



UNIVERSIDAD NACIONAL DEL LITORAL

Facultad de Ingeniería y Ciencias Hídricas

**SOPORTE COMPUTACIONAL PARA LA GESTIÓN DE  
EVENTOS DISRUPTIVOS EN LA CADENA DE  
SUMINISTRO**

**Ing. Lorena Andrea Bearzotti**

Tesis remitida al Comité Académico del Doctorado  
como parte de los requisitos para la obtención  
del grado de  
DOCTOR EN INGENIERIA  
Mención Mecánica Computacional  
de la  
UNIVERSIDAD NACIONAL DEL LITORAL

**2012**

Comisión de Posgrado, Facultad de Ingeniería y Ciencias Hídricas, Ciudad Universitaria, Paraje “El  
Pozo”,

S3000, Santa Fe, Argentina

UNIVERSIDAD NACIONAL DEL LITORAL

# **Soporte Computacional para la Gestión de Eventos Disruptivos en la Cadena de Suministro**

Lorena Andrea Bearzotti

**FICH**

FACULTAD DE INGENIERIA

Y CIENCIAS HIDRICAS

**INTEC**

INSTITUTO DE DESARROLLO TECNOLOGICO

PARA LA INDUSTRIA QUIMICA

Tesis de Doctorado 2012

III

**Doctorado en Ingeniería**

**Mención Mecánica Computacional**

Título de la obra:

**Soporte Computacional para la Gestión de Eventos  
Disruptivos en la Cadena de Suministro**

Autor: Lorena Andrea Bearzotti

Lugar: Santa Fe, Argentina

Palabras Claves:

Cadenas de suministro, gestión de la cadena de suministro,  
gestión de eventos, sistemas autónomos, sistemas complejos.

**Soporte Computacional para la Gestión de Eventos Disruptivos en la Cadena de  
Suministro**

Tesista: Ing. Lorena Andrea Bearzotti Pilomeno

Director: Dr. Omar Chiotti

Co-Director: Dr. Héctor Enrique Salomone

2012



### ACTA DE EVALUACIÓN DE TESIS DE DOCTORADO

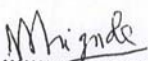
En la sede de la Facultad de Ingeniería y Ciencias Hídricas de la Universidad Nacional del Litoral, a los veintisiete días del mes de abril del año dos mil doce, se reúnen los miembros del Jurado designado para la evaluación de la Tesis de Doctorado en Ingeniería titulada "Soporte computacional para la gestión de eventos disruptivos en la cadena de suministro", desarrollada por la Ing. Lorena Andrea BEARZOTTI, DNI N° 23.929.712. Ellos son la Dra. N. Beatriz Brignole, el Dr. Oscar Iribarren, el Dr. Carlos Méndez y el Dr. Pablo Miranda.


Luego de escuchar la Defensa Pública y de evaluar la Tesis, el Jurado resuelve:

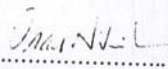
- La tesis presenta una valiosa contribución con una fuerte proyección.
  - Se destaca el Trabajo de carácter inter-disciplinario.
  - Se destaca la calidad, claridad y precisión de la escritura de la Tesis y su presentación.
  - La tesis disertó con claridad y respondió satisfactoriamente las preguntas del jurado.
- Se otorga la calificación:

Aprobada con correcciones menores oportunamente indicadas, con calificación 10 (sobresaliente).

Sin más, se da por finalizado el Acto Académico con la firma de los miembros del Jurado al pie de la presente

  
Dra. N. Beatriz Brignole

  
Dr. Carlos Méndez

  
Dr. Oscar Iribarren

  
Dr. Pablo Miranda

Universidad Nacional del Litoral      Ciudad Universitaria  
Facultad de Ingeniería y      C.C. 217  
Ciencias Hídricas      Ruta Nacional N° 168 - Km. 472,4  
      (3000) Santa Fe  
Secretaría de Posgrado      Tel: (54) (0342) 4575 229  
      Fax: (54) (0342) 4575 224

## Dedicatoria

A todos los que me acompañaron en este proceso, especialmente a mi familia, a mi Bicha, a Marce... y a mis amigos.

