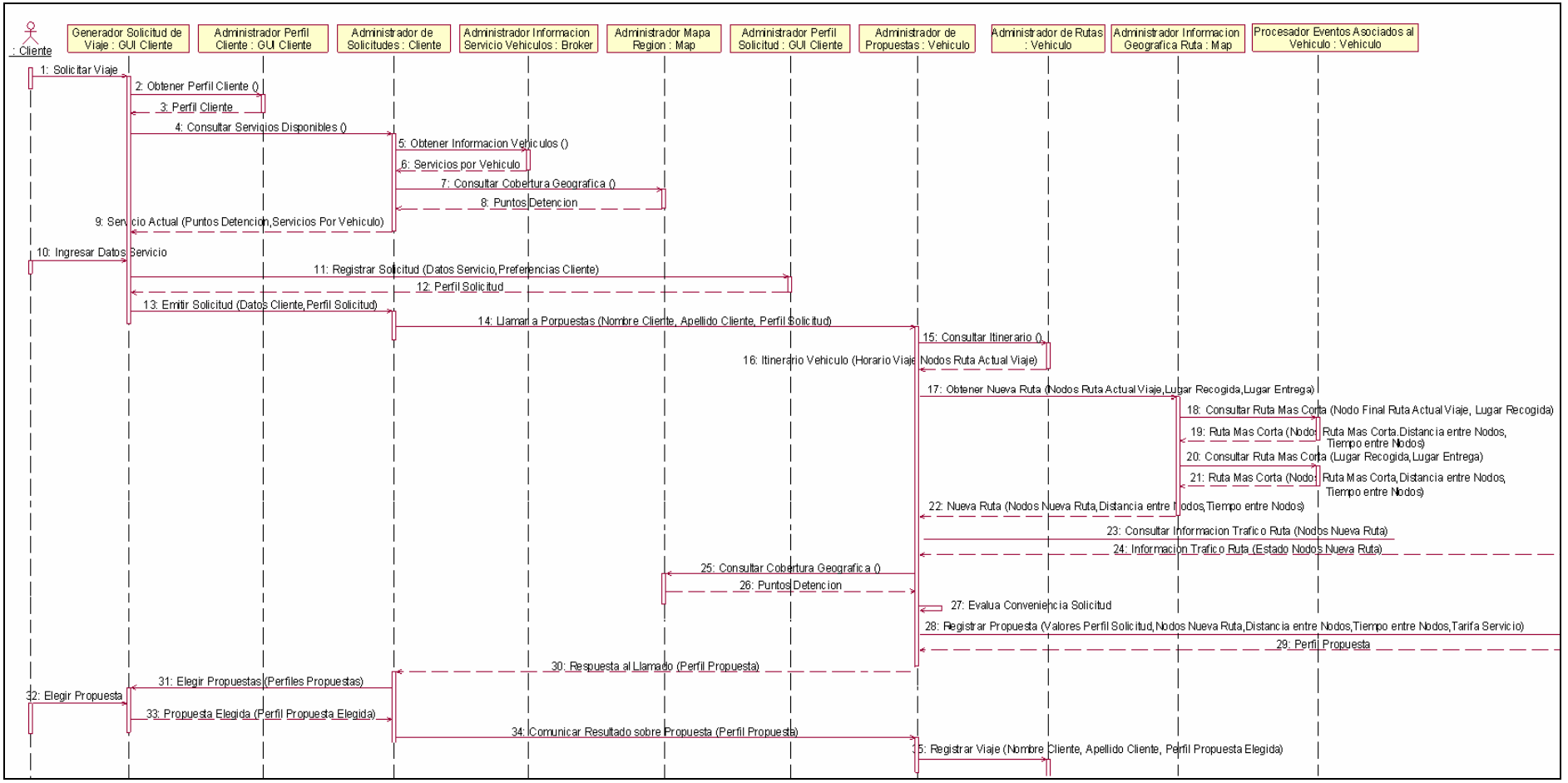

Figura A.4a R.Id 1, cliente realiza una solicitud de viaje (parte1).

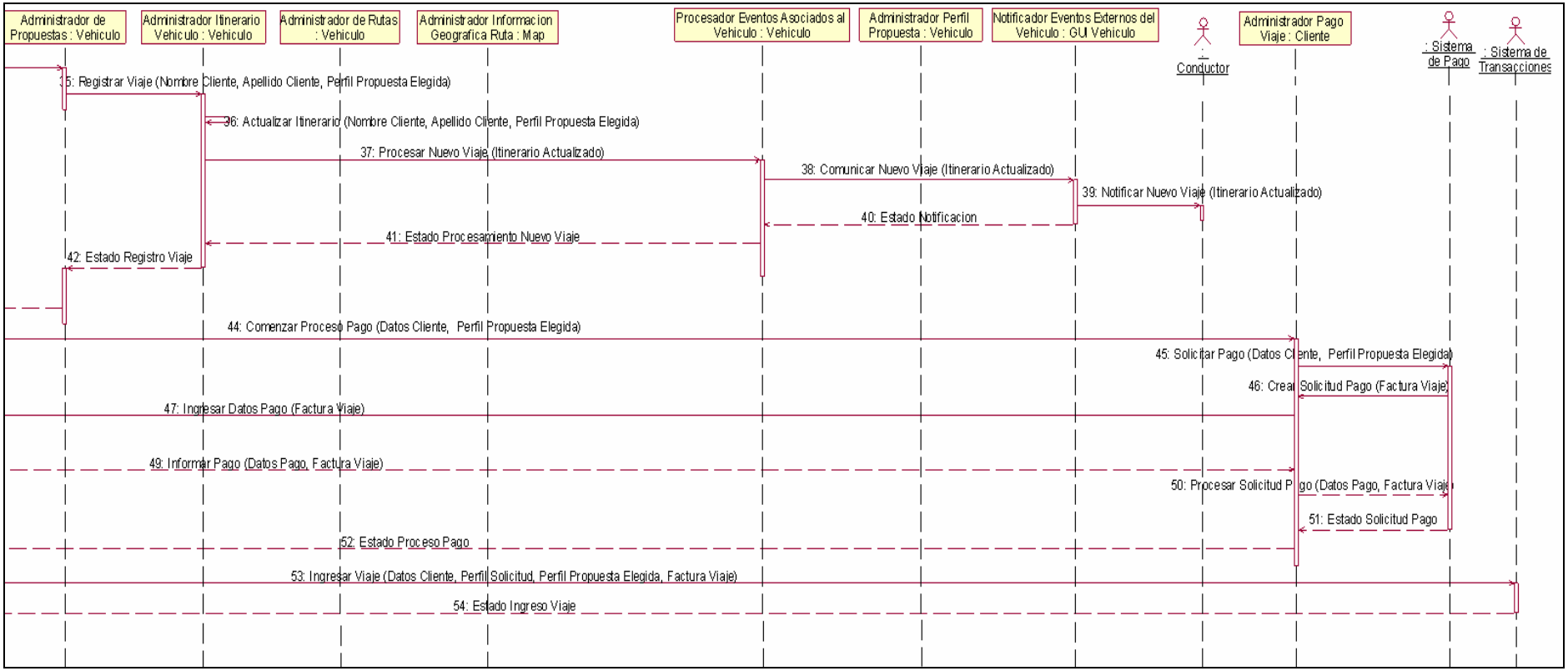

Figura A.4b R.Id 1, cliente realiza una solicitud de viaje (parte 2).

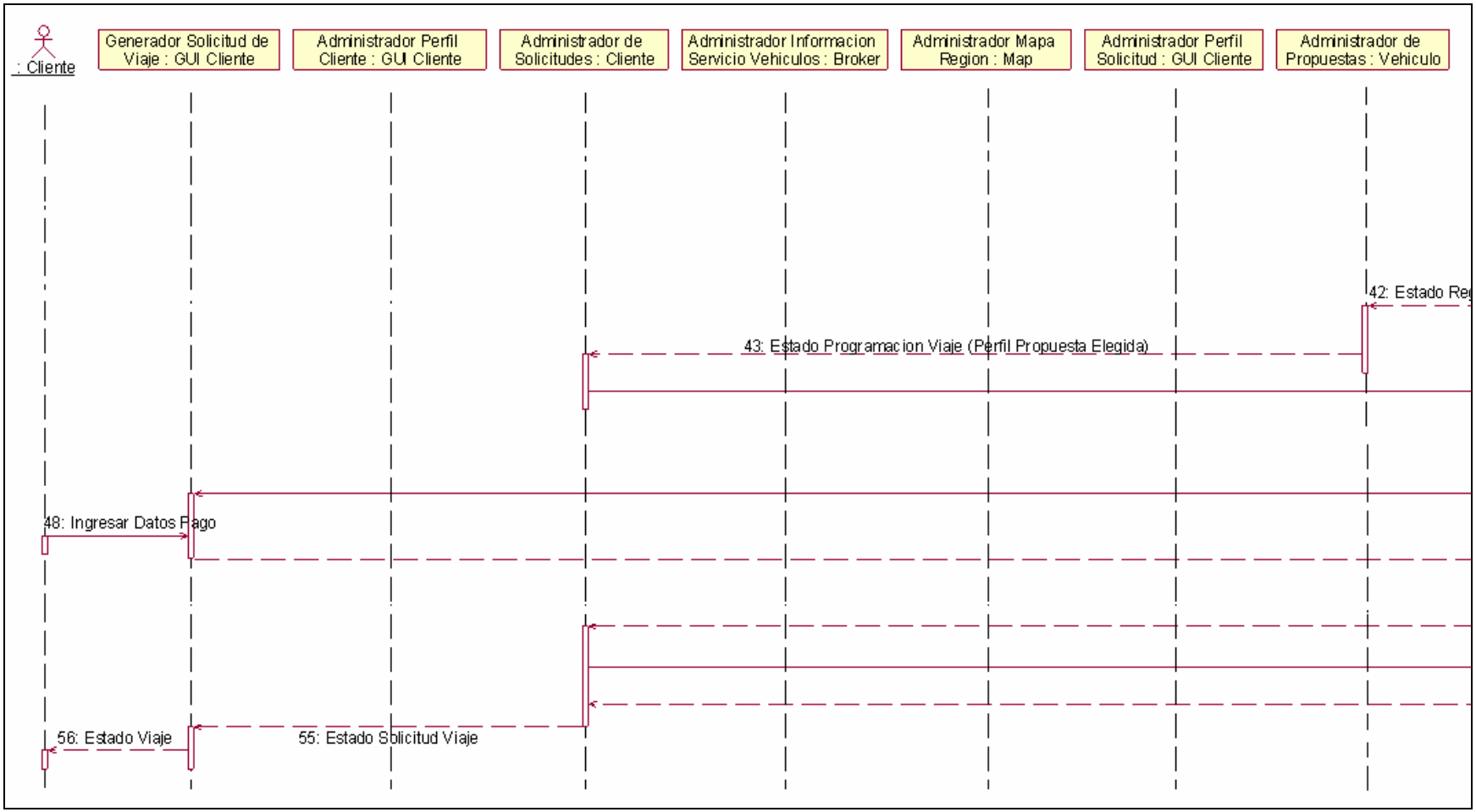

Figura A.4c R.Id 1, cliente realiza una solicitud de viaje (parte 3).

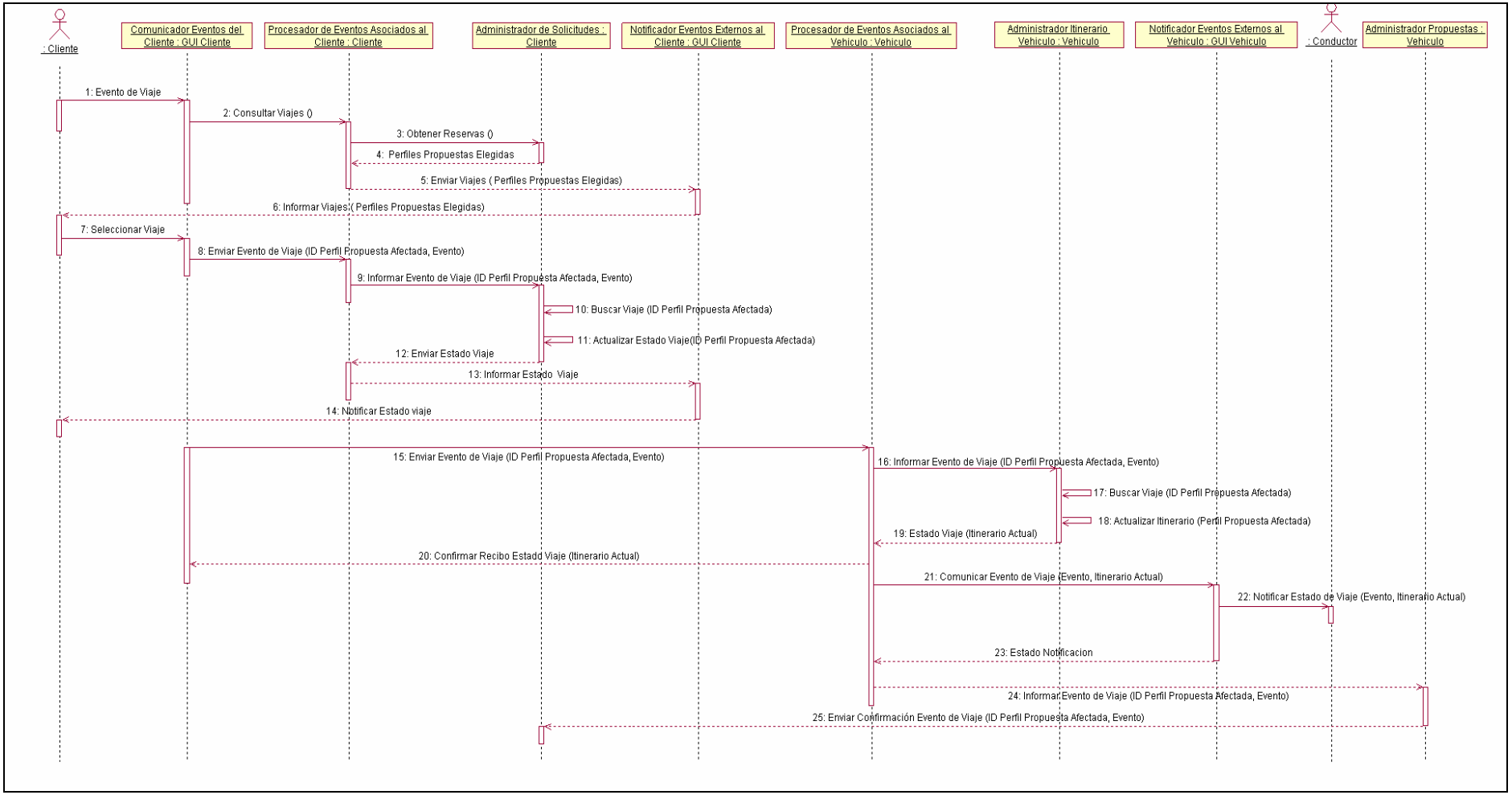

Figura A.5 R.Id 2, cliente comunica cancelación o retraso del viaje.

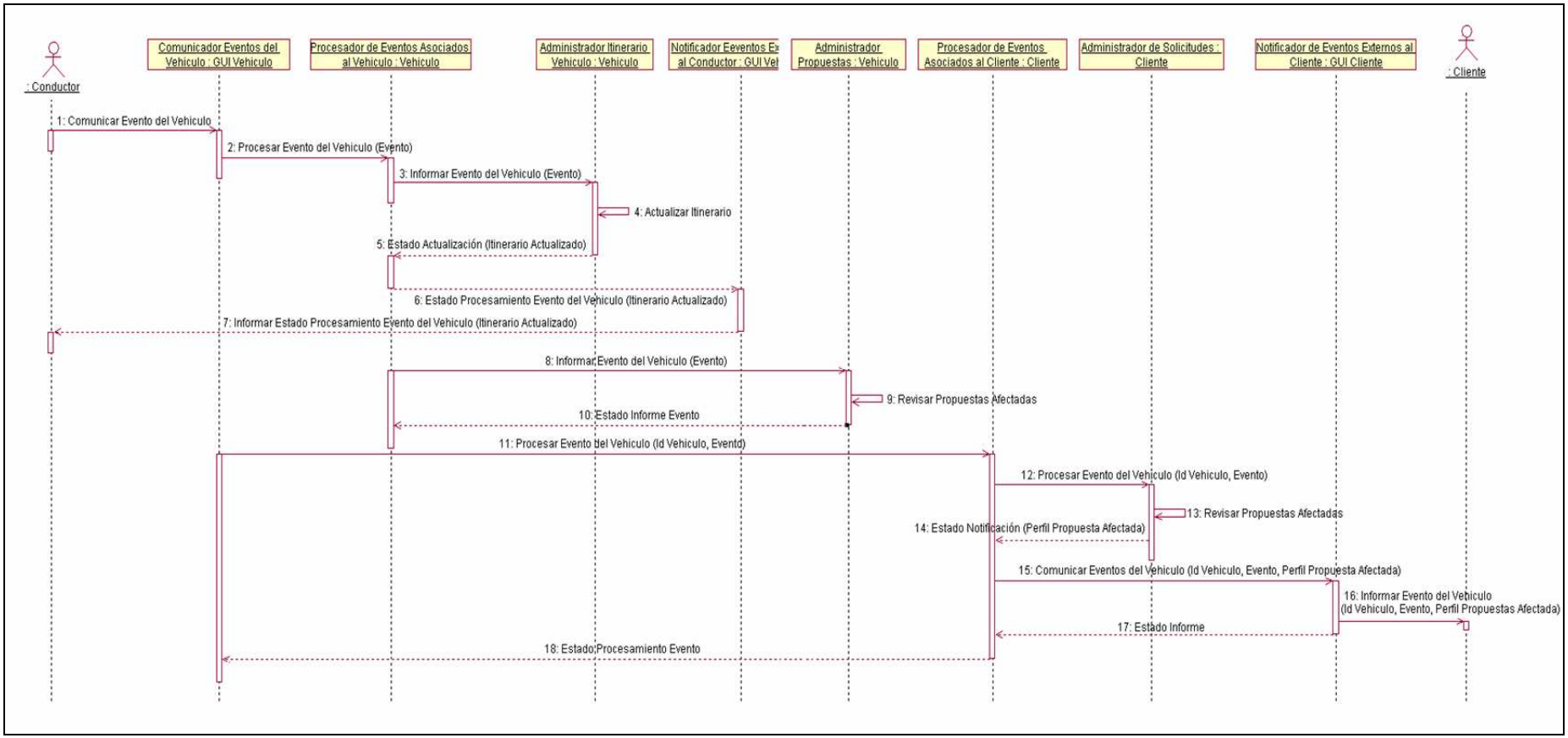

Figura A.6 R.Id 3, conductor comunica cancelación, retraso o falla del vehículo.

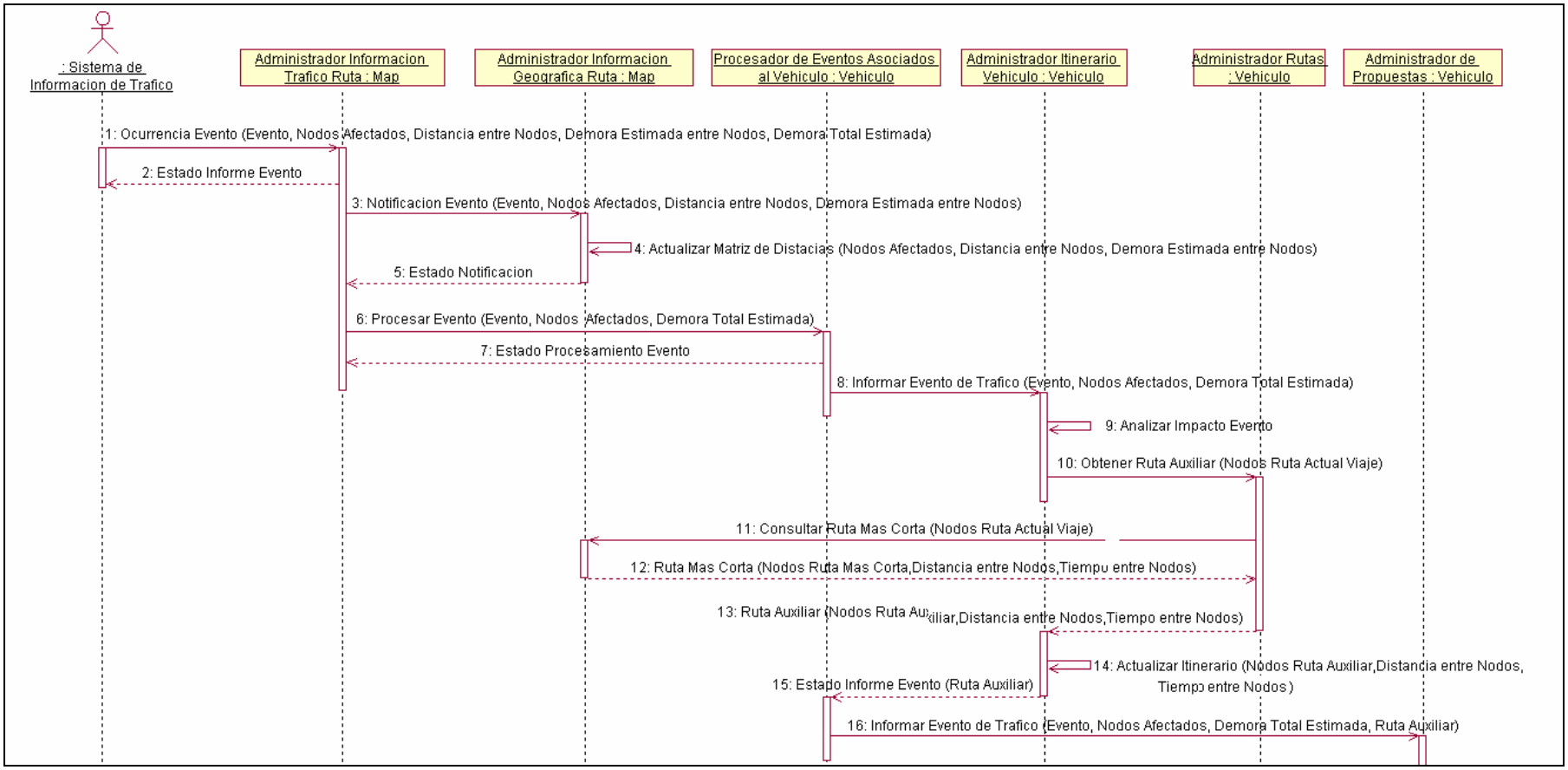

Figura A.7a R.Id 4, sistema de información de tráfico comunica evento en la ruta (parte 1).