## AGRADECIMIENTOS

No me alcanzan las palabras para dar las gracias a todos aquellos que estuvieron a mi lado ayudándome a cerrar este capítulo de mi vida.

A mis directores Omar y Enrique, gracias por su guía, su apoyo, sus consejos, por darme ese impulso cuando las fuerzas me fallaron.

Mis queridos principitos, no me podía olvidar de ustedes... fueron mis compañeros de estudio, de espacio, de tardes compartidas con mate y alguna especialidad culinaria y varias anécdotas... esta "principita" les da las gracias, Juanjo y Leo mis hermanos rebeldes, Mariana, Gabriela y Roxana mis grandes compañeras cuyos consejos fueron invaluable; Leila y Susana con quienes en distinto períodos compartí la oficina y tuvieron que sobrevivir a mis gustos decorativos...

Román y Sebastián por ser parte de este proceso al realizar una herramienta de gran utilidad, la cual concretó nuestras ideas.

José... gracias! Por ser mi amigo, por estar a mi lado en las buenas y en las malas, en todo momento sin importar los tiempos alejados o la distancia que nos separan, estamos juntos y cada reencuentro es maravilloso... gracias por seguir en mi vida y sinceramente espero que nuestra amistad perdure en el tiempo.

También le quiero dar las gracias a Max, por las conversaciones que hemos tenido y que tanto bien me han hecho, por los consejos, por la paciencia, por el aliento.

A mis amigas que la danza ha traído a mi vida: Silvia, Caro y Trudet... le doy las gracias a la vida de que eso sucediera, no solo descubrí la magia de la danza sino también la alegría de una amistad hermosa.

No puedo olvidarme de Hilda... Hilda, sin tu colaboración, sin tu ayuda, sin tu mano dispuesta a reemplazarme cuando me toca viajar, todo este proceso habría sido casi imposible... gracias por estar ahí haciéndote cargo, con toda la responsabilidad que eso significa, de mi bisabuela.

A mi querido Padrino y a Graciela por darme la sensación de hogar en mis visitas a Santa Fe.

Un gracias muy especial a Cristina... tus palabras Cris, me dieron mucha luz y fuiste una guía muy especial... fuiste una gran amiga de mi abuela y gracias a Dios hoy te puedo llamar amiga yo también.



Mamá... estoy muy orgullosa de tus logros, gracias por darme el ejemplo de seguir en la lucha, de aceptar los nuevos desafíos, de no bajar los brazos. A mis hermanas, Gaby, Ro, Pame, Marie, Aimé y Brisa... gracias chicas por las palabras de aliento, por ayudarme, por colaborar... y a veces simplemente por estar! Nunca olviden que las quiero y que siempre podrán contar conmigo. Y hablando de la familia, también tengo que darle las gracias a Miguel por haberme aceptado como su hija.

A mis abuelos, Anita y Alfonso... mejor dicho Tota y Toto... no están conmigo desde hace muchos, muchos años, sin embargo si hoy estoy acá escribiendo y finalizando este ciclo es por ustedes, por las semillas que ustedes sembraron... éste es un fruto para mí pero sin ustedes no lo podría haber cosechado... gracias, a los dos, por haber aceptado el desafío de criar a esta, la que alguna vez fuera, una muchachita inquieta. El tiempo todo lo cura, excepto la ausencia de un ser querido... sigan a mi lado.

¿Cómo te pudo dar las gracias Bicha? Sencillamente no puedo, porque no existen las palabras que me permitan agradecerte todos y cada uno de estos años de compañía, cuidado y amor... GRACIAS... gracias mi Bicha...

Marce, simplemente GRACIAS... simplemente TE AMO... y citando a Antonine de Saint-Exupéry “amar no es mirarse el uno al otro, sino mirar los dos en la misma dirección”, caminemos juntos en la dirección que hemos elegido.

Gracias a todos.

# INDICE GENERAL

Indice general .....	11
Indice de figuras .....	15
Indice de tablas .....	18
Resumen .....	20
Abstract .....	21
Capítulo 1 – Introducción .....	22
1.1. La Cadena de Suministro .....	22
1.2. La Gestión de la Cadena de Suministro .....	23
1.3. Eventos Disruptivos .....	24
1.4. Gestión de Eventos en la Cadena de Suministro .....	27
1.5. Sistemas SCCEM .....	28
1.5.1. Niveles de Automatización .....	28
1.5.2. SCCEM como un sistema de control complejo .....	30
1.6. Problema a Resolver .....	32
1.7. Trabajos relacionados: Estado del Arte .....	32
1.8. Objetivo e Hipótesis de la Tesis .....	35
Capítulo 2 – Un modelo conceptual de un sistema para la gestión de eventos disruptivos en la cadena de suministro .....	39
2.1. Modelo de componentes principales .....	39
2.2. Modelo funcional del sistema SCCEM .....	41
2.3. Modelo conceptual del Sistema SCCEM .....	43
2.3.1. Definición del Modelo Conceptual .....	44
2.3.2. Los elementos del modelo conceptual .....	46
2.3.3. La interacción en el modelo conceptual .....	53
2.3.3.1 La interacción entre componentes .....	54

2.3.3.2 La interacción entre elementos de la red .....	55
2.3.3.3 El proceso de Coordinación .....	57
2.4. Conclusiones .....	67
Capítulo 3 – Arquitectura de un sistema SCEM basado en agentes.....	69
3.1. Arquitectura Multi-Agente para SCEM .....	69
3.2. Arquitectura de los agentes .....	71
3.2.1. Agente PAGE.....	71
3.2.2. Agente EVA .....	74
3.2.3. Agente IOA .....	78
3.2.4. Agente RKU.....	80
3.2.5. Agente MKU.....	84
3.2.6. Agente SP.....	87
3.3. Proceso de coordinación.....	91
3.3.1. Protocolo de interacción.....	92
3.3.2. Modelo de Negociación .....	96
3.4. Conclusiones .....	98
Capítulo 4 – Prototipo de un sistema SCEM multi-agente .....	100
4.1. Plataforma de Desarrollo.....	100
4.2. Implementación de los agentes .....	101
4.2.1. Agente PAGE.....	101
4.2.2. Agente EVA .....	107
4.2.3. Agente RKU.....	109
4.2.4. Agente SP.....	115
4.3. Implementación de la Ontología .....	118
4.3.1. Ontologías en JaDE.....	118
4.3.2. Creación de las ontologías en JADE .....	120
4.3.3. Documento de negocio de SCEMS .....	121

4.6. Conclusiones .....	123
Capítulo 5 –Caso de estudio.....	125
5.1. Metodología .....	125
5.2. Caso de Estudio.....	125
5.3. Modelado del Caso de estudio .....	127
5.3.1 Modelado del programa de abastecimiento del proveedor.....	127
5.3.2 Modelado del programa de abastecimiento del distribuidor .....	133
5.3.3 Modelado del programa de abastecimiento del Minorista 1 .....	138
5.3.4 Modelado del programa de abastecimiento del Minorista 2 .....	140
5.4. Analisis de resultados.....	142
5.4.1. Proceso de Instanciación .....	142
5.4.2. Escenarios de Prueba.....	143
5.5. Conclusiones .....	148
Capítulo 6 – Conclusiones y Futuros Trabajos .....	150
6.1. Conclusiones .....	150
6.2. Futuros Trabajos.....	153
Glosario y Listado de Símbolos .....	154
ANEXO– Ejecución del SCEMS.....	157
A.1. SCEMS.....	157
A.2. Ejecución del SCEMS .....	158
A.3. Configuración de escenarios de prueba.....	161
A.4. Ejecución de un escenario de prueba.....	161
A.5. Resultado de una ejecución .....	162
A.6. Generación del reporte de ejecución .....	162
A.7. Configuración de los Agentes .....	163
A.7.1. Configuración de la agencia .....	164
A.7.2. Configuración de los agentes RKU .....	165

A.7.3. Configuración de los agentes SP .....	168
A.7.4. Definición de la secuencia de eventos.....	170
Referencias.....	172