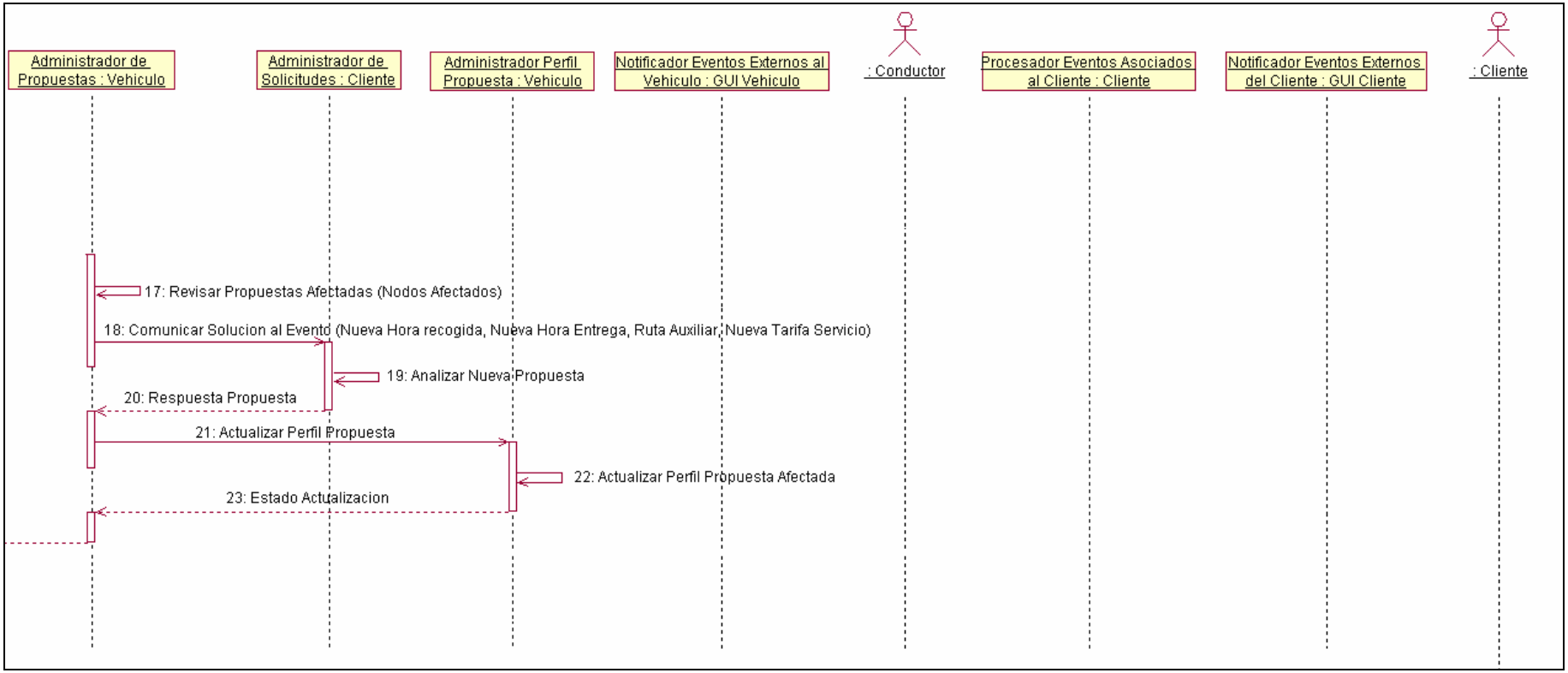

Figura A.7b R.Id 4, sistema de información de tráfico comunica evento en la ruta (parte 2).

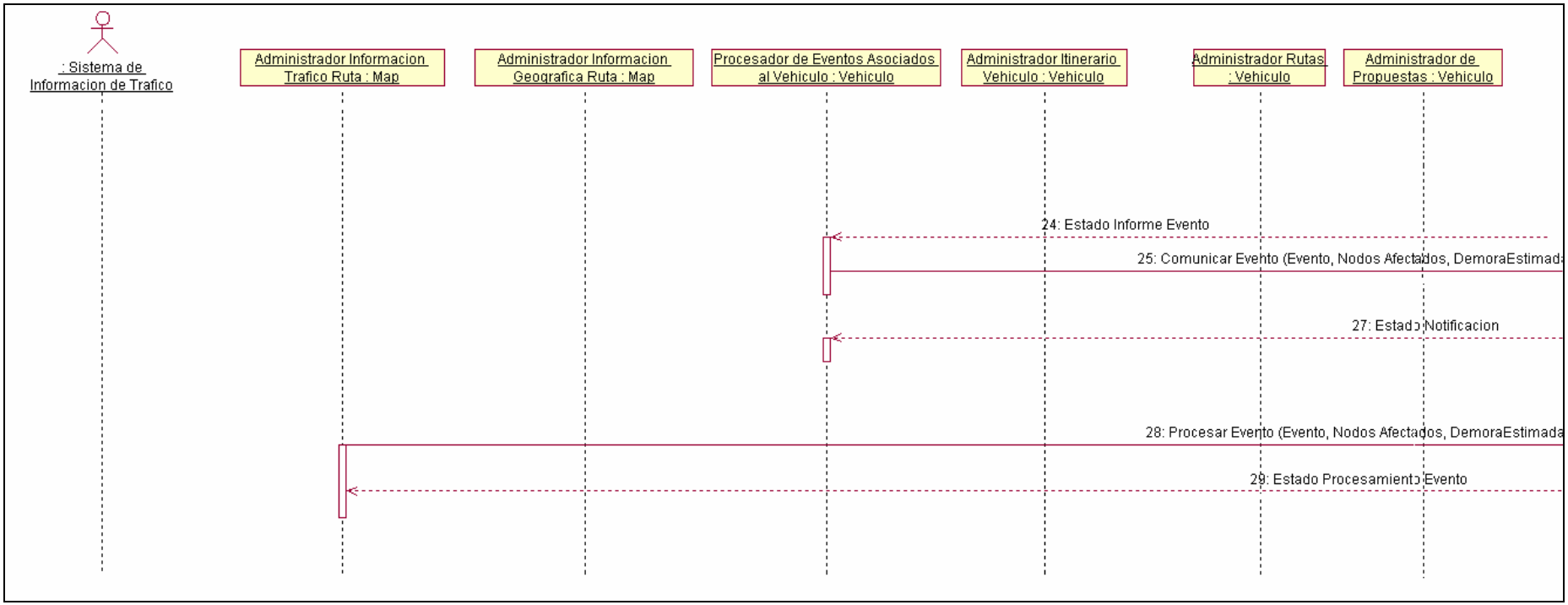

Figura A.7c R.Id 4, sistema de información de tráfico comunica evento en la ruta (parte 3).

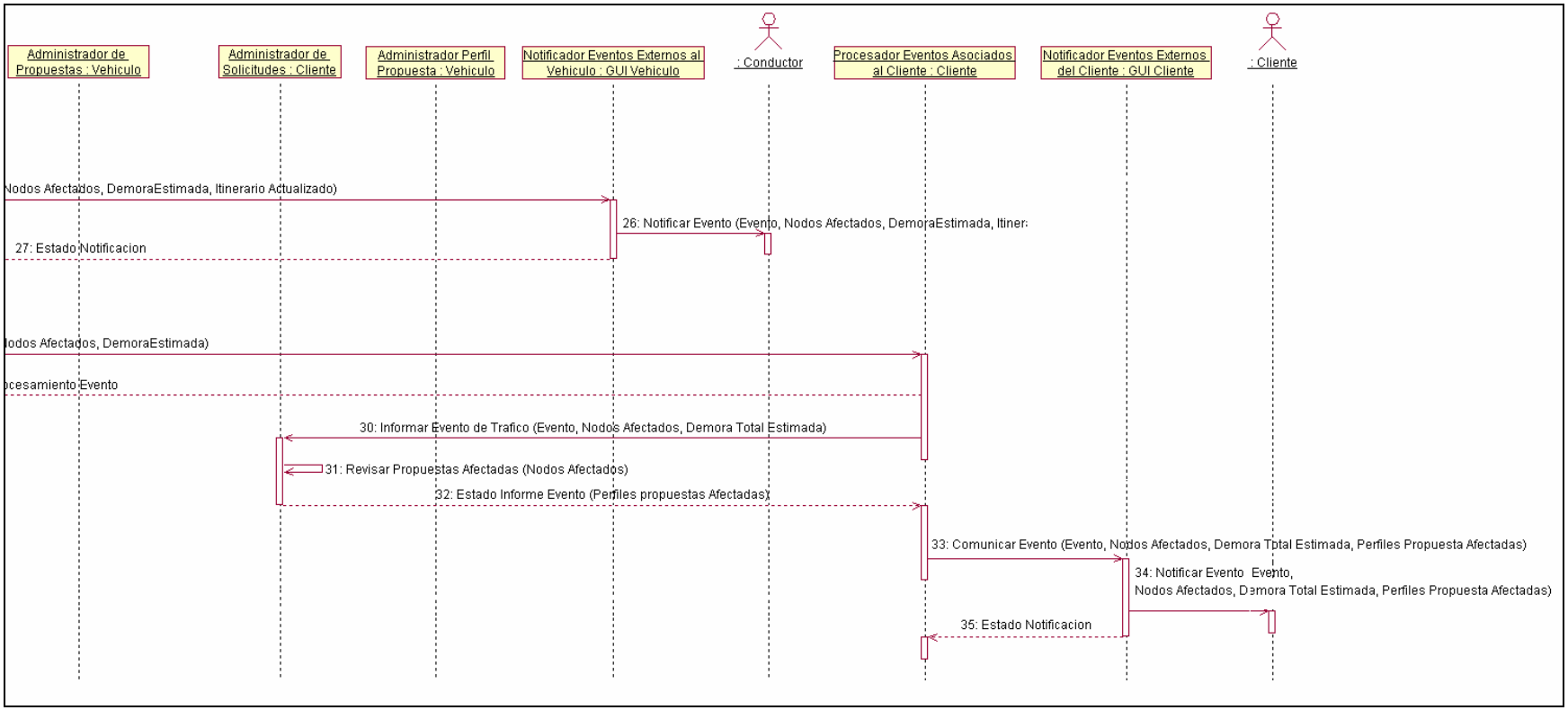

Figura A.7d R.Id 4, sistema de información de tráfico comunica evento en la ruta (parte 4).

- **R.Id 5 Sistema de información de tráfico envía datos de la región geográfica:** El sistema de transporte recibe desde el sistema de información de tráfico los siguientes datos: los puntos de detención, las posiciones donde se encuentran estos puntos y las distancias (en km. y unidad de tiempo) entre puntos. Estos datos son utilizados por el sistema *DRT* para crear una matriz de distancias (la cual le permitirá calcular la ruta más corta entre puntos) y para crear el mapa geográfico de la región. La Figura A.8 muestra el diagrama de este escenario.

- **R.Id 6 Operador inicia el servicio de un vehículo:** El Operador registra el servicio de un vehículo para que este aparezca en el sistema como disponible, y así pueda atender a las diversas solicitudes que lo requieran. La Figura A.9 muestra el diagrama de este escenario.

- **R.Id 7 Operador termina el servicio de un vehículo:** El Operador termina el servicio de un vehículo para que este no aparezca en el sistema como disponible. La Figura A.10 muestra el diagrama de este escenario.

- **R.Id 8 Operador crea el perfil de servicio de un vehículo:** El Operador crea el perfil de servicio de un vehículo, el cual contiene toda la información referente al servicio entregado por el vehículo (tiempo de recogida, tiempo de entrega, dirección de recogida, dirección de entrega, lugares requeridos, tiempo servicio de recogida, tiempo servicio de entrega y máximo tiempo de viaje) y a su respectiva función de utilidad (tiempo total de espera entrega, tiempo total de ocio del vehículo, tiempo total de exceso de viaje y tiempo total de viaje del vehículo). La Figura A.11 muestra el diagrama de este escenario.

- **R.Id 9 Operador modifica el perfil de servicio de un vehículo:** El Operador modifica el perfil de servicio de un vehículo. La Figura A.12 muestra el diagrama de este escenario.

- **R.Id 10 Operador modifica el perfil de servicio de un vehículo:** El Operador elimina el perfil de servicio de un vehículo. La Figura A.13 muestra el diagrama de este escenario.

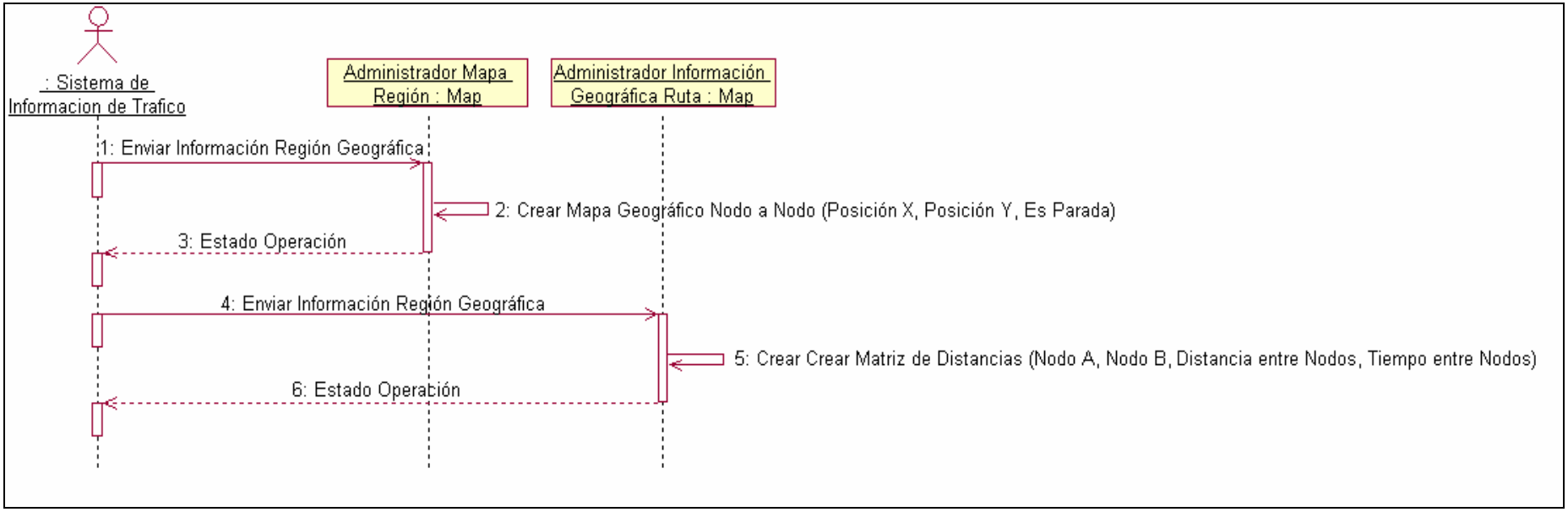

Figura A.8 R.Id 5, sistema de información de tráfico envía datos de la región geográfica.

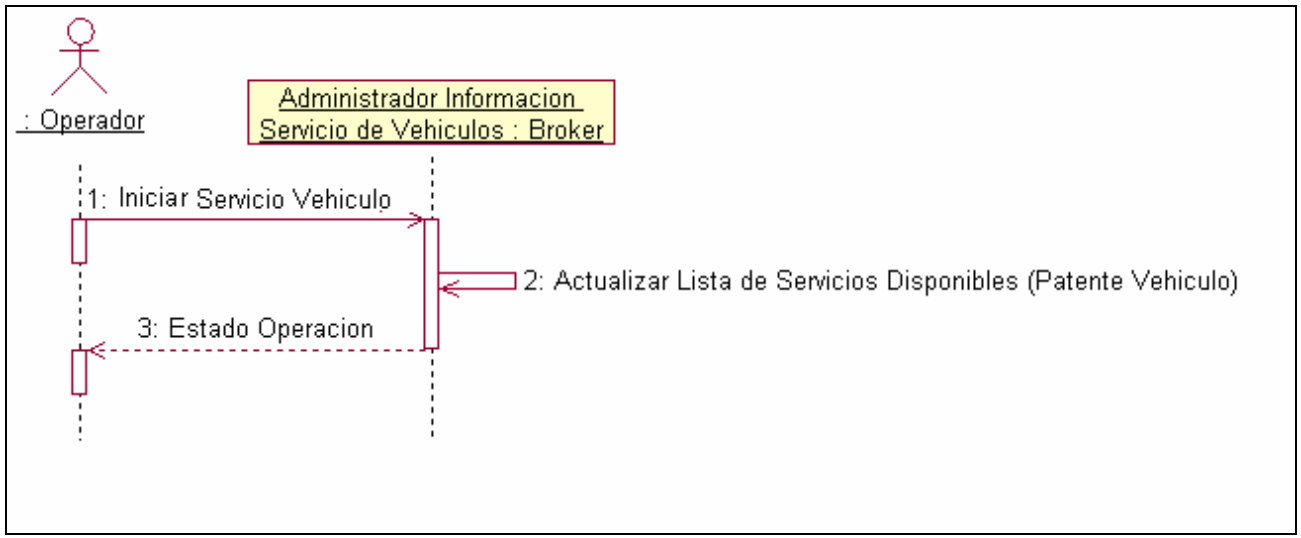

Figura A.9 R.Id 6, operador inicia el servicio de un vehículo.

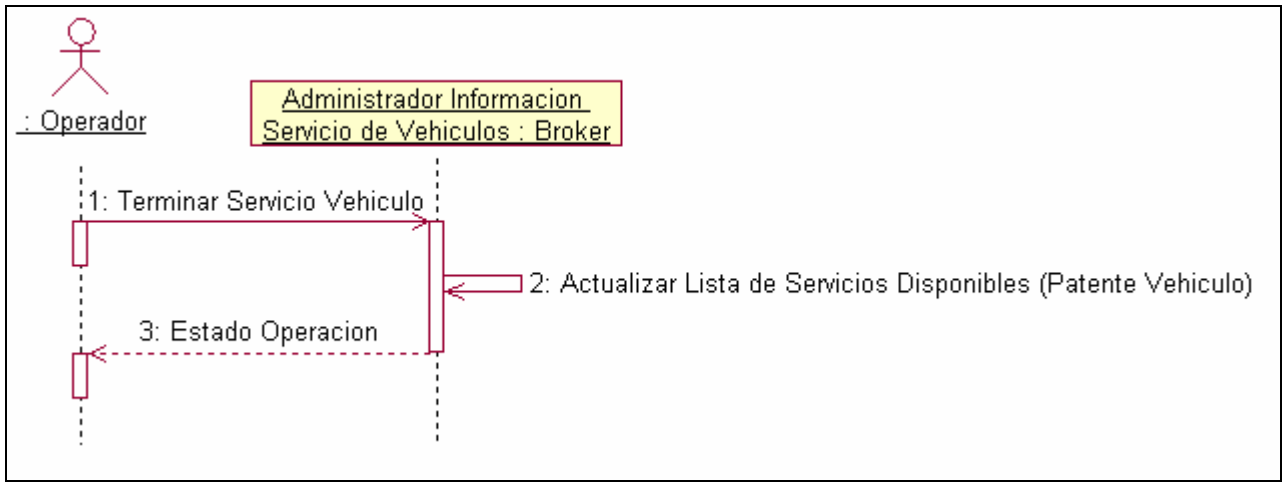

Figura A.10 R.Id 7, operador termina el servicio de un vehículo.

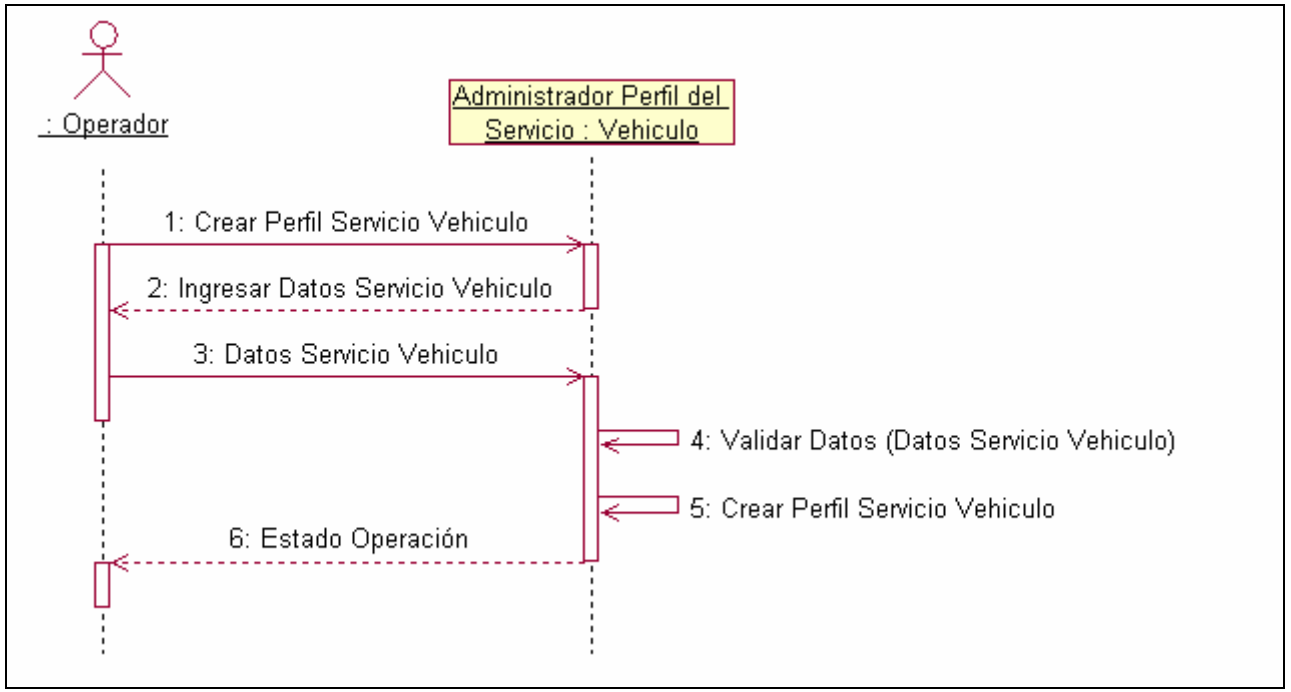

Figura A.11 R.Id 8, operador crea el perfil de servicio de un vehículo.