## INDICE DE FIGURAS

Figura 1.1: Representación gráfica de la variable de estado .....	31
Figura 2.1: Componentes principales de un sistema SCeM .....	40
Figura 2.2: Modelo Funcional para el proceso SCeM integrado (Empresas A y B) .....	41
Figura 2.3: Cadena de suministro. Empresas A (producción) Empresa B (distribución y venta) .....	45
Figura 2.4: Red de puntos de control del ejemplo.....	46
Figura 2.5: Definición del Modelo Conceptual.....	47
Figura 2.6: Perfil de Requerimientos y Perfil de Disponibilidad para el RKU .....	48
Figura 2.7: Perfil de Factibilidad para el RKU .....	48
Figura 2.8: Perfil de Disponibilidad actualizado.....	49
Figura 2.9: Perfil de Factibilidad actualizado .....	49
Figura 2.10: Perfil de inventario del MKU-m1g-silo.....	50
Figura 2.11: Perfil de inventario del MKU-m1e-almacén .....	51
Figura 2.12: Red de MKUs, RKUs y SP del ejemplo .....	53
Figura 2.13: Red con IOA del ejemplo .....	55
Figura 2.14: Lógica del proceso de coordinación .....	58
Figura 2.15: PRO1: Perfil de Requerimientos y de Disponibilidad para el RKU .....	60
Figura 2.16: PRO1: Perfil de Factibilidad para el RKU.....	60
Figura 2.17: PRO1: Perfil de inventario del MKU-m1g-silo.....	61
Figura 2.18: PRO1: Perfil de inventario del MKU-m1e-almacén.....	61
Figura 2.19: PRO2: Perfil de Requerimientos y de Disponibilidad para el RKU .....	62
Figura 2.20: PRO2: Perfil de Factibilidad para el RKU.....	62
Figura 2.21: PRO2: Perfil de inventario del MKU-m1g-silo .....	63
Figura 2.22: PRO2: Perfil de inventario del MKU-m1e-almacén.....	63
Figura 2.23: PRO4: Perfil de Requerimientos y de Disponibilidad para el RKU .....	64

Figura 2.24: PRO4: Perfil de Factibilidad para el RKU.....	64
Figura 2.25: PRO5: Perfil de inventario del MKU-m1e-almacén.....	65
Figura 2.26: PRO5: Perfil de inventario del MKU-m1e-almacén.....	66
Figura 3.1: Arquitectura basada en agentes de un sistema SCEM: Componentes e Interacciones .....	71
Figura 3.2: Diagrama de clases AUML de un agente PAGE.....	74
Figura 3.3: Diagrama de clases AUML para el agente EVA.....	77
Figura 3.4: Diagrama de clases AUML para el agente IOA.....	80
Figura 3.5: Diagrama de clases AUML del agente RKU.....	83
Figura 3.6: Diagrama de clases AUML para el agente MKU.....	87
Figura 3.7: Diagrama de clases AUML de un agente SP.....	91
Figura 3.8: Diagrama de Secuencia para el protocolo DCN.....	95
Figura 4.1: Diagrama de clases para el agente AP (PAGE).....	102
Figura 4.2: Ejemplo de un archivo “agentes.xsd”.....	106
Figura 4.3: Diagrama de clases para el agente AE.....	108
Figura 4.4: Ejemplo de archivo XML de configuración de eventos.....	109
Figura 4.5: Estrategias de negociación.....	111
Figura 4.6: Extracto de un archivo “estrategias.xml”.....	113
Figura 4.7: Diagrama de clases para calculadores de utilidades.....	113
Figura 4.8: Extracto de un archivo “calculadoresUtilidad.xml”.....	114
Figura 4.9: <i>Behaviours</i> del agente RKU.....	115
Figura 4.10: Extracto de un archivo “UnificadoresPropuestas.xml”.....	116
Figura 4.11: Diagrama de clases para los <i>Unificadores</i> de propuestas.....	117
Figura 4.12: <i>Behaviours</i> del agente SP.....	118
Figura 4.13: Clases que modelan los conceptos utilizados en la ontología de SCEMS.....	122
Figura 5.1: Cadena de suministro del caso de estudio.....	126
Figura 5.2: Agencia que modela el programa de abastecimiento del Proveedor.....	128
Figura 5.3: Perfil de inventario del Agente MKU-P11-P.....	129