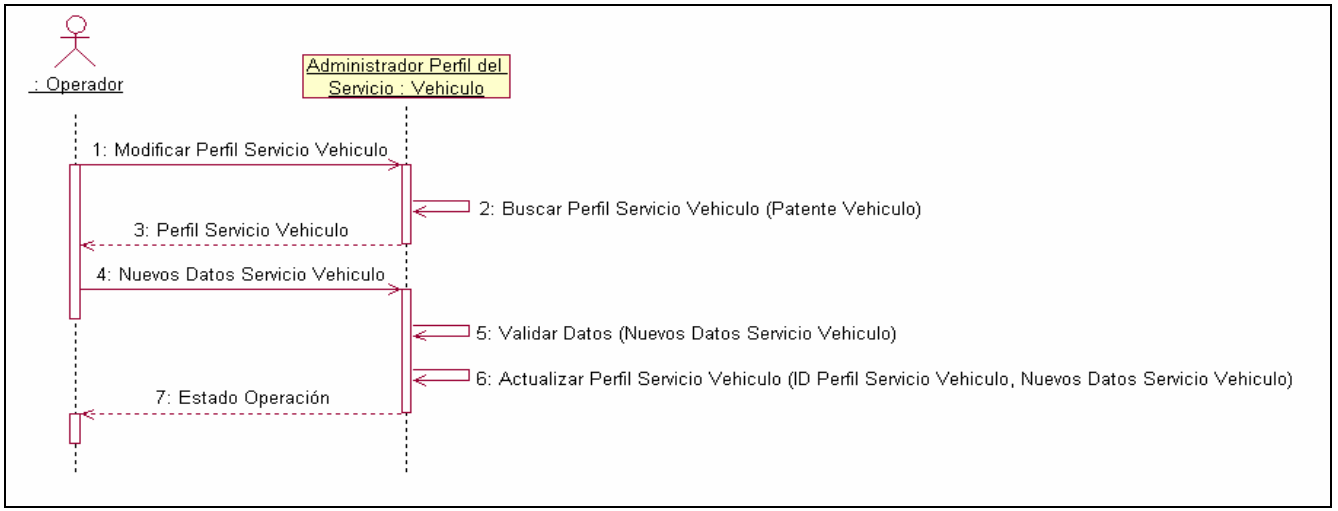

Figura A.12 R.Id 9, operador modifica el perfil de servicio de un vehículo.

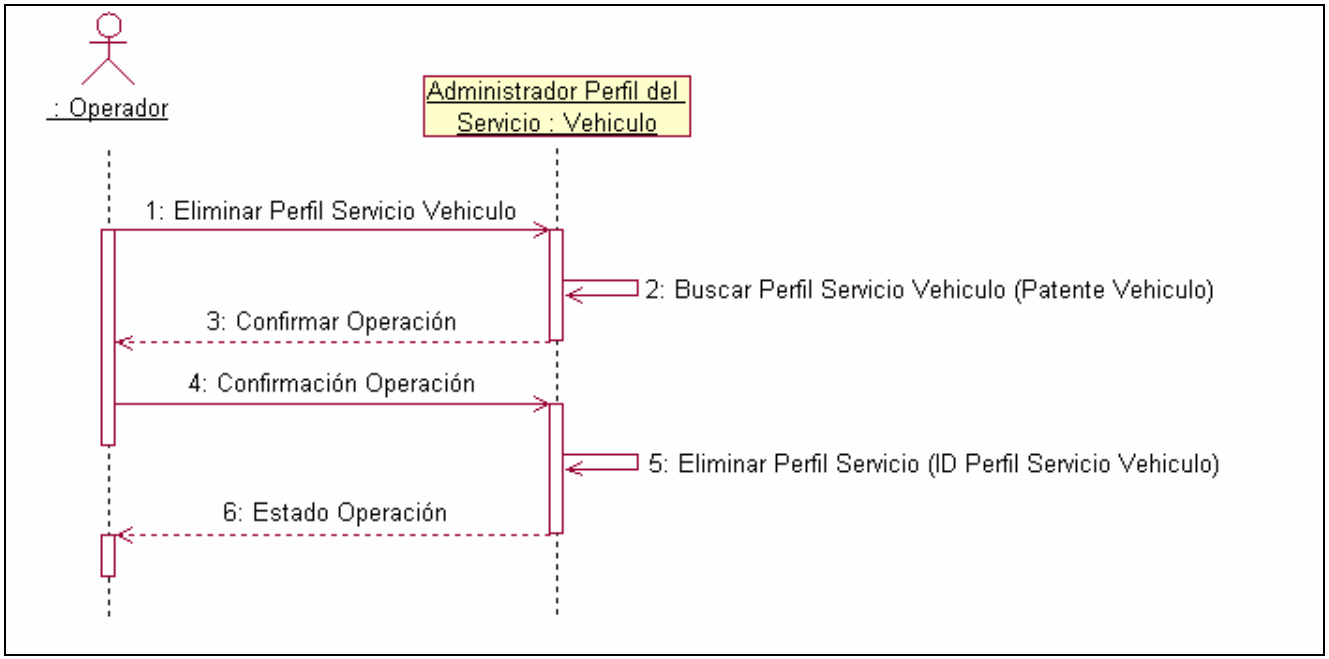

Figura A.13 R.Id 10, operador elimina el perfil de servicio de un vehículo.

Especificación de tareas

Esta fase permite la creación del cuarto y último diagrama dentro del *Modelo de Requerimientos del Sistema* de *PASSI*. En él se representan las distintas tareas que deberán llevar a cabo cada uno de los agentes identificados. Para esto se realiza un diagrama de tareas por agente (diagramas de estado), que permiten ver dos perspectivas en paralelo: las tareas propias del agente y las tareas de otros agentes o actores externos con las cuales estas deben interactuar.

- **T.Sp1 Agente GUI Cliente:** En este diagrama se pueden apreciar todas las tareas que ejecuta el agente *GUI Cliente*. El agente se mantiene a la escucha de una petición y cuando esta se realiza procede a iniciar todas las acciones relativas al proceso de solicitud de viaje, a ejecutar acciones de obtención/validación de datos o a informar/mostrar datos. En la Figura A.14 podemos ver este diagrama.

- **T.Sp2 Agente GUI Vehículo:** En este diagrama se pueden apreciar todas las tareas que ejecuta el agente *GUI Vehiculo*. El agente se mantiene a la escucha de una petición y cuando esta se realiza procede a ejecutar acciones de obtención/validación de datos o a informar/mostrar datos. En la Figura A.15 podemos ver este diagrama.

- **T.Sp3 Agente Cliente:** En este diagrama se pueden apreciar todas las tareas que ejecuta el agente *Cliente*. El agente se mantiene a la escucha de una petición y cuando esta se realiza procede a realizar acciones relativas a la obtención de información sobre el servicio actual, al proceso de pago del viaje, al recibimiento o envío de datos, al procesamiento de eventos, al ingreso del viaje o al proceso de llamado a propuesta. En la Figura 8.16 podemos ver este diagrama.

- **T.Sp4 Agente Vehículo:** En este diagrama se pueden apreciar todas las tareas que ejecuta el agente *Vehículo*. El agente se mantiene a la escucha de una petición y cuando esta se realiza procede a realizar acciones relativas a la obtención de información geográfica y del perfil del servicio del vehículo, al mantenimiento del itinerario, al recibimiento o envío de datos, al procesamiento de eventos y al proceso de llamado a propuesta .En la Figura A.17 podemos ver este diagrama.

- **T.Sp5 Agente Map:** En este diagrama se pueden apreciar todas las tareas que ejecuta el agente *Map*. El agente se mantiene a la escucha de una petición y cuando esta se realiza procede a realizar acciones relativas a la obtención de información geográfica (puntos de detención), al recibimiento de datos, al procesamiento de eventos, a la creación o actualización de la matriz de distancias y al cálculo de la ruta de viaje más corta. En la Figura A.18 podemos ver este diagrama.

- **T.Sp6 Agente GUI Broker:** En este diagrama se pueden apreciar todas las tareas que ejecuta el agente *Broker*. El agente se mantiene a la escucha de una petición, y cuando esta se realiza procede a realizar acciones relativas a la obtención, elección

y actualización de servicios disponibles. En la Figura A.19 podemos ver este diagrama.

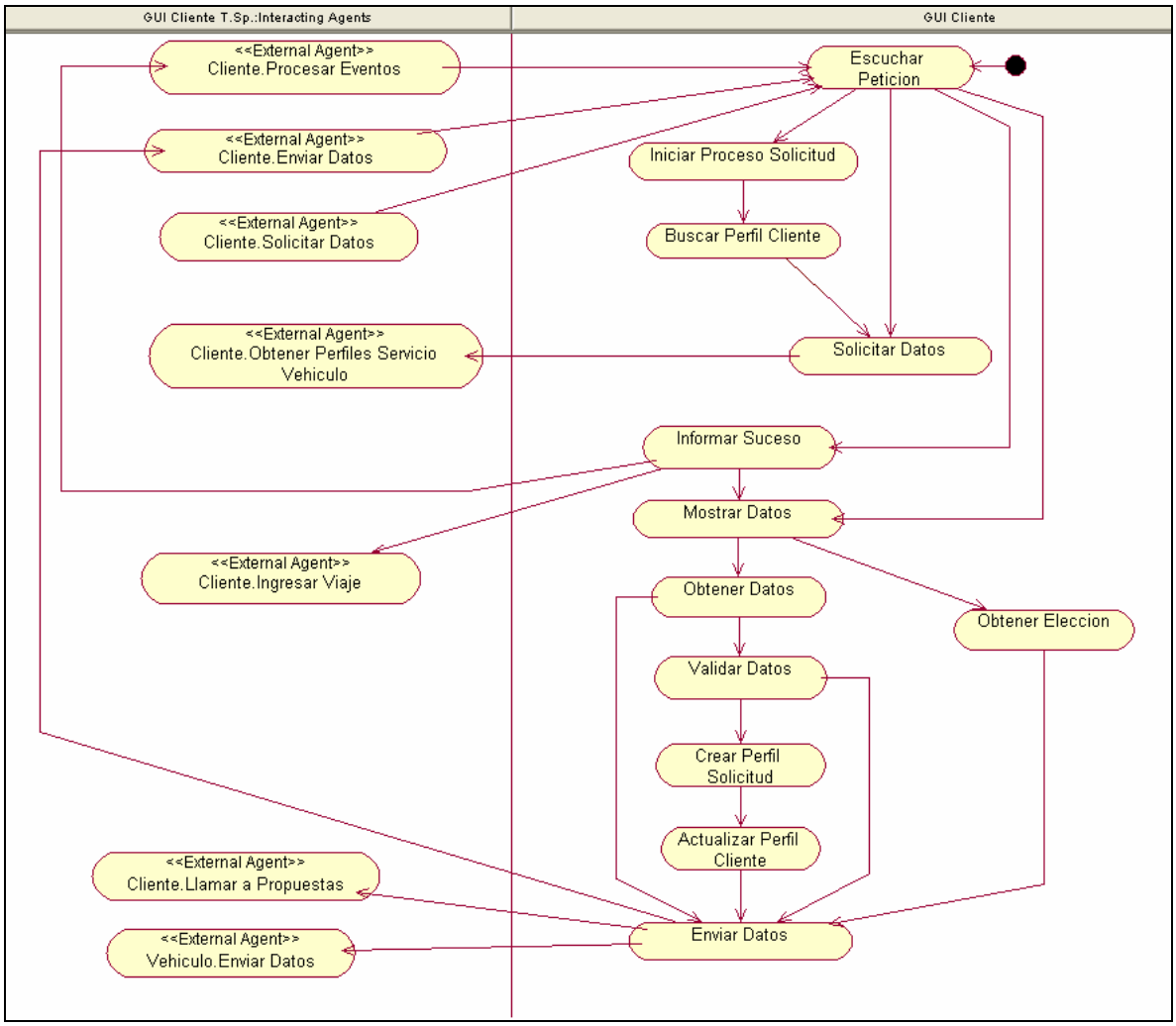

Figura A.14 T.Sp1, agente GUI Cliente.

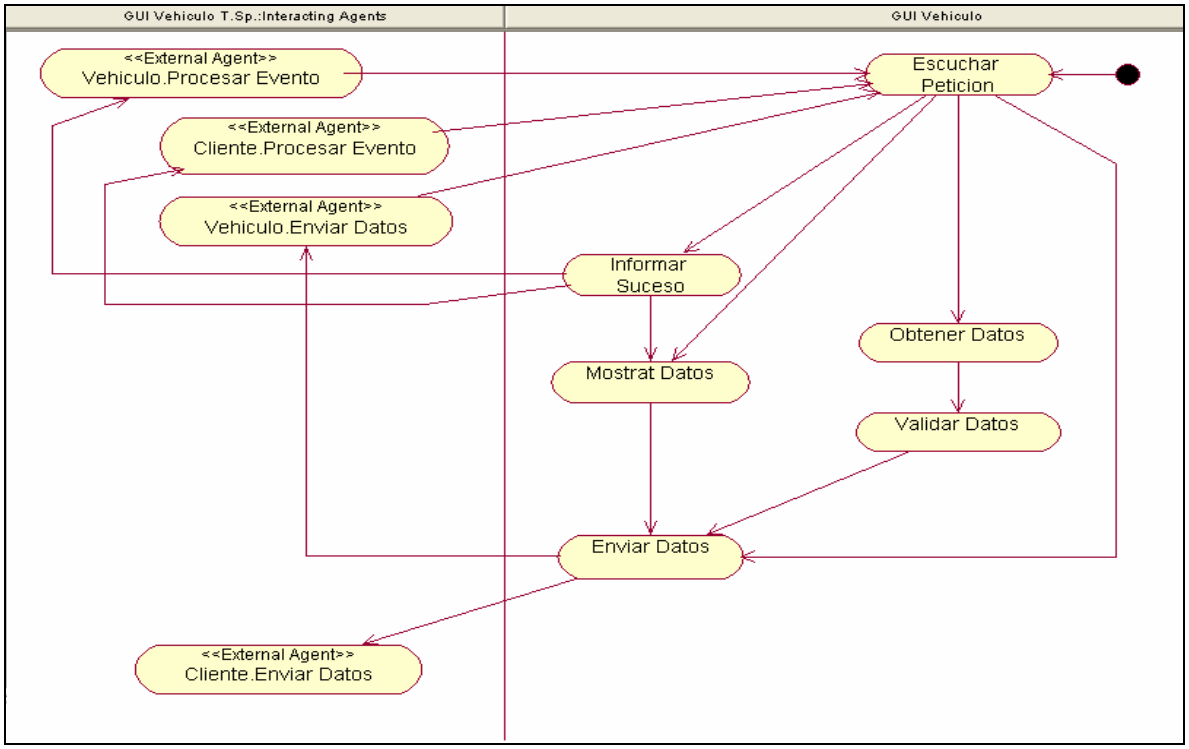

Figura A.15 T.Sp2, agente GUI Vehículo.

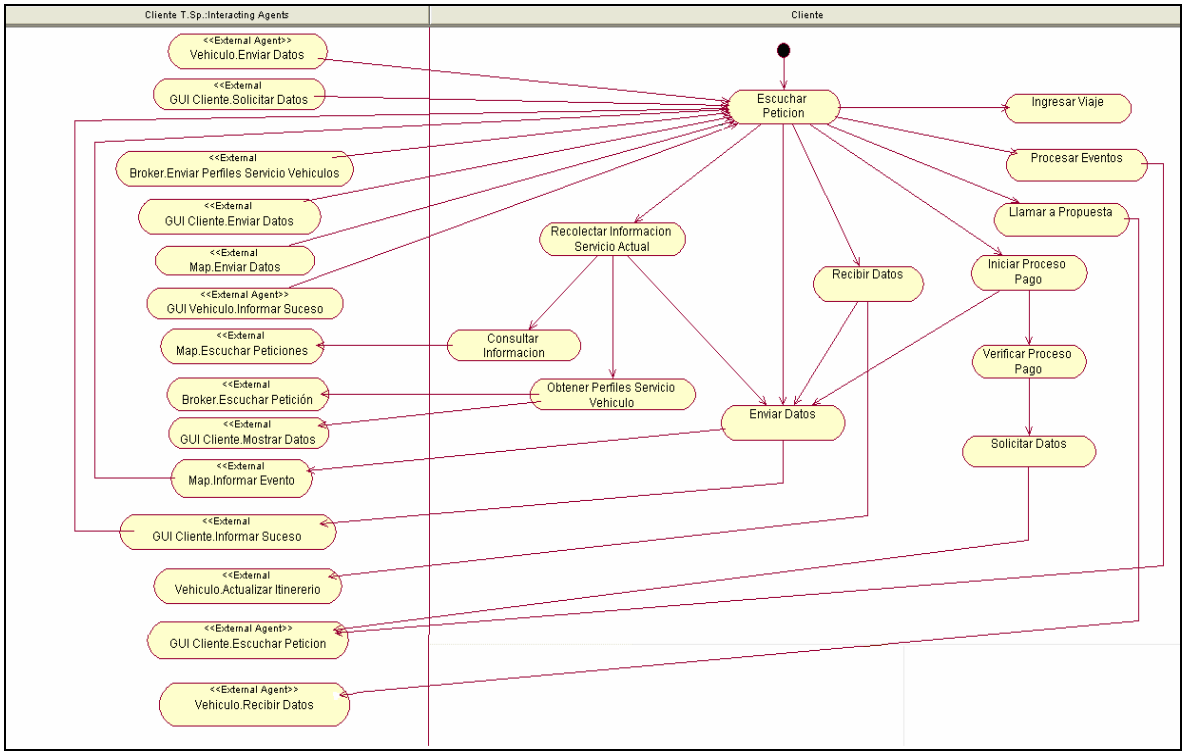

Figura A.16 T.Sp3, agente Cliente.

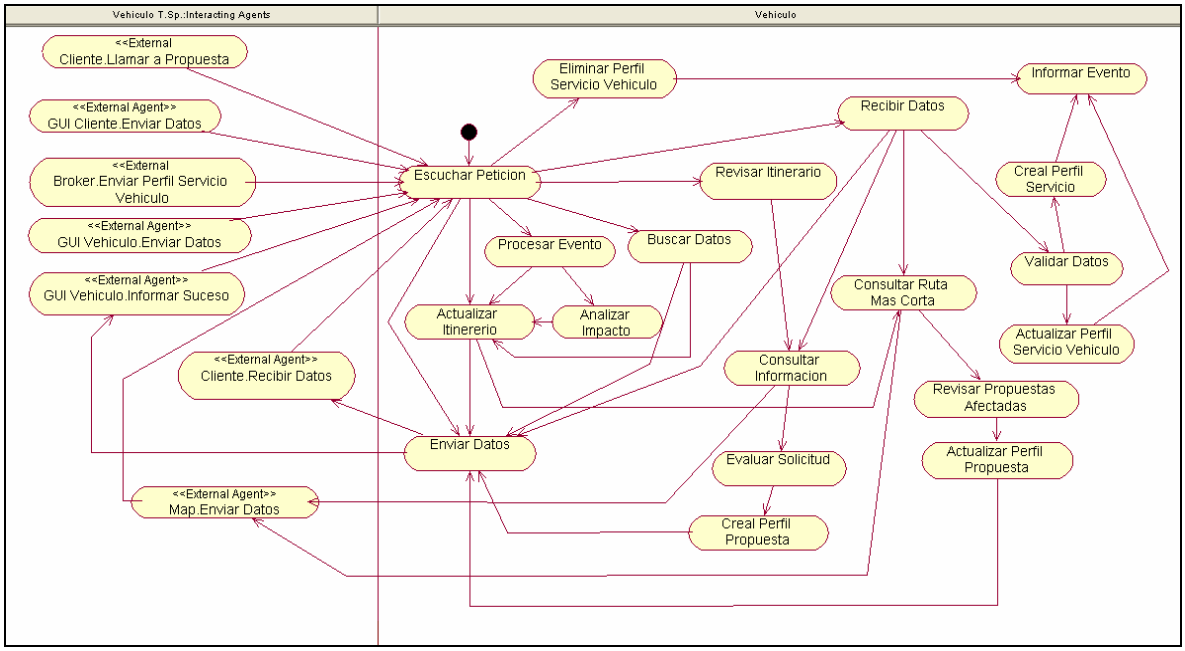

Figura A.17 T.Sp4, agente Vehículo.

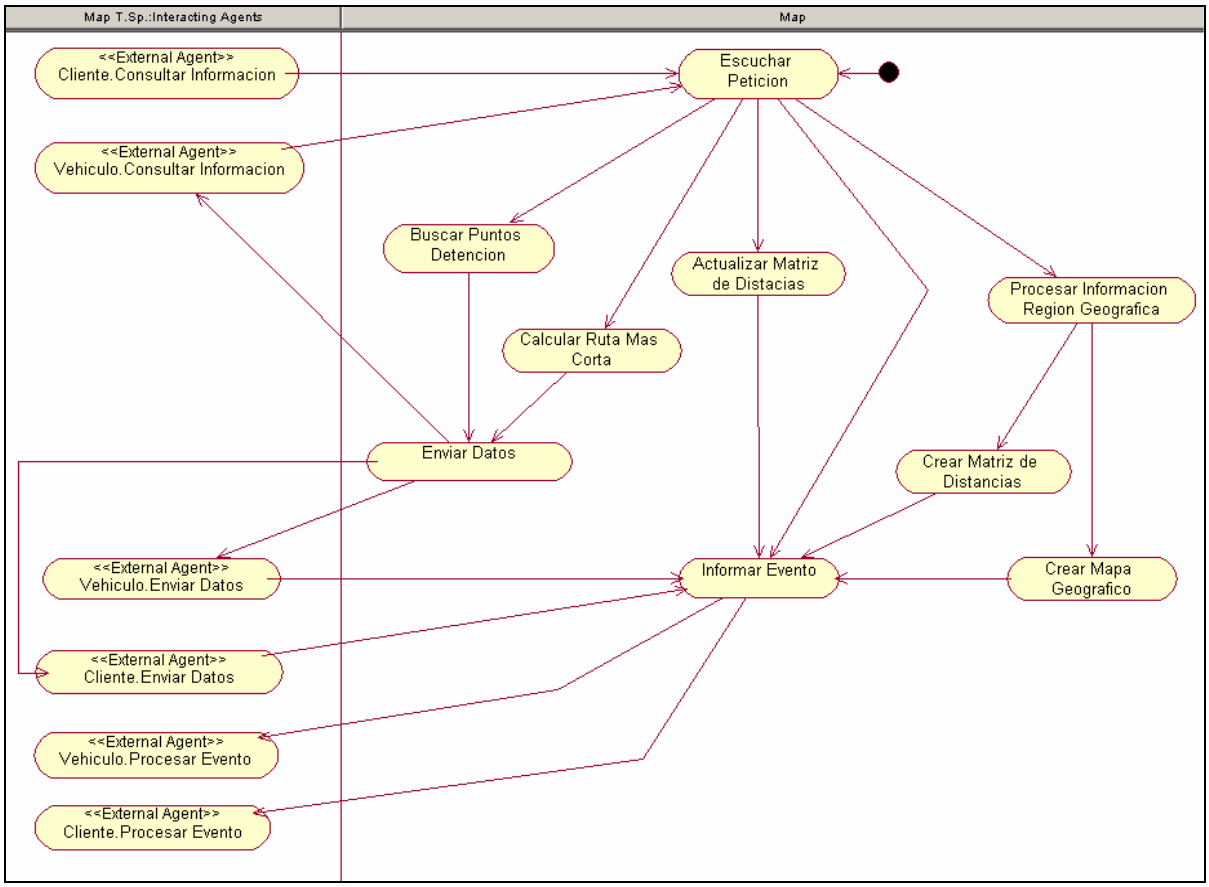

Figura A.18 T.Sp5, agente Map.

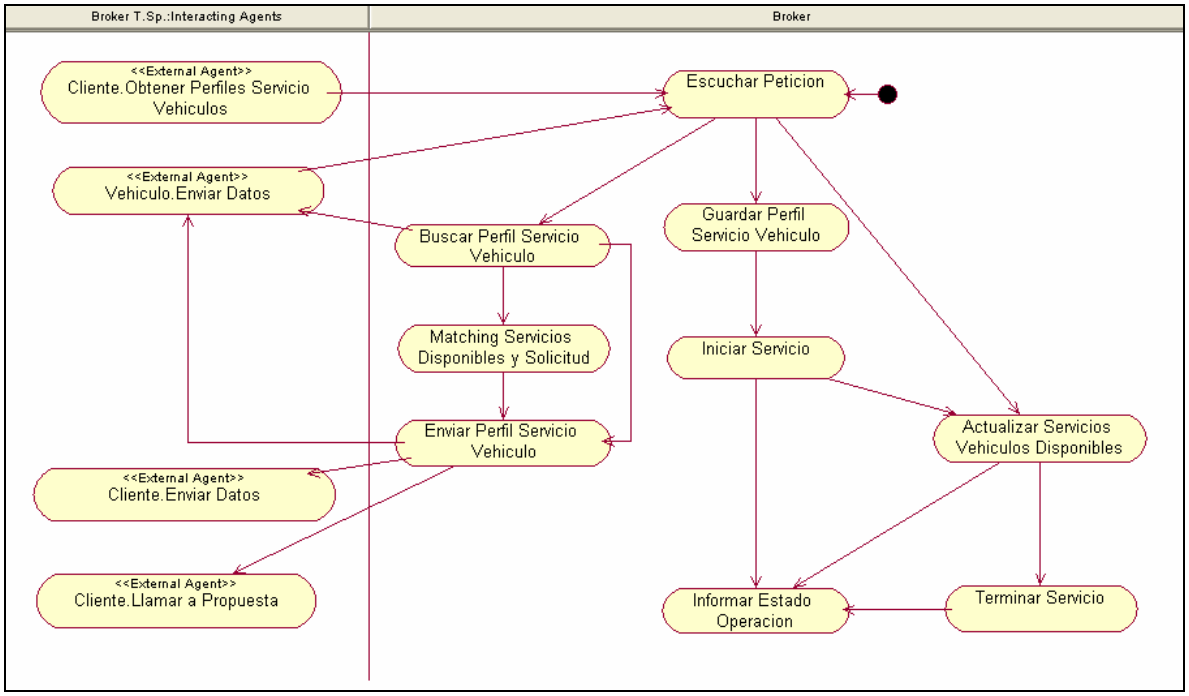

Figura A.19 T.Sp6, Agente GUI Broker.

A.1.2 Modelo de sociedad de agente

Descripción de la ontología del dominio

Esta fase intenta describir la sociedad de agentes desde el punto de vista del conocimiento que deben poseer los agentes para poder entenderse entre sí. En el *Diagrama de Descripción de la Ontología del Dominio (D.O.D.)* la ontología es descrita (usando un diagrama de clases) en términos de conceptos, predicados y acciones. Por ejemplo, en la Figura A.20a podemos ver el concepto *Ruta*, el cual representa la ruta que el vehículo recorrerá para llevar al cliente desde el lugar de recogida hasta el lugar de destino. La *Ruta* tiene asociadas las siguientes acciones: *Crear Ruta* (permite a un agente determinar la ruta de viaje), *Solicitar Ruta* (permite a un agente solicitar una ruta de viaje) y *Actualizar Ruta* (permite modificar la ruta original de viaje en caso de que ocurra un *Evento de Tráfico*). Por otra parte, una *Ruta* está compuesta por uno o más *Nodos* (puntos cartesianos previamente establecidos en el mapa geográfico), los cuales pueden o no ser paradas. Esta propiedad es manejada como predicado; ya que puede tomar los valores verdadero o falso.

Las Figuras A.20a y A.20b muestran el *Diagrama de Descripción de la Ontología del Dominio* (*D.O.D.*) del sistema *DRT*.

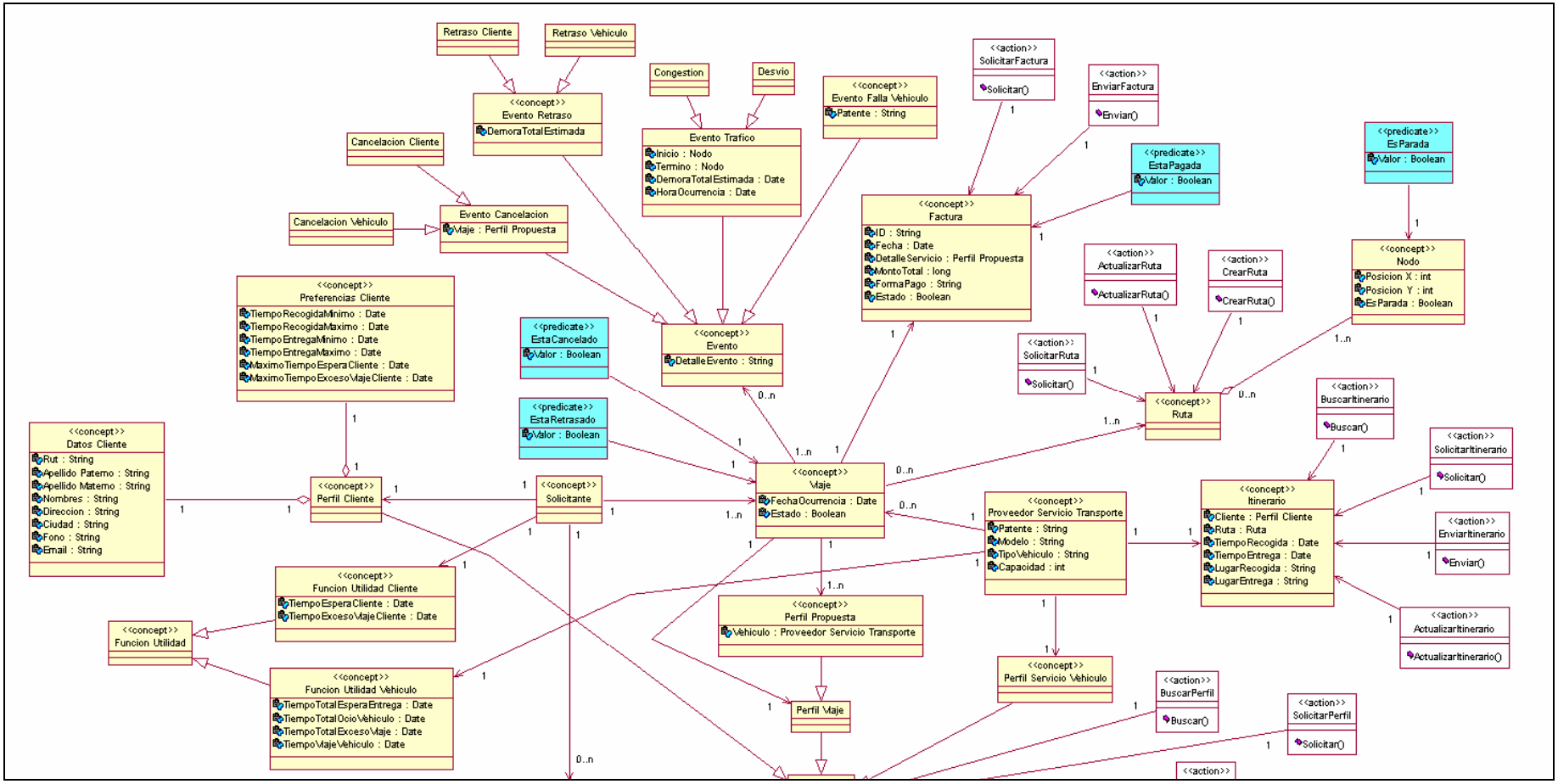

Figura A.20a D.O.D., descripción de la ontología de dominio (parte1).

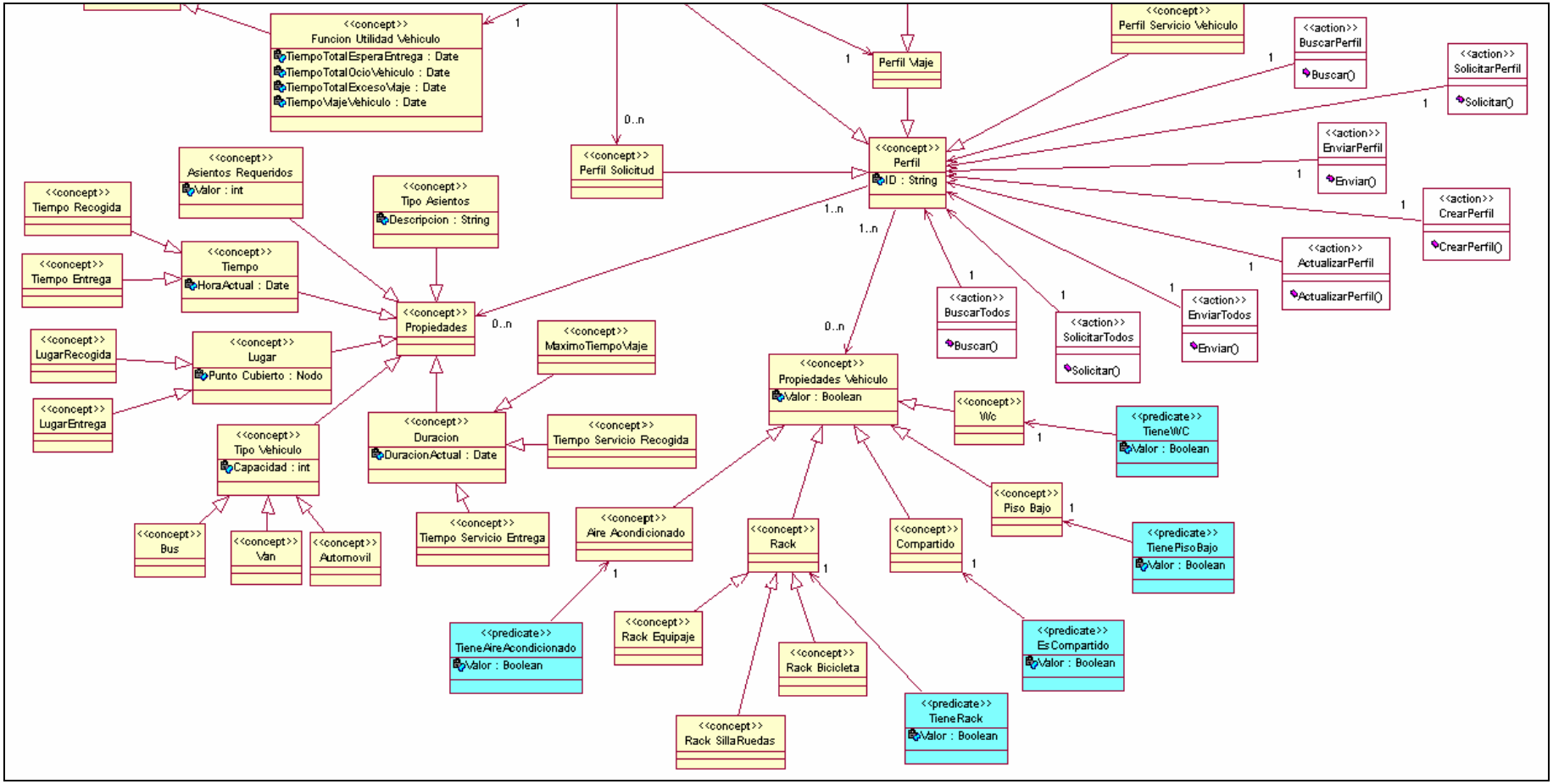

Figura A.20b D.O.D., descripción de la ontología de dominio (parte2).

Descripción de la ontología de comunicación

Esta fase intenta describir la sociedad de agentes desde el punto de vista de la comunicación (interacción) entre los diversos agentes que componen el sistema. Cada agente se describe en términos de su conocimiento (partes de la ontología descrita en el diagrama anterior). Hay una relación entre dos agentes para cada comunicación en la que están implicados. En cada relación los roles desempeñados por los agentes durante la comunicación también son reportados. Cada comunicación es representada por la relación entre dos agentes y se detalla dentro de los atributos de la clase de la relación. La clase es identificada por un nombre único (también reportado en la relación entre dos agentes) y es descrita por los campos de ontología, lenguaje y de protocolo. El campo de ontología refiere a un elemento del *D.O.D.* (descripción de la ontología de dominio); el lenguaje apunta a los contenidos del lenguaje de comunicación, mientras que el protocolo precisa el protocolo de interacción adoptado por *FIPA*.

Por ejemplo, en la Figura A.21a podemos ver al agente *GUI Cliente*, el cual entiende y maneja los siguientes conceptos de la ontología de dominio del *DRT*: *Perfil Solicitud*, *Factura*, *Perfil Propuesta*, *Evento y Perfil Servicio Vehículo*. Este agente establece comunicación con el agente *Cliente* (comunicación *GUI Cliente – Cliente3*) para enviarle los datos de la factura del viaje, la cual es posible porque ambos agentes entienden el concepto *Factura* (ontología). Además de establecer la ontología de dominio, se establece también el lenguaje a utilizar (en este caso *FIPA ACL*) y el protocolo de comunicación (*FIPA Request*). Por otra parte, para realizar el rol de *Generador de Solicitud de Viaje*, el agente *GUI Cliente* realiza la tarea de *Enviar Datos*, mientras que el agente *Cliente* realiza la tarea de *Escuchar Petición* para desempeñar el rol de *Administrador Pago Viaje*.

Las Figuras A.21a y A.21b muestran el *Diagrama de Descripción de la Ontología de Comunicación* (*C.O.D.*) del sistema *DRT*.

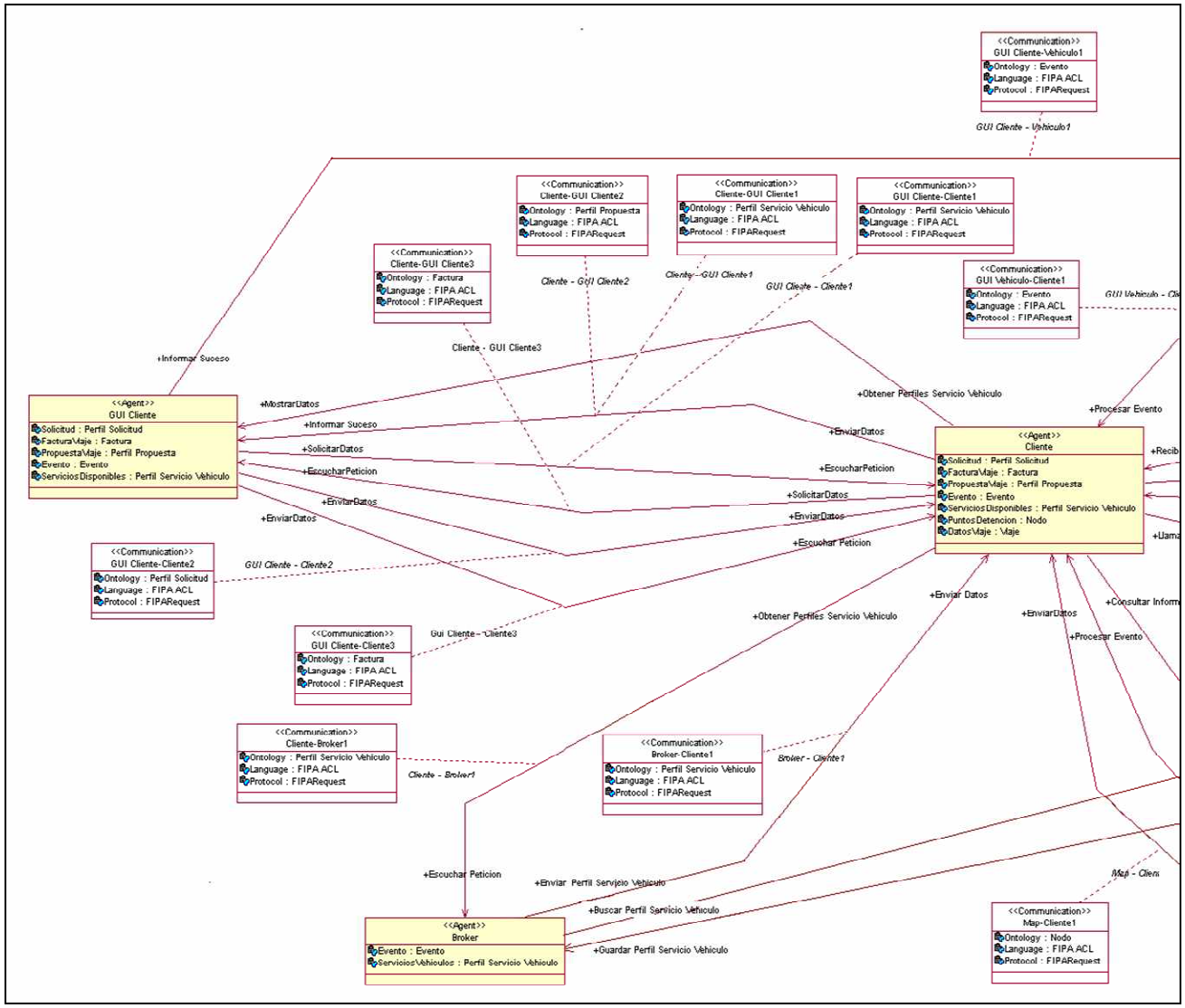

Figura A.21a C.O.D., descripción de la ontología de comunicación (parte1).

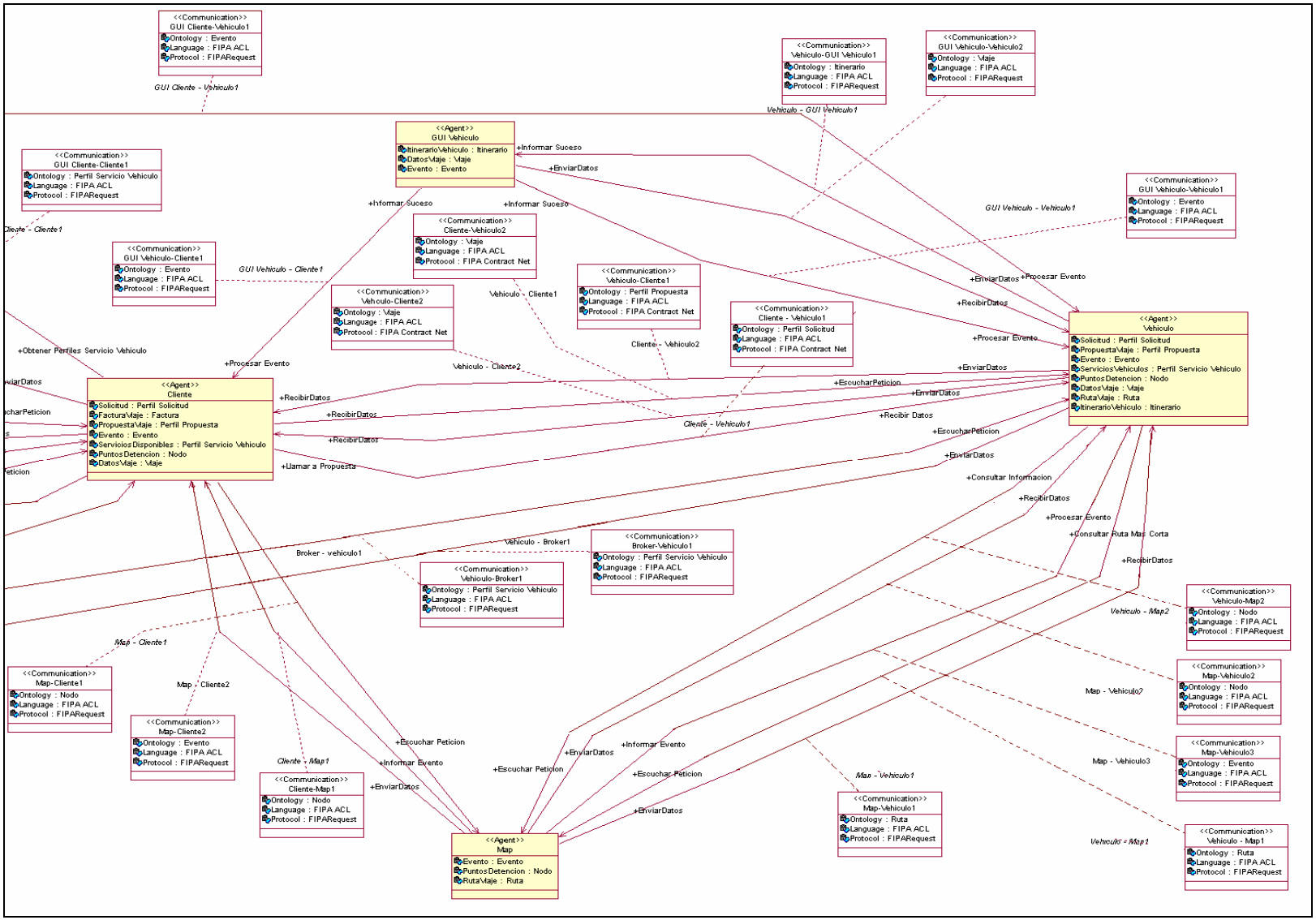

Figura A.21b C.O.D., descripción de la ontología de comunicación (parte2).

Descripción de roles

Representamos el diagrama de descripción de roles como un diagrama de clases, donde los roles son clases agrupadas en paquetes que representan a los agentes. Los roles se pueden conectar por relaciones que representan cambios de roles (*ROL CHANGE*), por dependencias de un servicio o disponibilidad de un recurso y por comunicaciones. Cada rol obtenido está compuesto por varias tareas, las cuales se especifican en cada clase. Por ejemplo, el agente *Cliente* como *Administrador De solicitudes* necesita consultar al agente *Broker* sobre los servicios que se encuentran disponibles y que encajan con el perfil de la solicitud, a través de la tarea *Recolectar Información del Servicio*. Por su parte, el agente *Broker* como *Administrador de la Información del Servicio de Vehículos* debe realizar la tarea *Matching Servicios Disponibles* para enviar el *Perfil del Servicio Disponible* en caso de que el matching sea favorable.