Figura 5.4: Perfil de inventario del agente MKU-P12-P.....	130
Figura 5.5: Perfil de inventario del agente MKU-P2-P.....	131
Figura 5.6: Agencia que modela el programa de abastecimiento del Distribuidor .....	137
Figura 5.7: Agencia que modela el programa de abastecimiento del Minorista1 .....	139
Figura 5.8: Agencia que modela el programa de abastecimiento del Minorista2 .....	141
Figura 5.9: Extracto del archivo de configuración XML para la definición de la agencia .....	143
Figura 5.10: Planes de estados de los agentes involucrados en el escenario 1.....	144
Figura 5.11: Planes de estados de los agentes involucrados en el escenario 2.....	145
Figura 5.12: Planes de estados de los agentes involucrados en el escenario 3.....	148
Figura A.1: Ventana inicial de SCEMS .....	157
Figura A.2: Importación de proyectos.....	159
Figura A.3: Selección del proyecto “Produccion” .....	161
Figura A.4: <i>SCEMS Navigator</i> .....	161
Figura A.5: Visualización de reporte HTML .....	163
Figura A.6: Extracto del archivo general de configuración .....	165
Figura A.7: Extracto del archivo de configuración de un agente RKU.....	166
Figura A.8: Detalle en la especificación de una orden para el agente RKU .....	167
Figura A.9: Detalle de configuración de los agentes SP .....	168
Figura A.10: Detalle para la configuración del plan de actividades.....	169
Figura A.11: Estructura general del archivo de configuración de los eventos.....	170
Figura A.12: Estructura para la especificación de eventos .....	171



## INDICE DE TABLAS

Tabla 2-1: Programas de abastecimiento de las Empresas A y Empresa B .....	45
Tabla 2-2: Agenda de uso para el RKU .....	48
Tabla 2-3: Lista de entrada/salida para el MKU-m1g-silo.....	50
Tabla 2-4: Lista entrada/salida para el MKU-m1e-almacén .....	50
Tabla 2-5: Plan de actividades para el SP-1 .....	52
Tabla 4-1: Variaciones en cantidad y tiempo para una orden y sus utilidades.....	123
Tabla 5-1: Recurso y tiempo requerido por los procesos de transporte .....	127
Tabla 5-2: Programa de abastecimiento del Proveedor.....	128
Tabla 5-3: Lista de Entrada/Salida del Agente MKU-P11-P .....	129
Tabla 5-4: Lista de Entrada/Salida del agente MKU-P12-P .....	130
Tabla 5-5: Lista de Entrada/Salida del agente MKU-P2-P .....	130
Tabla 5-6: Agenda de uso del agente RKU-C1 .....	131
Tabla 5-7: Plan de actividad del SP-4-P.....	131
Tabla 5-8: Plan de actividad del SP-17-P.....	132
Tabla 5-9: Plan de actividad del SP-5-P.....	132
Tabla 5-10: Plan de actividad del SP-18-P.....	132
Tabla 5-11: Plan de actividad del SP-6-P.....	132
Tabla 5-12: Plan de actividad del SP-19-P.....	132
Tabla 5-13: Programa del agente IOA-Dist .....	132
Tabla 5-14: Programa de abastecimiento del Distribuidor.....	134
Tabla 5-15: Programa del agente IOA-Proveedor.....	135
Tabla 5-16: Programa del agente IOA-Minorista1.....	135
Tabla 5-17: Programa del agente IOA-Minorista2.....	136
Tabla 5-18: Lista de Entrada/Salida del agente MKU-P12-C .....	136
Tabla 5-19: Lista de Entrada/Salida del agente MKU-P2-C .....	136

Tabla 5-20: Lista de Entrada/Salida del agente MKU-P12-S .....	138
Tabla 5-21: Lista de Entrada/Salida del agente MKU-P2-S .....	138
Tabla 5-22: Programa de abastecimiento del Minorista1 .....	139
Tabla 5-23: Programa de abastecimiento del Minorista2.....	141

## RESUMEN

La ocurrencia de eventos disruptivos que afectan el normal cumplimiento de un plan o programa en ejecución es un problema bien conocido por la actividad de planificación. Debido a esto los planes y programas son elaborados con holguras para poder enfrentar los desvíos que se generan. Sin embargo, el problema surge debido a que los Sistemas de Ejecución, donde los eventos disruptivos son detectados, carecen de funcionalidades apropiadas que les permitan emplear las holguras para mitigar el efecto de dichos eventos.

En los últimos años se ha manifestado un creciente interés en cómo gestionar estos eventos disruptivos, con el fin de minimizar su impacto, surgiendo conceptos como monitoreo de eventos, gestión de riesgos y gestión de eventos en la cadena de suministro. Si bien se han realizado avances importantes las propuestas aún no satisfacen completamente los requerimientos planteados. En la literatura relacionada se describen propuestas de solución, la mayoría basada en el seguimiento de órdenes; donde se evidencia la ausencia de autonomía en la búsqueda e implementación de una solución frente a un desvío detectado en un programa en ejecución.

En esta tesis se describen los requerimientos para un sistema de gestión de eventos disruptivos en la cadena de suministro. Se identifica que las propuestas existentes no satisfacen completamente los requerimientos, y que en la mayoría de ellas se plantea como trabajo futuro la definición de un comportamiento autónomo para gestionar los eventos.

Con el fin de contribuir a solucionar la necesidad planteada, el objetivo de esta tesis es desarrollar una arquitectura para la gestión autónoma de eventos disruptivos en la cadena de suministro, donde las acciones de control se basan en la hipótesis de que el uso adecuado de las holguras definidas en los programas reducirá la brecha existente entre los Sistemas de Planificación y los Sistemas de Ejecución. Como resultados se ha obtenido: Una forma diferente de descomponer el problema buscando respetar las propiedades de autonomía de los componentes distribuidos; un modelo conceptual y un modelo multi-agente para la gestión de eventos disruptivos en la cadena de suministro; y un prototipo experimental que ha sido aplicado a un caso de estudio, mostrando que un uso adecuado de las holguras permite distribuir entre los participantes las implicaciones negativas de un evento disruptivo.

La propuesta de la tesis introduce una nueva forma de modelar el problema permitiendo la automatización del proceso de identificación de excepción, búsqueda de solución e implementación de la misma.

## ABSTRACT

The occurrence of disruptive events affecting the normal performance of a plan or schedule is a well known problem for the planning activity. For this reason plans and schedules are created including slacks to deal with deviations that are generated. However, the problem arises because in the Execution Systems, where disruptive events are detected, there is a lack of appropriate functionality for using the slacks to mitigate the impact of such events

In recent years there has been a growing interest in how to manage these disruptive events, in order to minimize its impact, emerging concepts such as event monitoring, risk management and event management in the supply chain. While significant progress has been made, proposals still do not fully satisfy the requirements. In the literature, most of the solutions that have been presented are based on orders monitoring, showing low levels of autonomy in the pursuit and implementation of a solution to a deviation detected in a schedule that is currently running.

This thesis describes the requirements for a supply chain event management system. It is identified that existing proposals do not completely satisfy the requirements, and that in most of them it is proposed as future work defining an autonomous behavior to disruptive events management.

With the aim of addressing the presented need, the objective of this thesis is to develop an architecture for the autonomous management of supply chain disruptive events, where the self-control actions are based on the assumption that an appropriate use of the slacks defined in the schedules will reduce the gap between Planning Systems and Execution Systems. The following results were obtained: a novel way of decomposing problem which preserves the autonomy of the distributed components; a conceptual model and a multi-agent model for supply chain disruptive event management; and an experimental prototype, applied to a case study, proofing the concept of using slacks to distribute the negative implications of a disruptive event among the participants.

The proposal of the thesis introduces a new way to model the problem by allowing the automation of the process of identifying an exception, search for a solution and implementing it.