La Figura A.22 muestra el *Diagrama de Descripción de Roles* (*R.D.*) del sistema *DRT*.

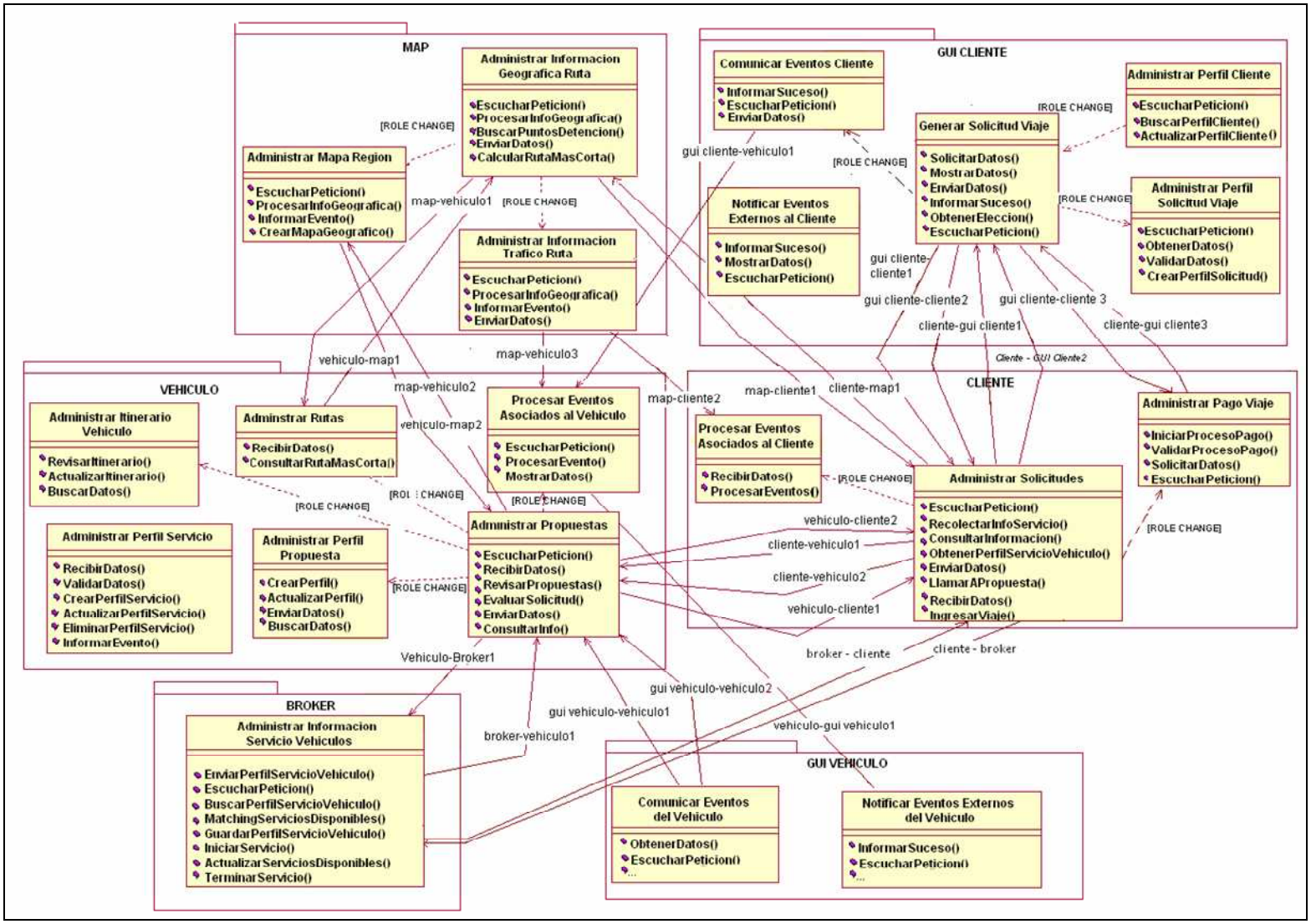

Figura A.22 R.D., descripción de roles.

Descripción de protocolos

En esta fase se utilizan diagramas de secuencia para especificar los protocolos de interacción. Sin embargo, como los protocolos utilizados son protocolos de interacción *FIPA* no se necesita su especificación.

A.1.3 Modelo de implementación de agentes

Definición de la estructura del sistema multiagente

La definición de la estructura del sistema multiagente es representada a través de un diagrama de clases, el cual contiene clases y actores. Cada clase simboliza un agente del sistema. Los atributos se pueden utilizar para representar el conocimiento del agente (que refiere a las entidades definidas en el *D.O.D.*), mientras que las operaciones se utilizan para representar las tareas del agente. Las relaciones indican el flujo de la información intercambiada (las comunicaciones).

La Figura A.23 muestra el *Diagrama de definición de la estructura del sistema multiagente (M.A.S.D.)* del sistema *DRT*.

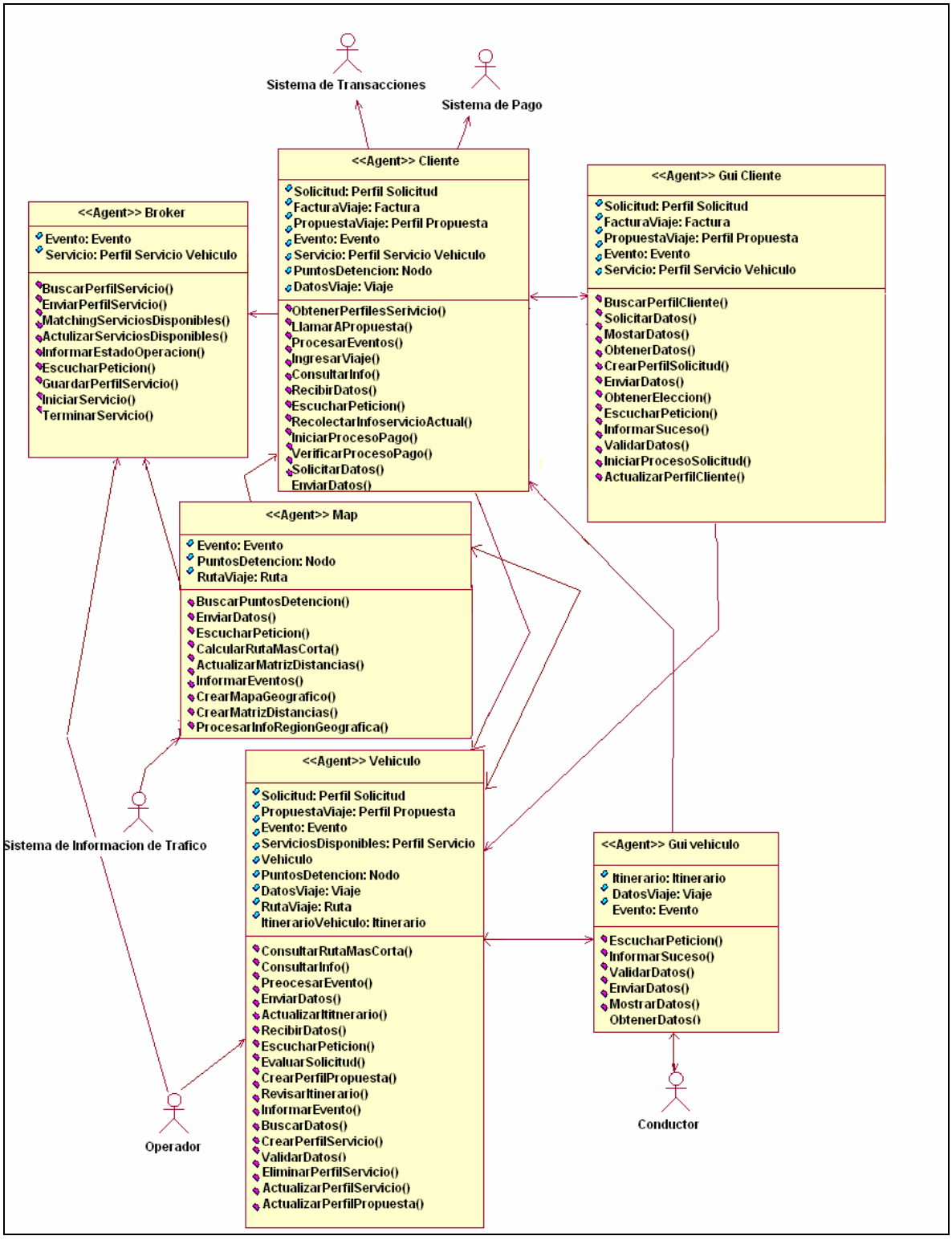

Figura A.23 M.A.S.D., definición de la estructura del sistema multiagente.

Definición de la estructura del agente

El *Diagrama de Definición de la estructura de un Agente* (*S.A.S.D.*) se representa a través de un diagrama de clases. Se debe crear un diagrama de clases para cada agente. Este diagrama describe la estructura del agente y todas sus tareas. Cada clase representa el agente o una de sus tareas. Los atributos y métodos son los elementos que constituirán la puesta en marcha verdadera del sistema (código). Es posible producir automáticamente código a partir de este diagrama con muchas herramientas comerciales.

Las Figuras A.24, A.25, A.26, A.27a, A.27b, A.28a, A.28b y A.29 muestran los diagramas correspondientes a cada agente.

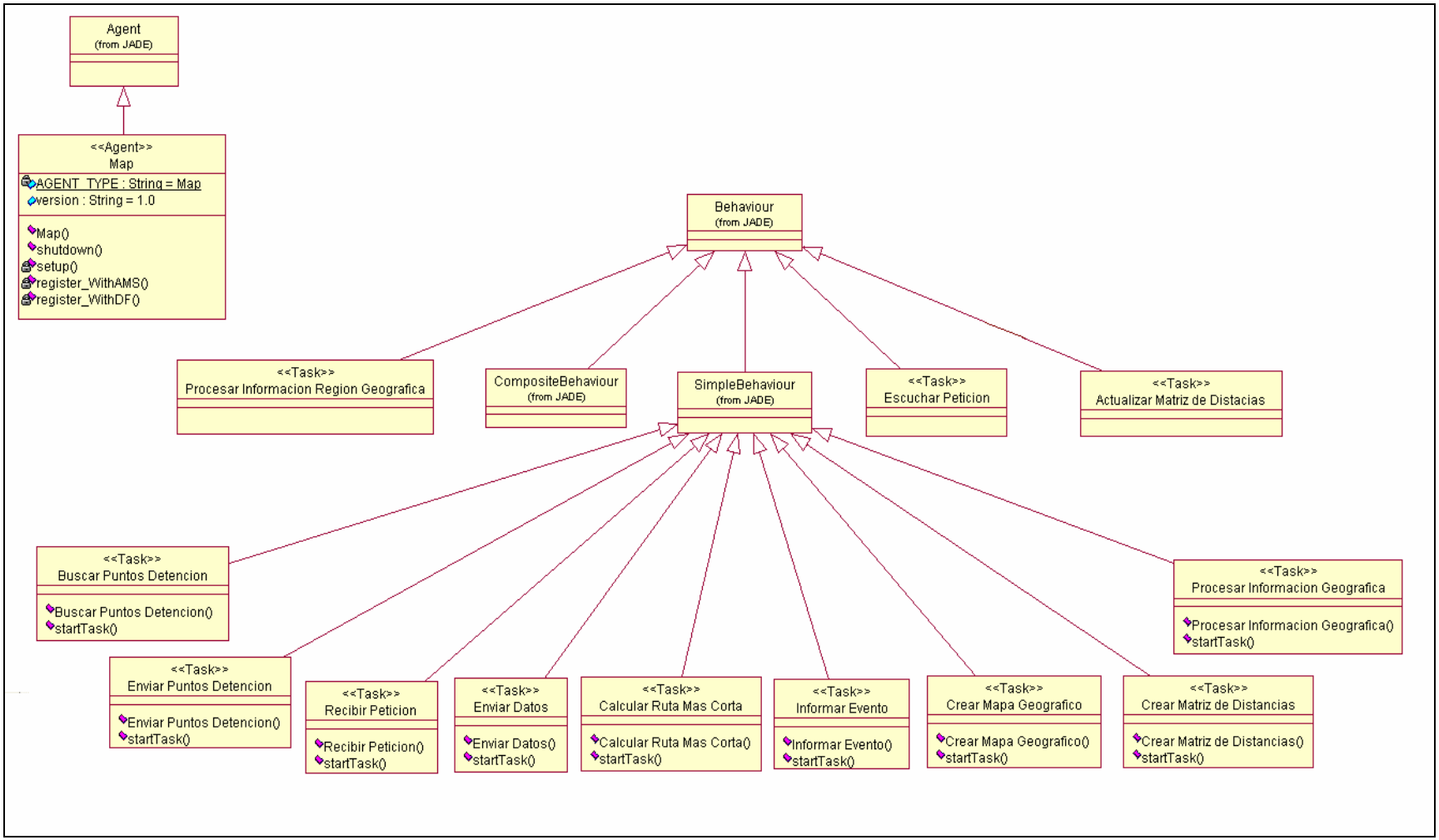

Figura A.24 S.A.S.D., definición de la estructura del agente Map.

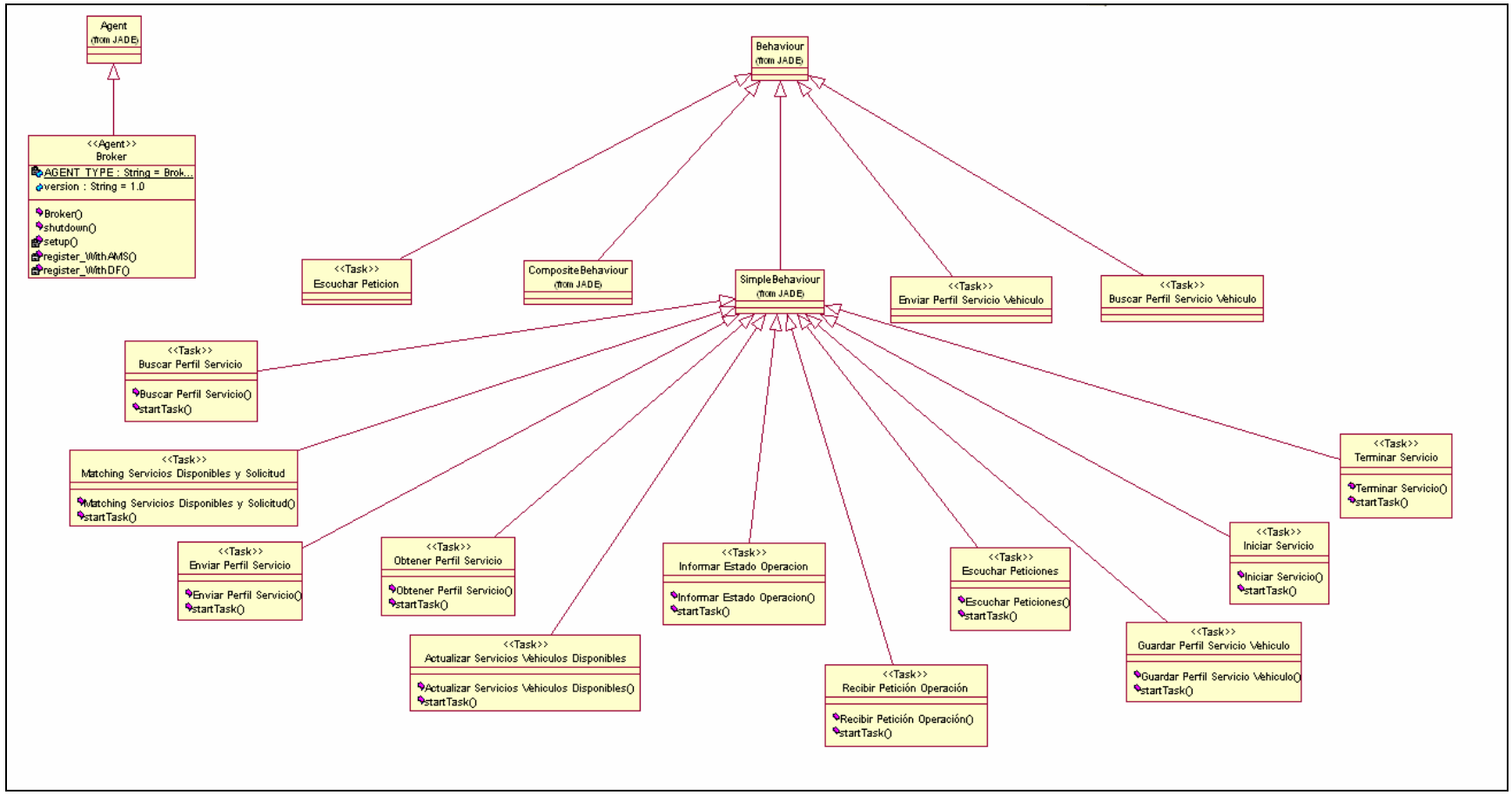

Figura A.25 S.A.S.D., definición de la estructura del agente Broker.

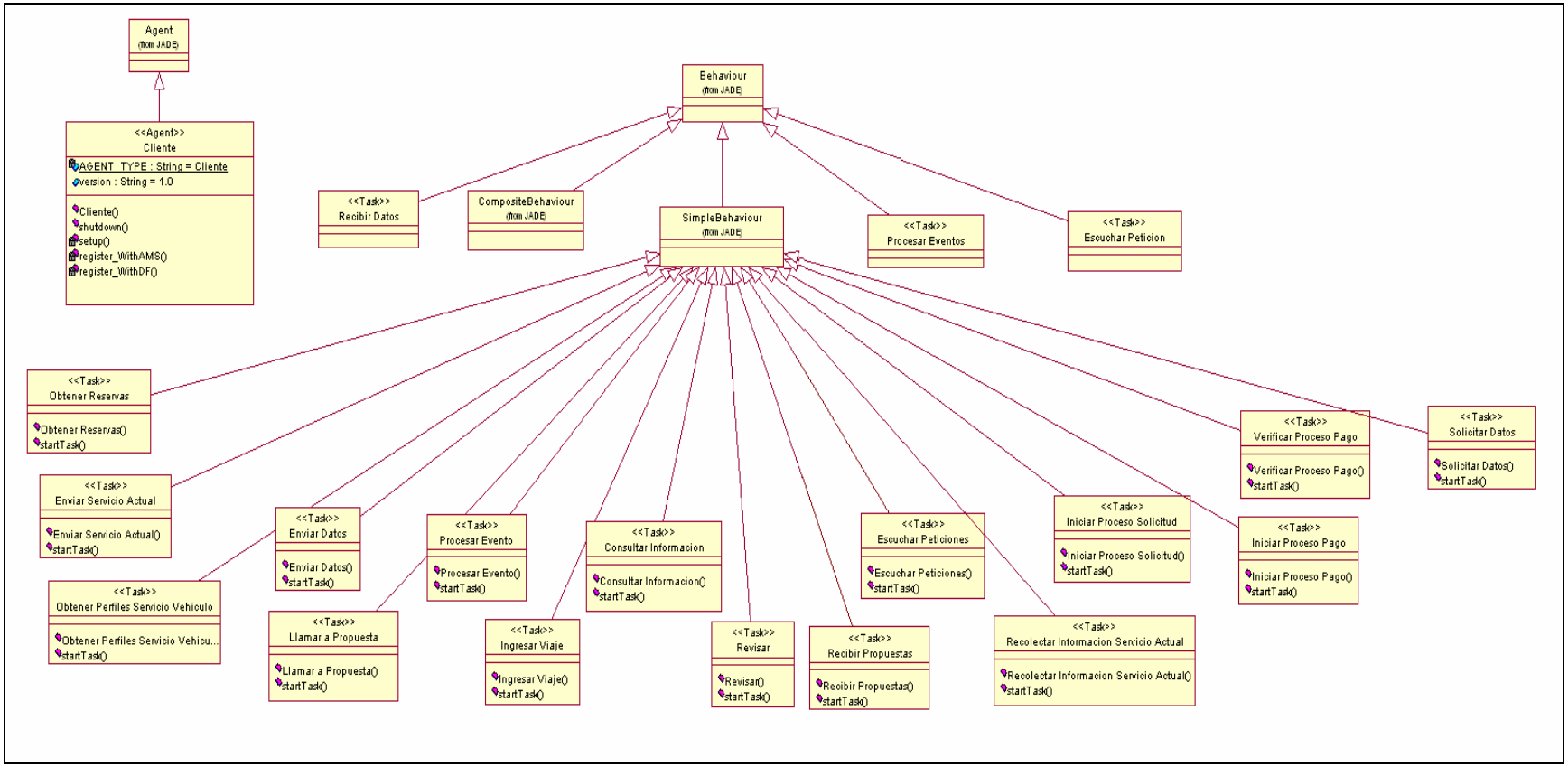

Figura A.26 S.A.S.D., definición de la estructura del agente Cliente.

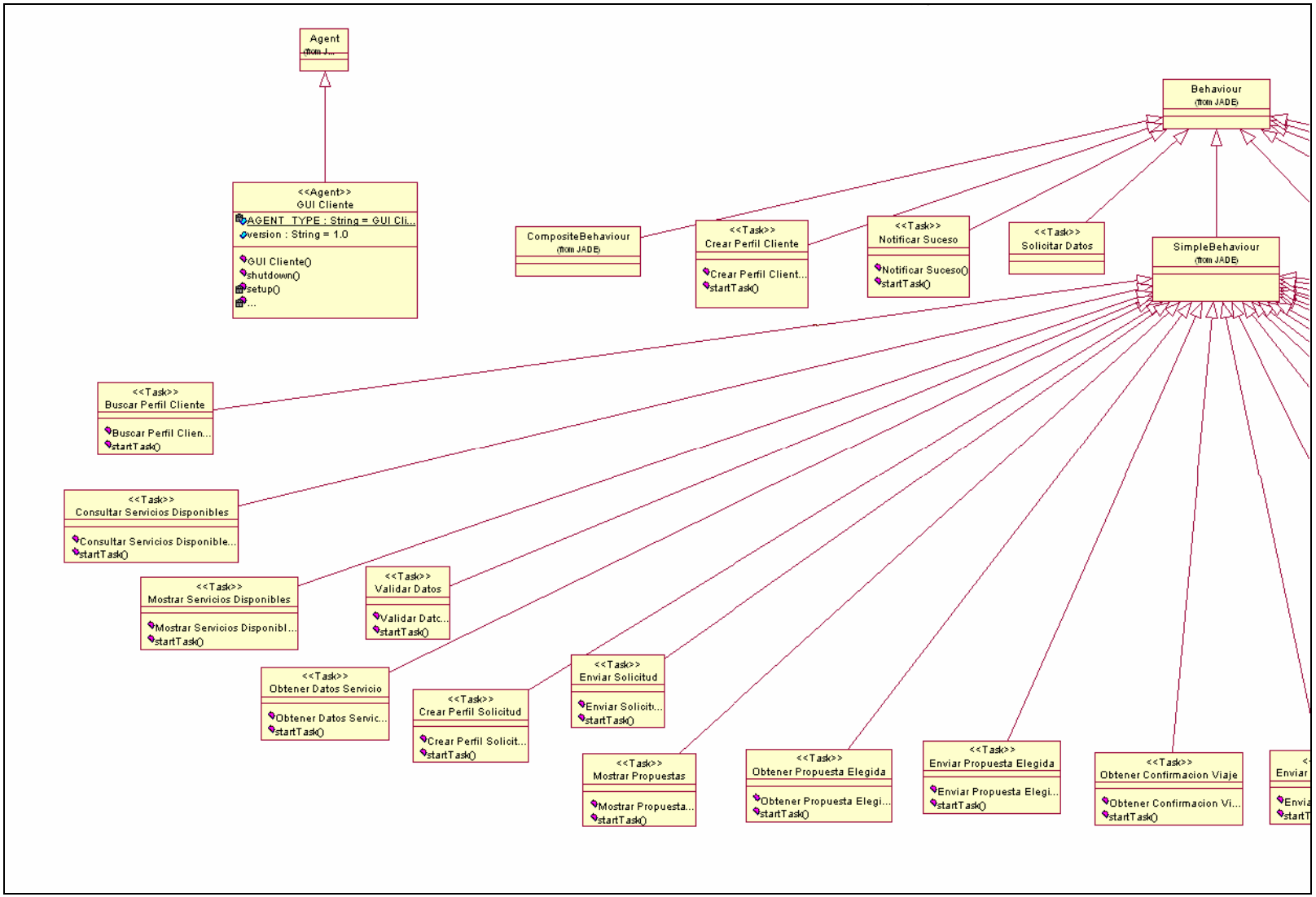

Figura A.27a S.A.S.D., definición de la estructura del agente GUI Cliente.

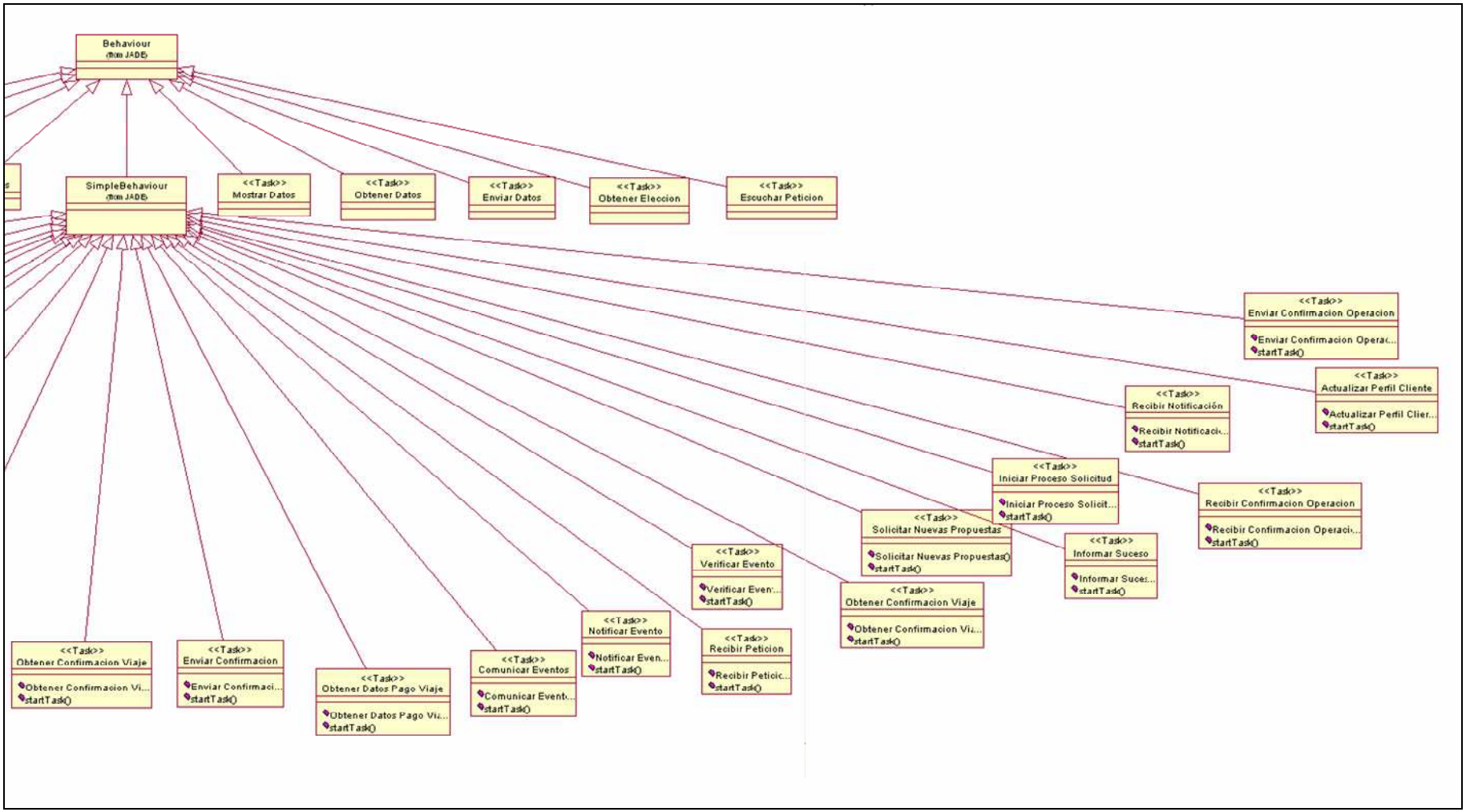

Figura A.27b S.A.S.D., definición de la estructura del agente GUI Cliente.

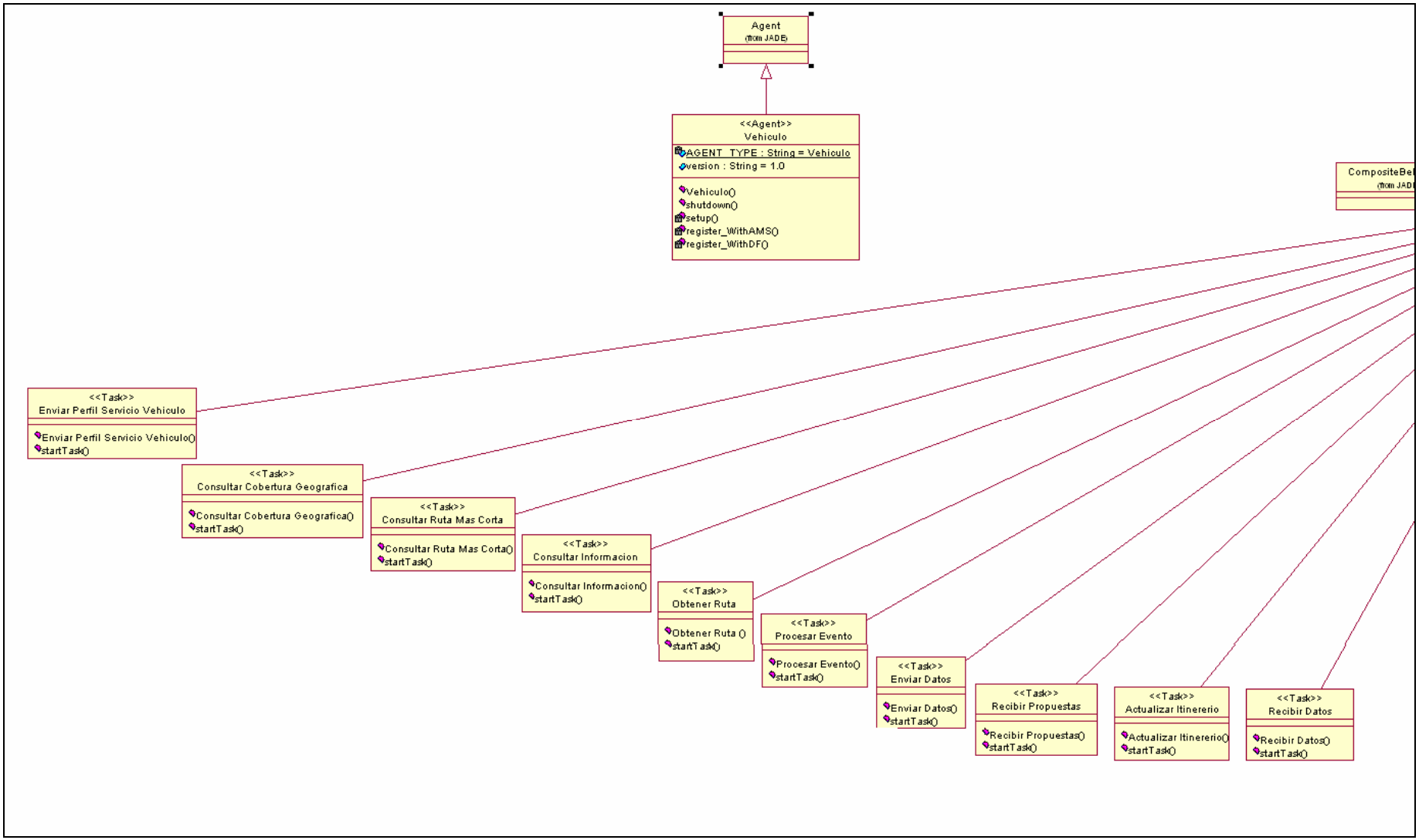

Figura A.28a S.A.S.D., definición de la estructura del agente Vehículo.

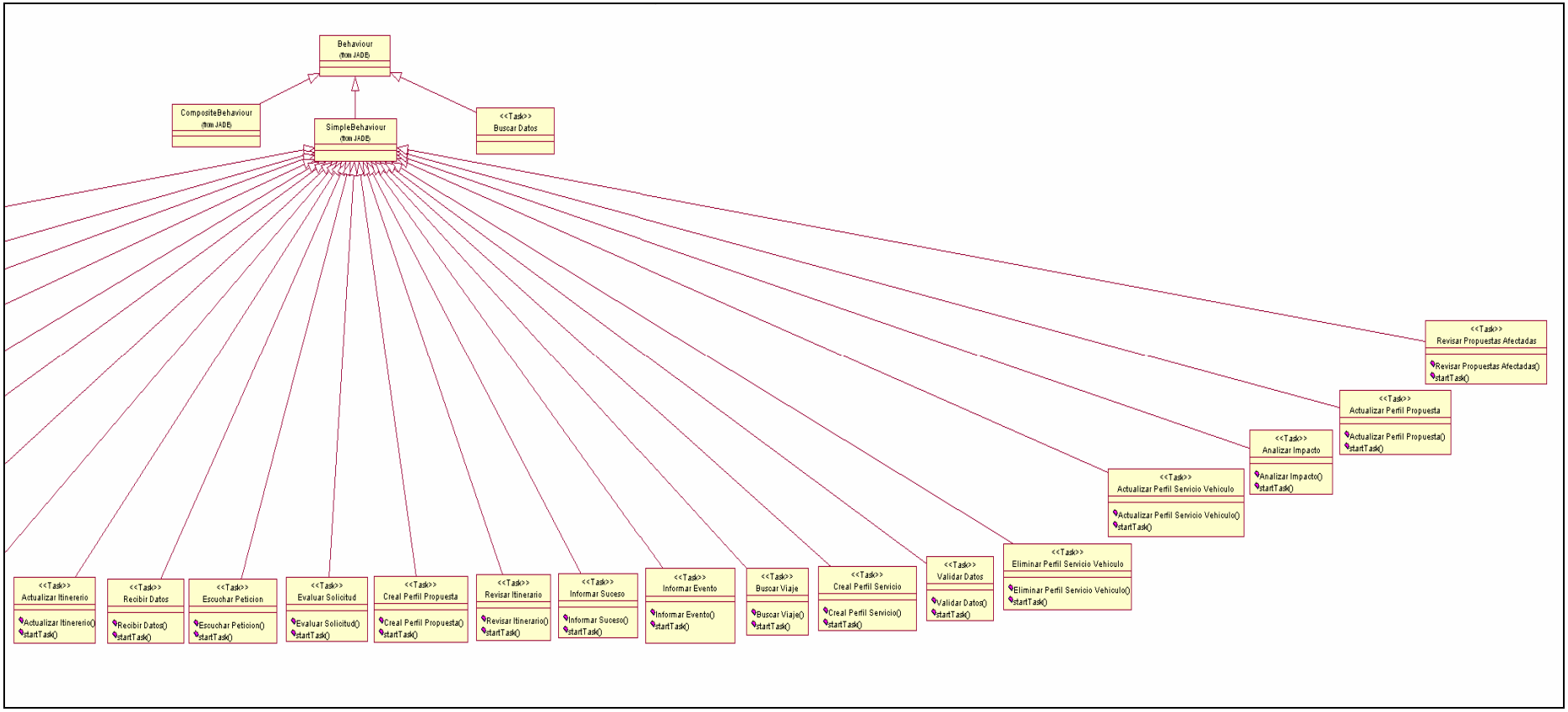

Figura A.28b S.A.S.D., definición de la estructura del agente Vehículo.

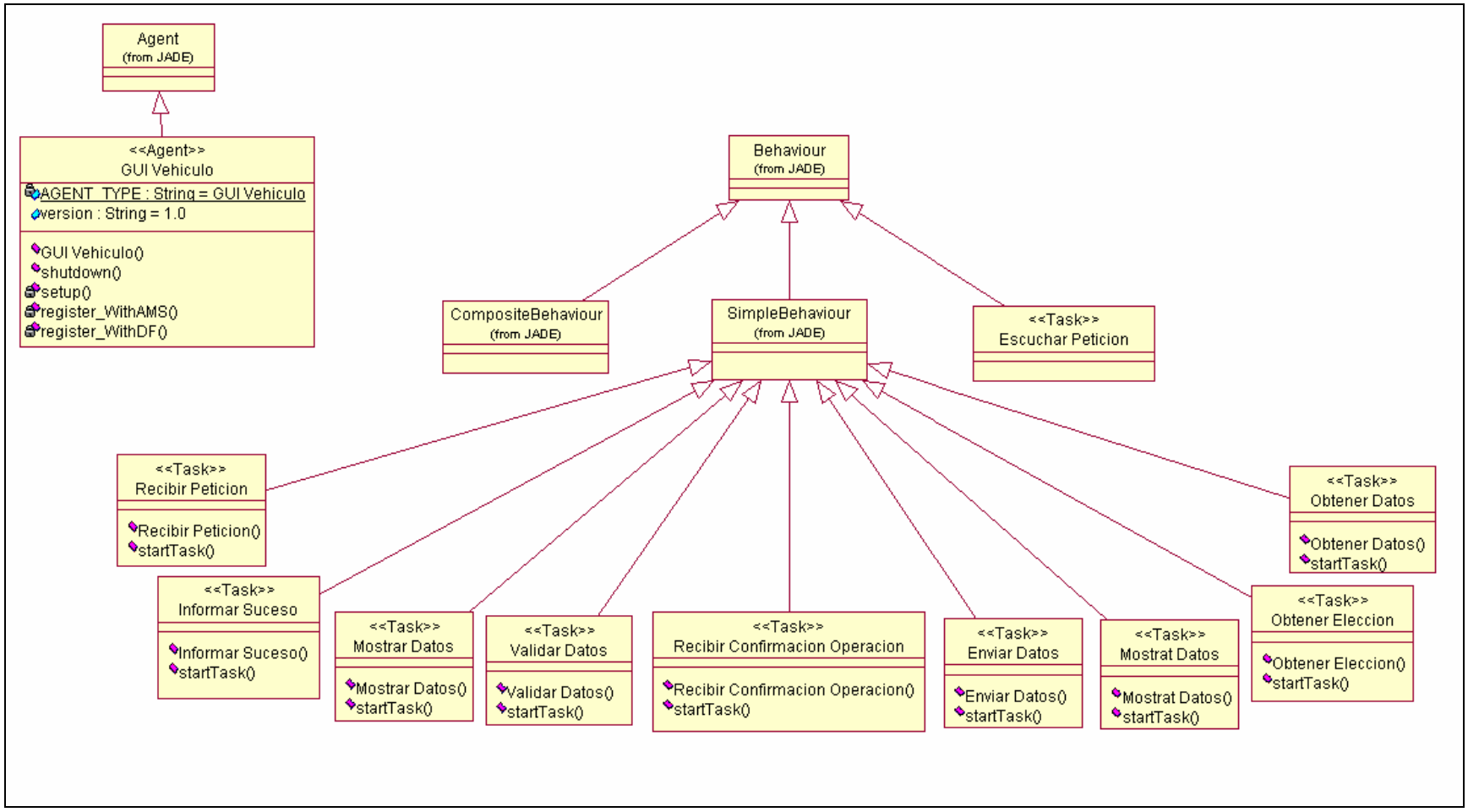

Figura A.29 S.A.S.D., definición de la estructura del agente GUI Vehículo.

A.1.4 Modelo de código

En esta etapa de la metodología el *PTK* genera automáticamente las estructuras (carcasas) de cada agente. Por motivos de alcance del proyecto, esta etapa no se especificará. Sin embargo será contemplada dentro de la evaluación.

A.1.5 Modelo de despliegue

El diagrama de despliegue se utiliza para describir la diseminación de los agentes a través de las plataformas disponibles y sus movimientos (agentes móviles). Las plataformas se describen como procesos de nodos, los agentes como componentes y sus comunicaciones conectan el iniciador con el símbolo de interfaz del participante. Los movimientos del agente se representan con una relación (etiqueta *move_to*) del agente en la anterior posición (nodo que procesa) al agente en la posición de destino (después de mover).

En la Figura A.30 se puede observar el *Diagrama de Despliegue* (*D.D.*) para el sistema *DRTS*.

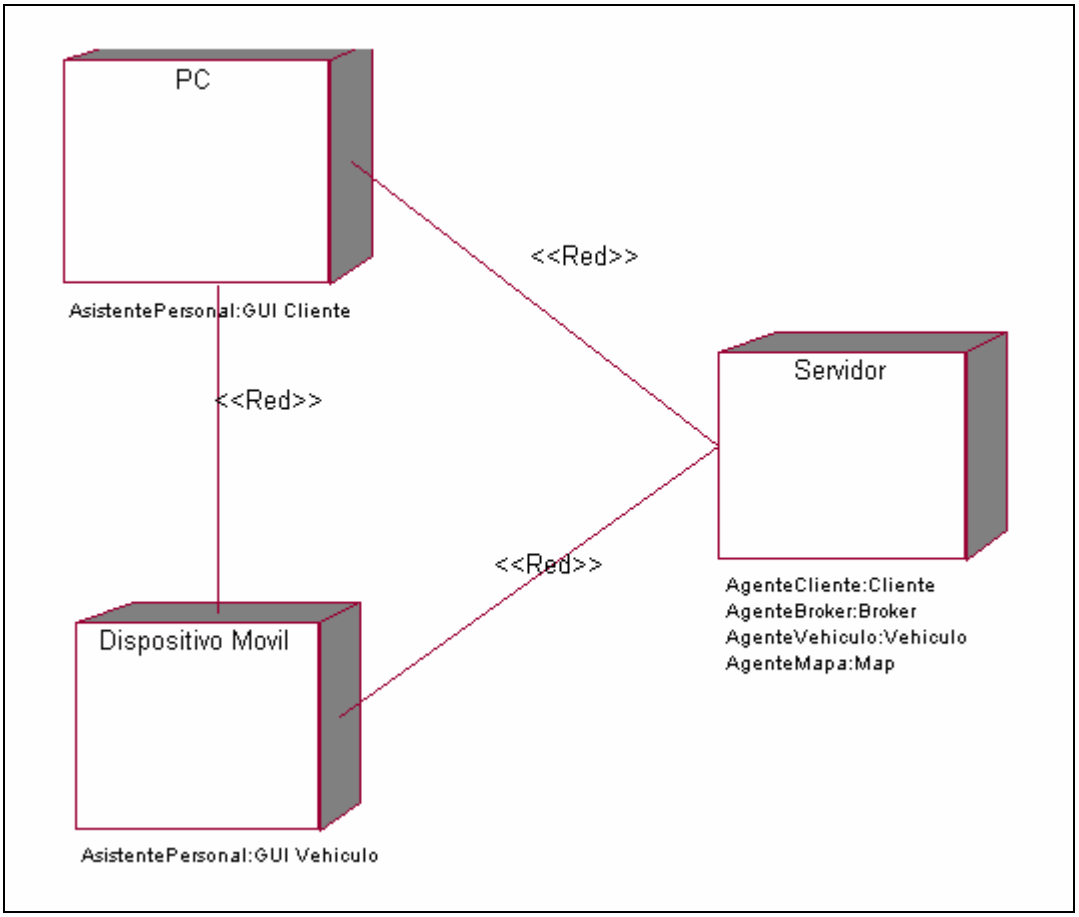

Figura A.30 D.D., diagrama de despliegue DRTS.

A.2 Modelado del sistema con MASE

A.2.1 Capturar objetivos

Esta fase toma lo más importante de la especificación inicial del sistema y la transforma en un conjunto estructurado de objetivos del sistema. Los objetivos son utilizados para encapsular los requerimientos porque contienen lo que el sistema trata de alcanzar.

Las Figuras A.31a, A.31b y A.31c muestran los objetivos que satisfacen los requerimientos de transporte gestionados por el sistema *DRT*. La Figura A.31a muestra los objetivos del sistema que gestionan información relacionada con los clientes y las solicitudes, la Figura A.31b muestra el objetivo del sistema que gestiona información relacionada con los vehículos, y por último, la Figura A.31c muestra el objetivo del sistema que gestiona información relacionada con las rutas.

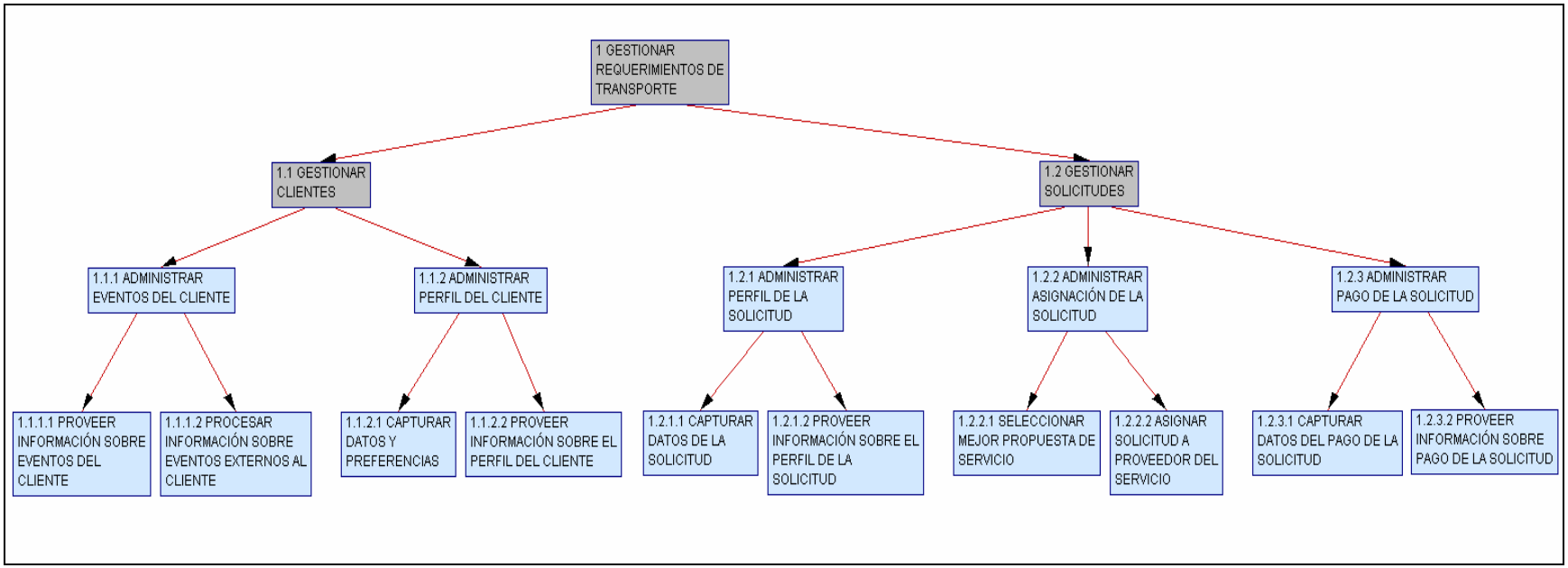

Figura A.31a Objetivos 1.1 y 1.2 del diagrama de jerarquía de objetivos.

La Figura A.31a muestra los objetivos que gestionan información relacionada con los clientes (objetivo *1.1 Gestionar Cliente*) y las solicitudes de servicio de transporte (objetivo *1.2 Gestionar Solicitudes*). Ambos objetivos son particionados, esto quiere decir que no serán implementados como roles en las fases posteriores, pero sí lo serán sus sub-objetivos.

El objetivo *Gestionar Cliente* debe ser capaz de mantener informados a los clientes sobre los eventos que ocurran durante el proceso de petición de solicitudes (sub-objetivo *Administrar Eventos del Cliente*), y de administrar toda la información referente a sus datos personales y preferencias (sub-objetivo *Administrar Perfil del Cliente*). Por otra parte, el objetivo *Gestionar Solicitudes* debe ser capaz de proveer y mantener información referente a la solicitud del servicio de transporte (sub-objetivo *Administrar Perfil de la Solicitud*), de administrar el proceso de selección y asignación del proveedor del servicio de transporte (sub-objetivo *Administrar Asignación de la Solicitud*), y de proveer y mantener información referente al pago de la solicitud (sub-objetivo *Administrar Pago Solicitud*).

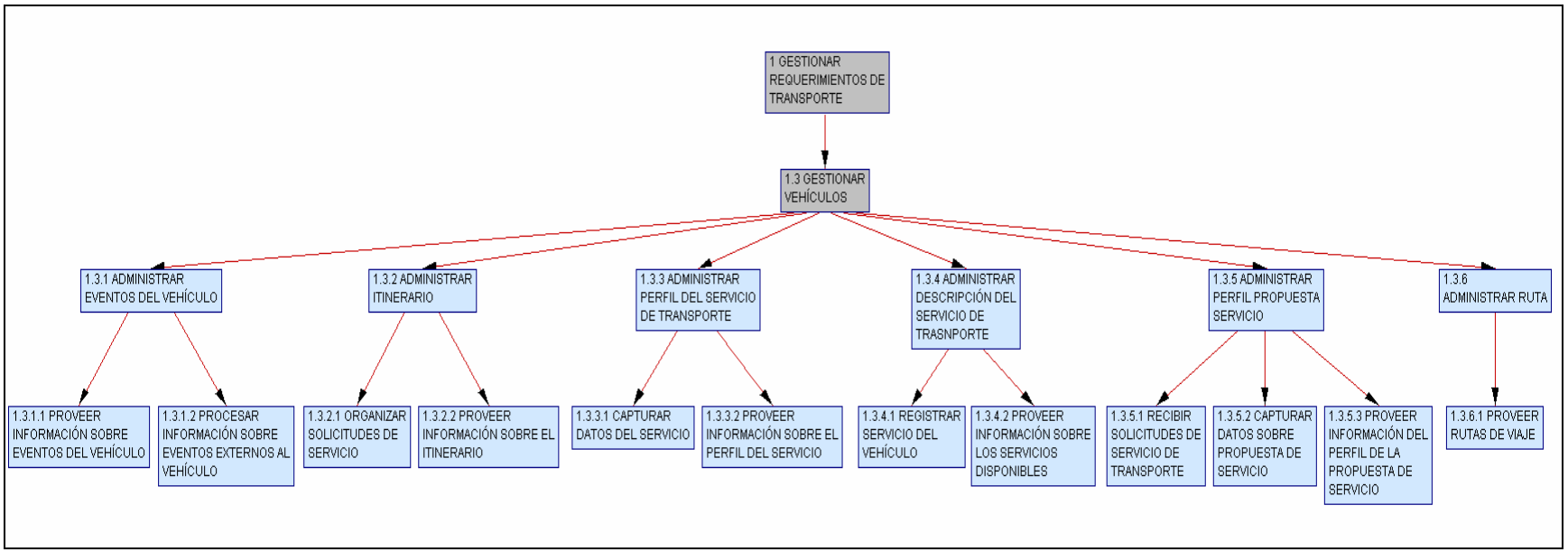

Figura A.31b Objetivo 1.3 del diagrama de jerarquía de objetivos.

La Figura A.31b muestra el objetivo encargado de gestionar información relacionada con los vehículos (objetivo *1.3 Gestionar Vehículos*). Al igual que los objetivos anteriores, este objetivo es particionado, lo cual indica que sólo sus sub-objetivos serán implementados en las fases posteriores.

El objetivo *Gestionar Vehículos* debe ser capaz de mantener informados a los conductores de los vehículos sobre los eventos que ocurran durante el proceso de solicitud del servicio y/o el viaje (sub-objetivo *Administrar Eventos del Vehículo*), de proveer y mantener información sobre el itinerario del vehículo que presta el servicio (sub-objetivo *Administrar Itinerario*), de proveer y mantener información referente al perfil del servicio de transporte (sub-objetivo *Administrar Perfil del Servicio de Transporte*), de registrar y proveer información sobre la descripción del servicio (sub-objetivo *Administrar Descripción del Servicio de Transporte*), de recibir, enviar y mantener información referente al perfil de la propuesta del servicio (sub-objetivo *Administrar Perfil Propuesta Servicio*), y de proveer información referente a la ruta (sub-objetivo *Administrar Ruta*).

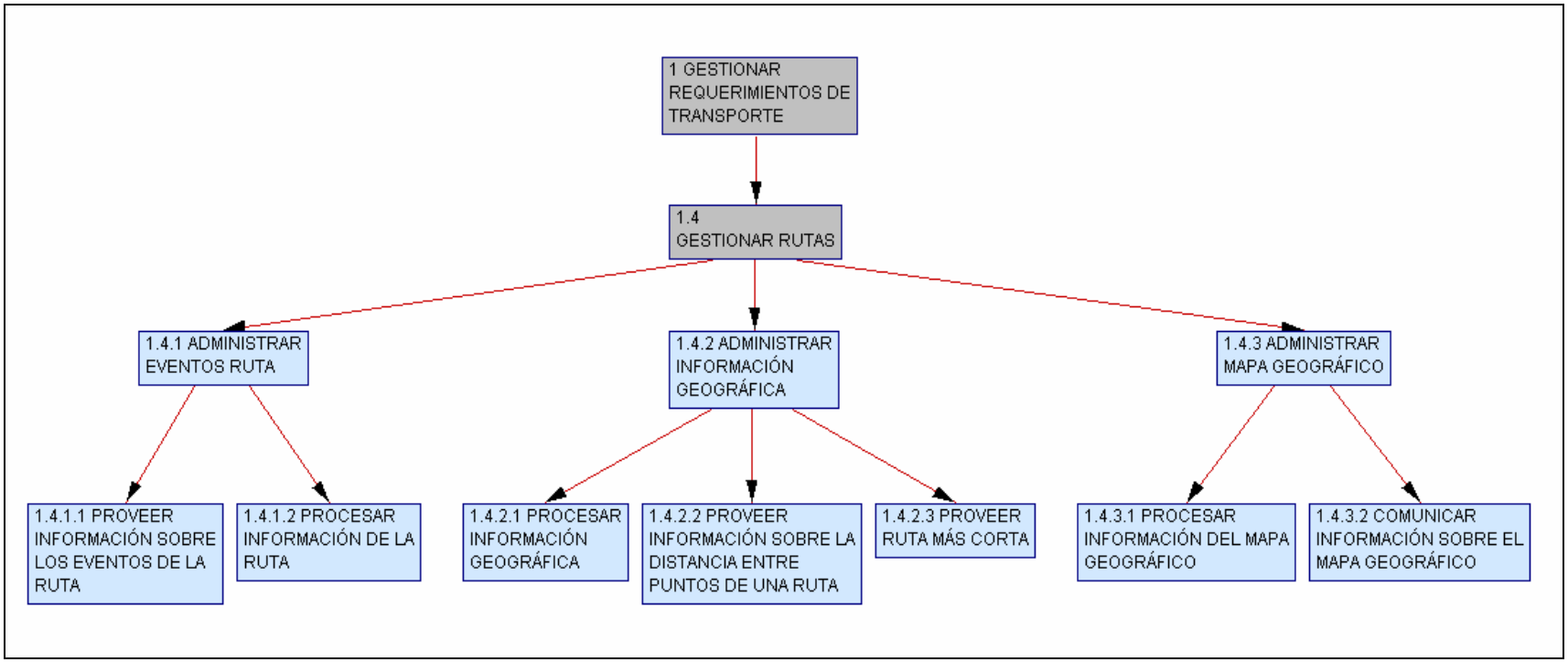

Figura A.31c Objetivo 1.4 del diagrama de jerarquía de objetivos.

La Figura A.31c muestra el objetivo encargado de gestionar información relacionada con las rutas (objetivo *1.4 Gestionar Rutas*). Al igual que los objetivos anteriores, este objetivo es particionado, lo cual indica que sólo sus sub-objetivos serán implementados en las fases posteriores.

El objetivo *Gestionar Rutas* debe ser capaz de proveer y procesar información referente a los eventos que afectan a una ruta (sub objetivo *Administrar Eventos Ruta*), de procesar información geográfica sobre la ruta, de proveer información sobre las distancias entre puntos de una ruta y proveer la ruta más corta entre dos puntos (sub-objetivo *Administrar Información Geográfica*), y por último, procesar y comunicar información sobre el mapa geográfico (sub-objetivo *Administrar Mapa Geográfico*).

A.2.2 Transformar objetivos a roles

En esta fase los objetivos definidos en la fase anterior son transformados en roles y sus tareas asociadas. Cada uno de los objetivos están asociados a un rol y cada uno de los roles es ejecutado o representado por una clase de agentes.

En la Figura A.32 se puede apreciar que para alcanzar el sub-objetivo *Administrar Eventos Clientes*, se crea el rol *Administrador de Eventos Cliente*, el cual debe realizar las tareas de *Comunicar Evento Cliente*, *Notificar Evento Externo Cliente* y *Procesar Eventos*.

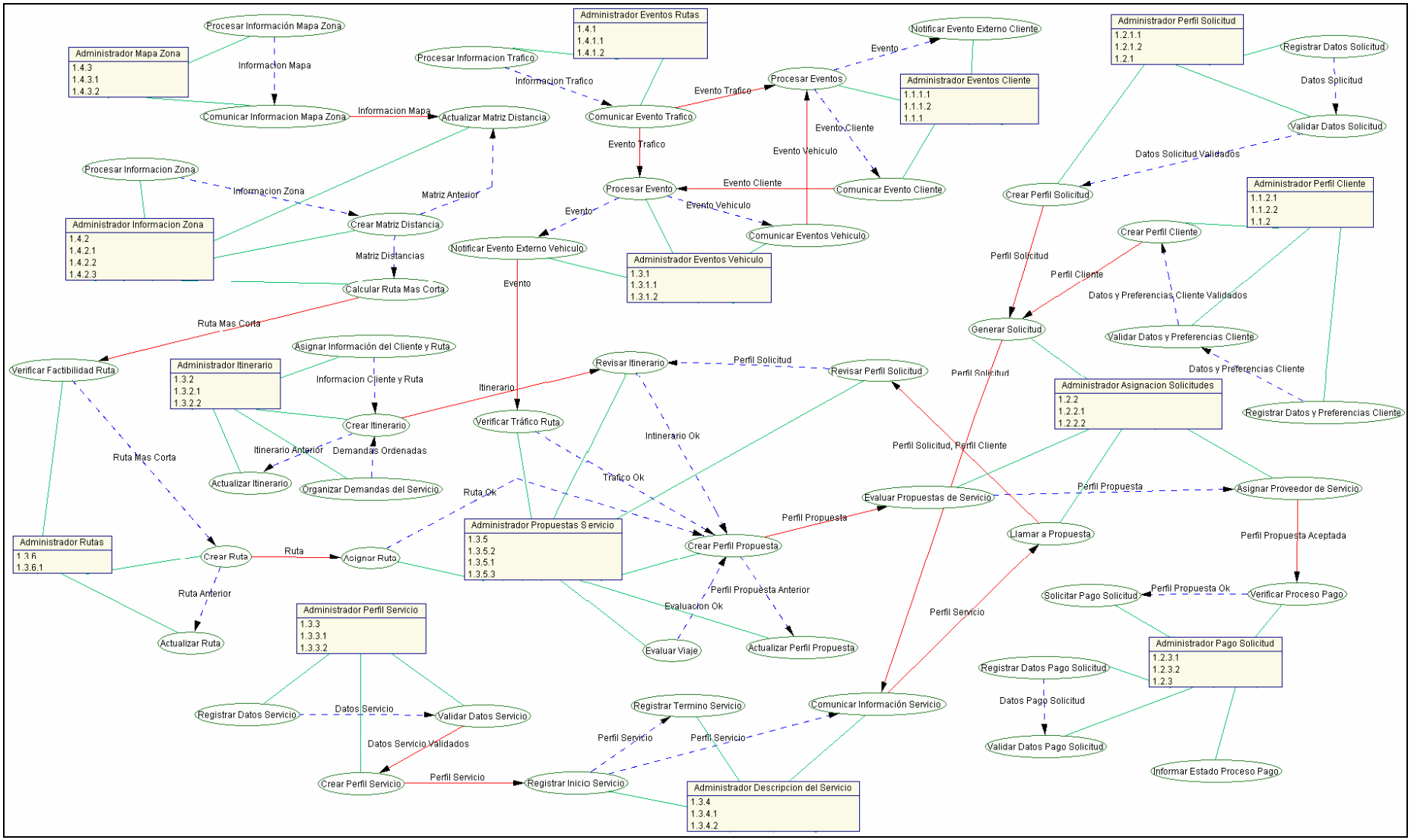

Figura A.32 Diagrama de roles.

A.2.3 Aplicar casos de uso

El propósito de esta fase es apoyar la posterior construcción de conversaciones en la metodología. Aplicar los casos de uso requiere tomar los casos de uso identificados en la *Captura de Objetivos* y reestructurarlos como diagramas de secuencia. El diagrama de secuencia es utilizado para determinar el conjunto mínimo de mensajes que deben intercambiar los roles. Si un mensaje es traspasado entre dos roles, entonces debe existir la correspondiente comunicación entre ellos. Además, se utilizan para identificar los roles del sistema que participan en los eventos. Cada participante en los diagramas de secuencia es un rol; por lo tanto, si hay un participante en los casos de uso que no es un rol en el sistema, se debe crear un nuevo rol.

A continuación, se presentarán diagramas de secuencia para los siguientes escenarios:

- **R.Id 1 El cliente comunica la cancelación o retraso del viaje:** El cliente informa que sufrió un retraso o que cancelará su viaje, para lo cual debe elegir del listado de sus viajes pendientes aquel a cancelar y luego enviar un aviso sobre esta situación. En el itinerario del vehículo se cancelará el viaje y se notificará dicha cancelación al conductor del vehículo y al agente vehículo. La Figura A.33 muestra el diagrama de este escenario.

- **R.Id 2 El conductor comunica la cancelación, retraso o falla del vehículo:** Si ocurre una falla del vehículo o se cancela el servicio, el conductor debe dar aviso para cancelar el perfil de servicio del vehículo, recibiendo la confirmación de la operación. Luego envía las cancelaciones de los viajes programados en su itinerario actual, las cuales deberán ser posteriormente comunicadas a los clientes afectados. Para el caso en que ocurra un retraso por parte del vehículo, el conductor envía el correspondiente aviso recibiendo la confirmación de la operación. El sistema evalúa las solicitudes afectadas y eventualmente podría sugerir, ya sea una nueva solicitud o una solución alternativa (por ejemplo una nueva ruta). La Figura A.34 muestra el diagrama de este escenario.

- **R.Id 3 El sistema de información de tráfico comunica la ocurrencia de un evento en la ruta:** Una vez procesados los eventos, que pueden ser congestión o desvío de la ruta de viaje, son enviados a los vehículos los puntos de cobertura de la empresa de transporte que se ven afectados, junto con el margen de tiempo que estos puntos permanecerán bloqueados. Además, el conductor debe recibir los cambios que estos eventos hayan producido en su itinerario actual, los cuales también deberán ser informados a los clientes respectivos. La Figura A.35 muestra el diagrama de este escenario.

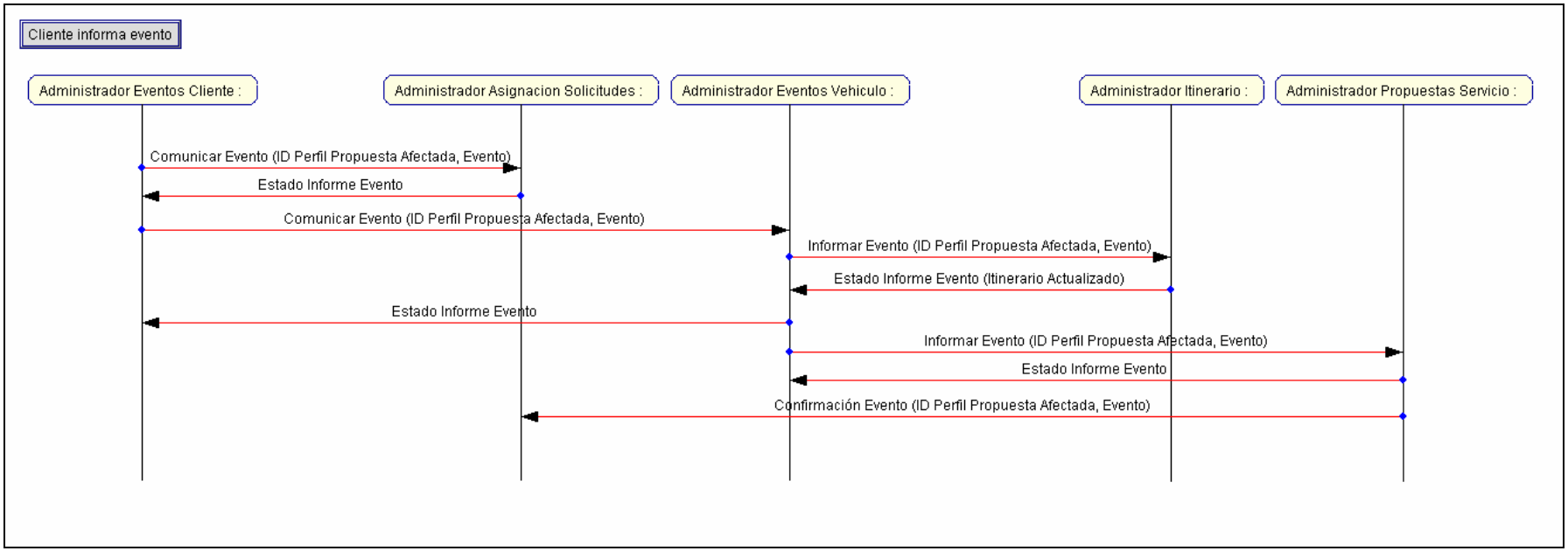

Figura A.33 R.Id 1, cliente comunica cancelación/retraso viaje.

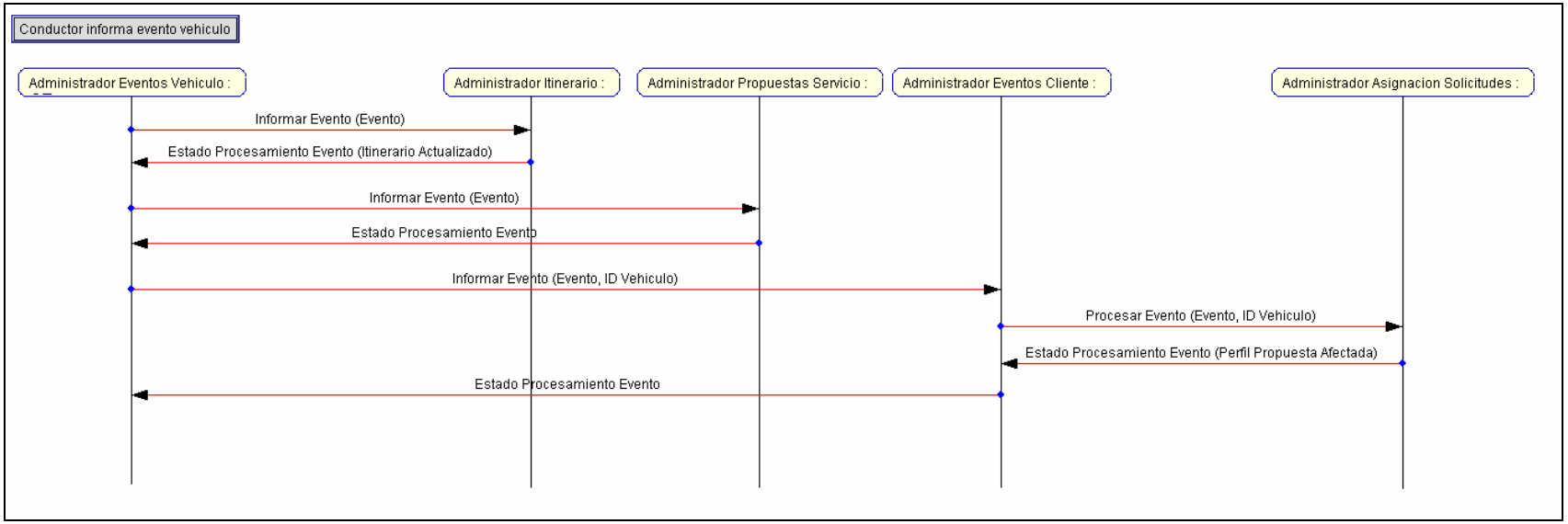

Figura A.34 R.Id 2, conductor comunica cancelación/retraso/falla del vehículo.

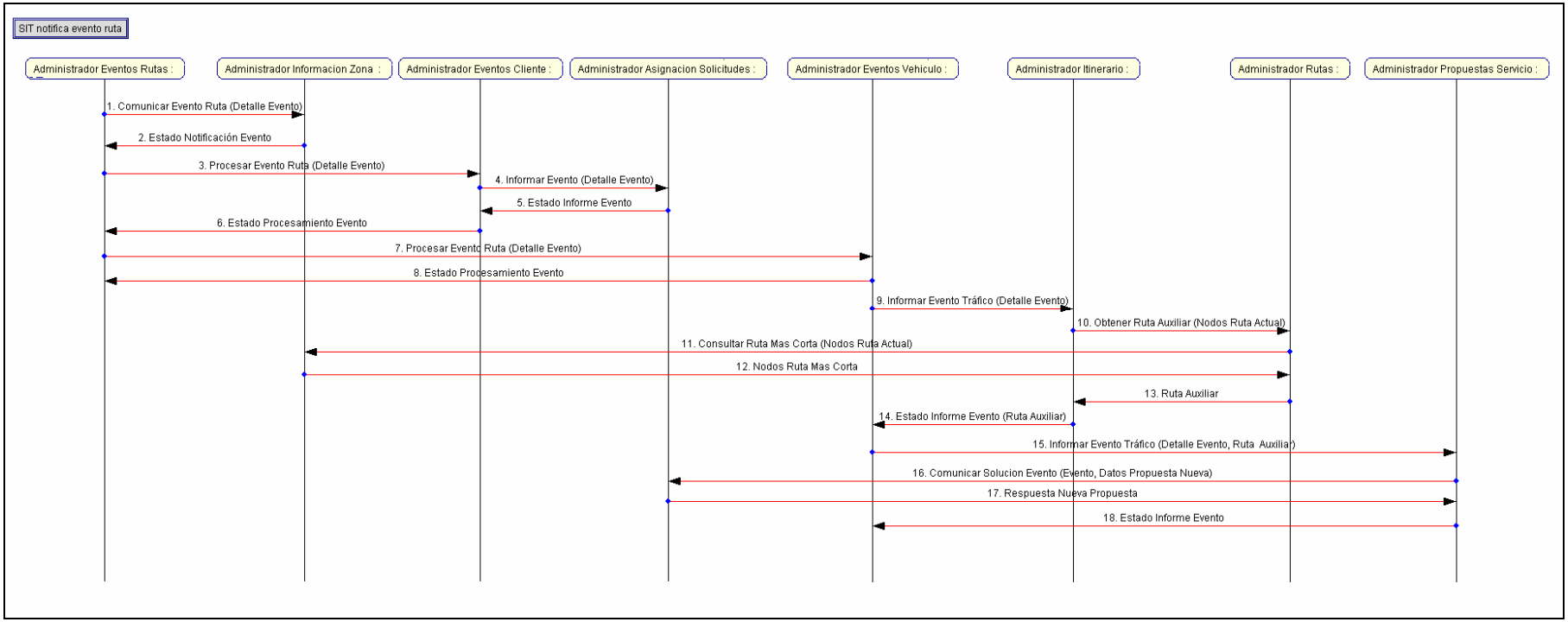

Figura A.35 R.Id 3, sistema de información de tráfico comunica evento en la ruta.

A.2.4 Crear clases de agente

En esta fase se crean las clases de los agentes pertenecientes a la solución. De esta fase se obtiene el *Diagrama de Clases de Agentes* (ver Figura A.36) que es muy similar al *Diagrama de Objetos*, aunque las relaciones en el primero se interpretan como conversaciones.

Como se muestra en la Figura A.36, el agente *GUI Cliente* mantiene diversas conversaciones con el agente *Cliente*. Primero, existe una conversación que representa la petición de solicitud de transporte de parte del cliente (conversación *Solicitud De viaje*). Una vez que ocurre esto, el agente *Cliente* conversa con el agente *Broker* para pedirle el listado de servicios disponibles que se ajustan a la solicitud del cliente. Luego de que el *Broker* envía la lista de servicios disponibles que cumplen con el perfil de la solicitud, el agente *Cliente* comienza el proceso de negociación con el agente *Vehículo*. El agente *Cliente* informa al agente *GUI Cliente*, para que éste informe al cliente todas las propuestas de servicio. Una vez que el cliente le comunica su decisión al *GUI Cliente*, éste le informa al *Cliente* el cual cierra la negociación con el respectivo agente *Vehículo*. Por último, el *Cliente* le envía la confirmación de viaje al agente *GUI Cliente*.

Por otra parte, es posible ver también en el diagrama los roles que desempeñan cada uno de los agentes que componen el sistema. Por ejemplo, el agente *Cliente* cumple los roles de *Administrador Asignación Solicitud* y *Administrador Pago Solicitud*.

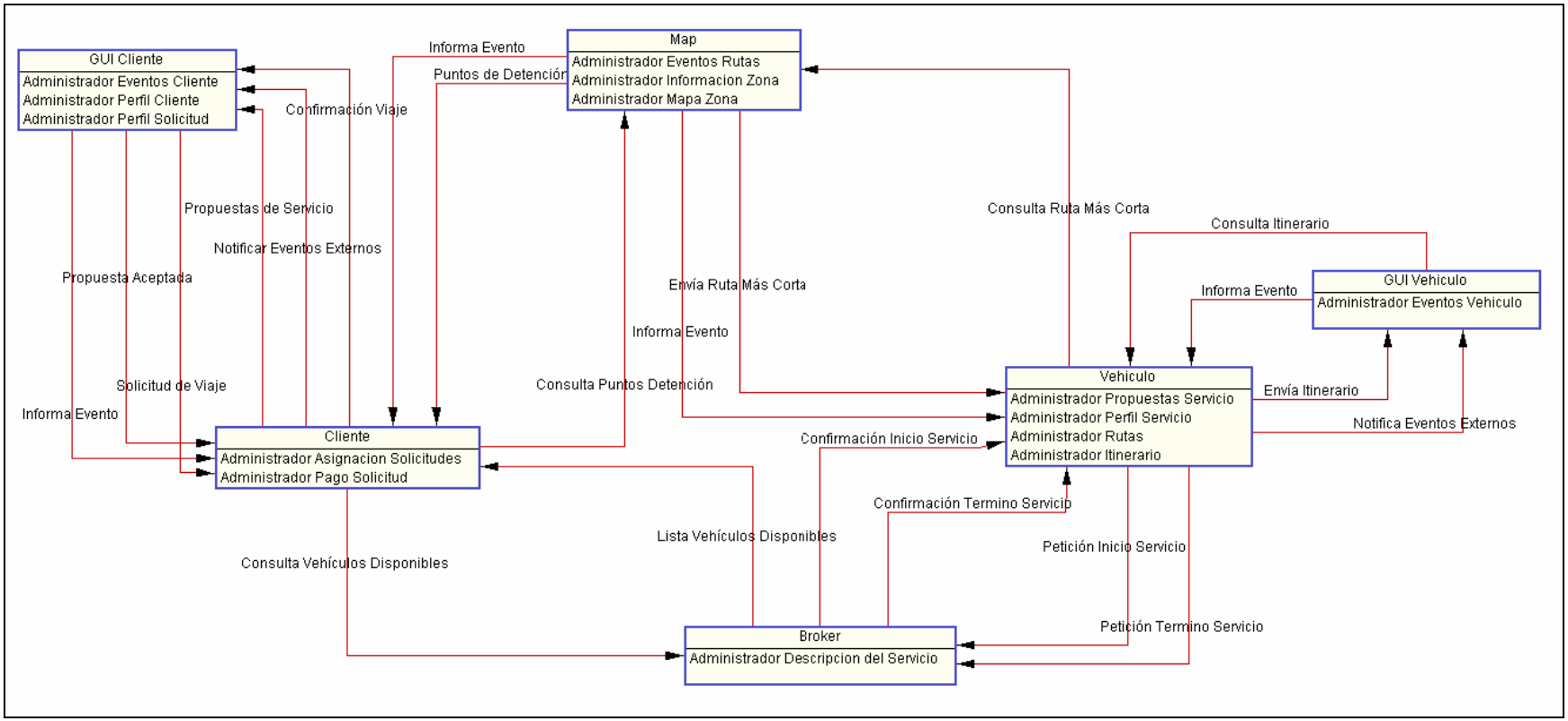

Figura A.36 Diagrama de clases de agente.

A.2.5 Construir conversaciones

En esta fase se utiliza el *Diagrama de Transición de Estados*, el cual modela la transición de estados de un agente y el estado de la conversación entre dos agentes. Para cada comunicación entre agentes existe una conversación, la cual define un protocolo de coordinación entre dos agentes. Específicamente, una conversación consiste en dos diagramas de transición de estados, uno para el agente que inicia la conversación y otro para el que responde.

En las Figuras A.37 y A.38 podemos ver los *Diagramas de Transición de Estados* de la conversación entre el agente *Cliente* y *GUI Cliente*.

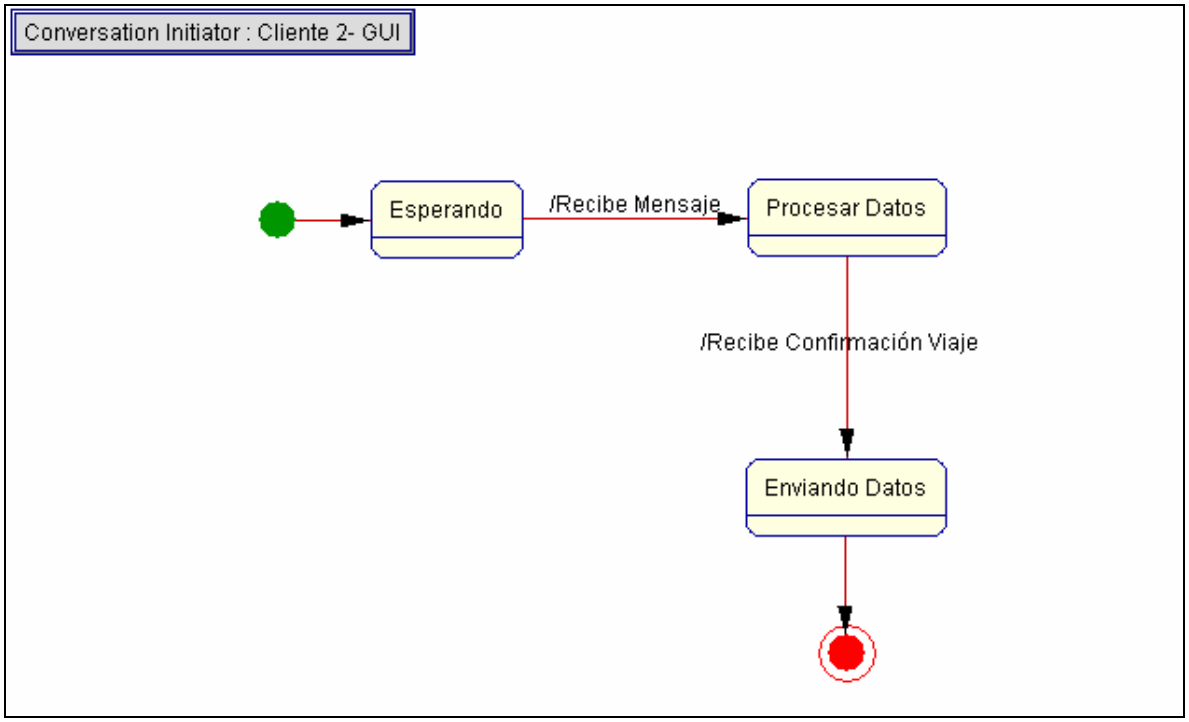

Figura A.37 Transición de estados agente Cliente para la conversación Cliente2-GUI Cliente.

La Figura A.37 muestra el *Diagrama de Transición de Estados* del agente *Cliente* para la conversación *Cliente2-GUI Cliente*. El agente *Cliente* solicita al agente *GUI Cliente* la confirmación del viaje para su posterior procesamiento.

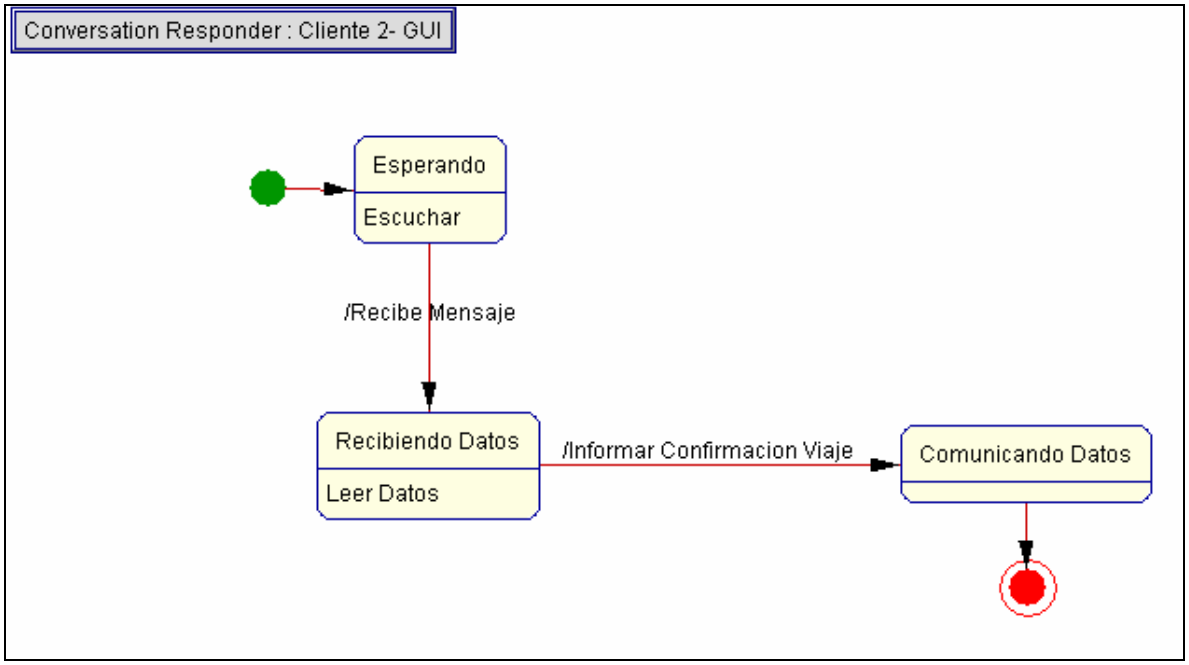

Figura A.38 Transición de estados agente GUICliente para la conversación Cliente2-GUI Cliente.

La Figura A.38 muestra el *Diagrama de Transición de Estados* del agente GUI *Cliente* para la conversación *Cliente2-GUI Cliente*. El agente GUI *Cliente* pide la confirmación del viaje al cliente y luego envía la respuesta al agente *Cliente*.

A.2.6 Ensamblado de clases de agentes

En esta fase se define la arquitectura de los agentes mediante componentes. Es importante destacar que existe un diagrama de arquitectura para cada agente, pero no existe uno que muestre la arquitectura del sistema completo.

La Figura A.39 muestra la arquitectura del agente *Broker*. En ella se aprecia que el agente posee tres componentes: los componentes *Registrar Inicio Servicio* y *Registrar Termino Servicio*, los cuales se comunican con el componente *Comunicar Información Servicio* para efectuar el inicio/término del servicio de transporte respectivamente.

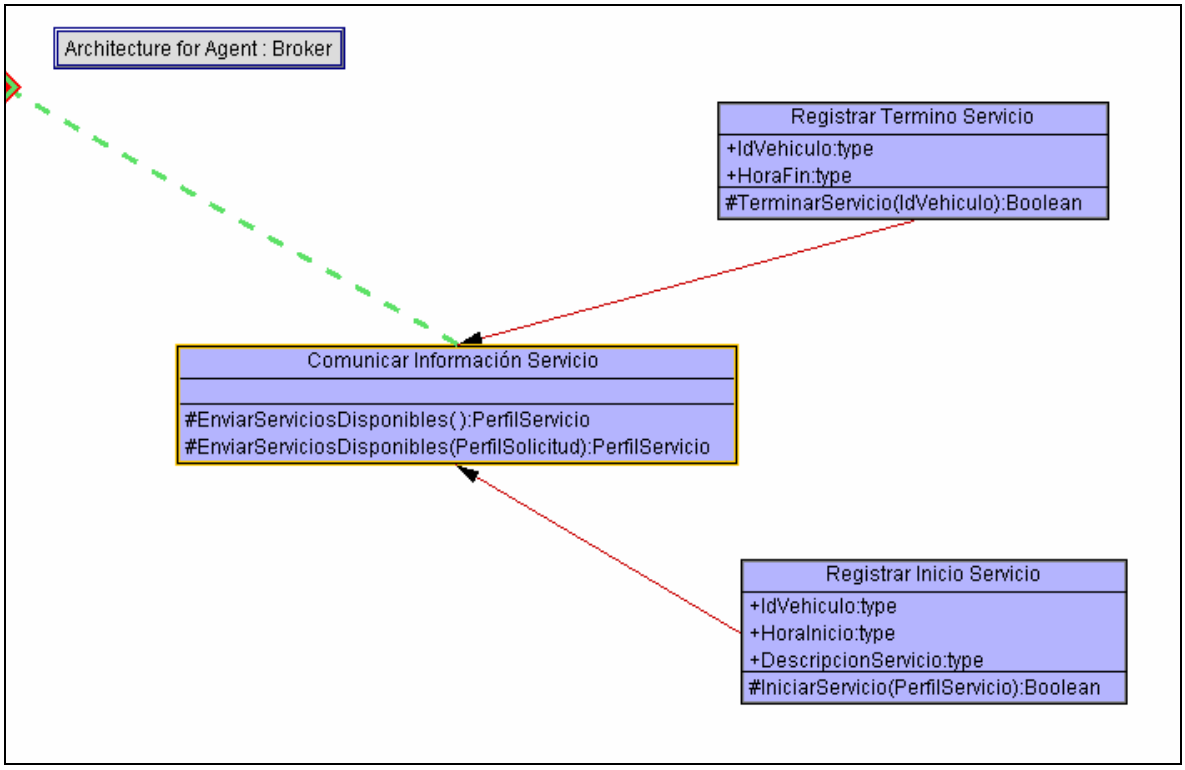

Figura A.39 Diagrama de arquitectura agente Broker.

A.2.7 Despliegue del sistema

La fase *Despliegue del Sistema* es un modelo de distribución de las partes del sistema, a través de las unidades de hardware de procesado.

Como se puede apreciar en la Figura A.40, el sistema *DRT* es un sistema distribuido en donde, por ejemplo, los agentes *GUI Vehículo* y *GUI Cliente* puede habitar en entornos distintos del resto de los agentes.

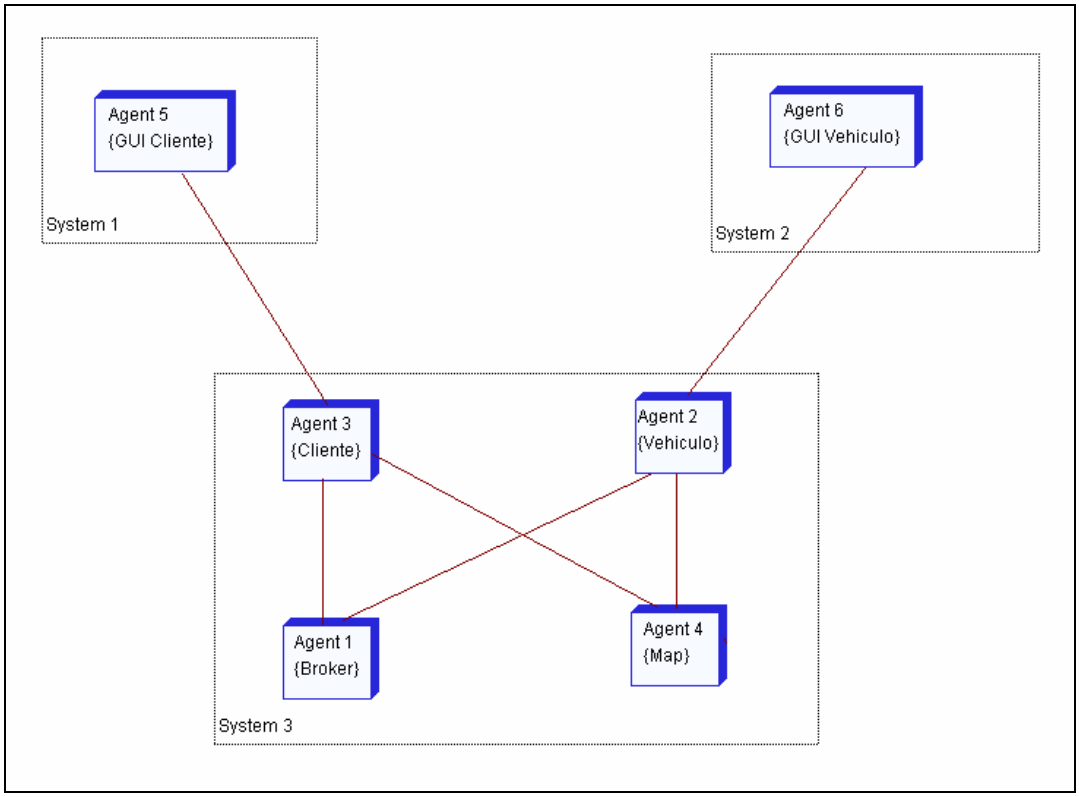

Figura A.40 Diagrama de despliegue DRTS.