# CAPÍTULO 1 – INTRODUCCIÓN

En este capítulo se presentan en primer lugar los conceptos que describen el dominio general en el cual hace foco esta tesis. Estos involucran: cadena de suministro, gestión de la cadena de suministro, eventos disruptivos, gestión de eventos disruptivos en la cadena de suministro (SCEM), Sistema SCEM y niveles de automatización de los Sistemas SCEM.

Se describe al Sistema SCEM como un sistema de control complejo, se presenta el problema a resolver y el estado del arte en relación al mismo a través de una discusión de trabajos relacionados. Finalmente se formulan los objetivos de la tesis y las hipótesis en la que se sustenta.

## 1.1. LA CADENA DE SUMINISTRO

En esta tesis, la expresión *cadena de suministro* refiere a *una red de entidades de negocio autónomas, colectivamente responsables por la obtención, producción y distribución de productos*. [Swaminathan et al, 1998; Simchi-Levi et al, 1999].

En los ambientes altamente competitivos y dinámicos en los que actualmente opera, la cadena de suministro genera fuertes obligaciones entre los participantes con el fin de lograr ventajas competitivas. Tal condición permite observar la necesidad de establecer vínculos más estrechos entre las entidades de negocio participantes que las obliga a mantener interacciones de manera sistemática implicando un cambio en las posiciones tradicionales. Los proveedores, mayoristas y minoristas se ven como “socios”, comparten información, delinean planes de negocios, ventas y promociones en forma conjunta, participan como un solo equipo de trabajo en la investigación y desarrollo de productos, analizan en forma conjunta la demanda, planifican y generan los programas de abastecimiento. Esta nueva forma de relacionarse se denomina *colaboración*.

La evolución de la cadena de suministro impulsó también la evolución de su gestión, surgiendo el concepto de *gestión de la cadena de suministro* como una nueva forma de conducir los negocios de manera integrada [Handfield and Nichols, 1999; Simchi-Levi et al, 1999].

## 1.2. LA GESTIÓN DE LA CADENA DE SUMINISTRO

En esta tesis la expresión *gestión de la cadena de suministro* refiere al propósito de *integrar de manera eficiente y eficaz el conjunto de entidades de negocio autónomas que la componen, tal que los productos sean producidos y distribuidos en las cantidades correctas, en los lugares correctos y en el tiempo correcto de modo de minimizar los costos totales cumpliendo con los requerimientos de nivel de servicio requerido por los clientes.*

Para ser competitiva la entidad de negocio implementa sistemas de información que le permite gestionar de manera integrada todas sus áreas de negocio. Los sistemas ERP (*Enterprise Resource Planning*) constituyen hoy una solución de software que trata las necesidades de la entidad de negocio desde la perspectiva de los procesos de negocio que la misma debe llevar a cabo para alcanzar los objetivos integrando todas sus funciones. La implementación de un sistema ERP permite identificar e implementar un conjunto de mejores prácticas, procedimientos y herramientas diseñadas para lograr la excelencia organizacional a través de la integración funcional [Mabert et al, 2002].

En lo que refiere a integración de las funciones internas de una entidad de negocio, las propuestas de solución han evolucionado brindando respuestas a los cambiantes requerimientos del mercado. Sin embargo, los actuales requerimientos plantean la necesidad de nuevos enfoques orientados al fortalecimiento de las relaciones de colaboración entre las entidades de negocio. En este nuevo escenario los sistemas de gestión de la cadena de suministro representan una propuesta integradora, focalizada en las relaciones inter-empresariales.

El propósito de los sistemas de gestión de la cadena de suministro es coordinar la programación y ejecución de las actividades que tiene lugar a lo largo de la misma. Esta tesis hace foco en la coordinación de los procesos de generación y ejecución de los programas de abastecimiento.

Un *programa de abastecimiento* se define como un conjunto de órdenes, donde cada orden representa un proceso de suministro (producción o distribución) que asigna materiales a un lugar, define los recursos necesarios, el periodo de tiempo durante el cual cada recurso es utilizado, su capacidad y el estado requerido. En una cadena de suministro gestionada en forma integrada puede haber varios programas de

abastecimiento sincronizados, generalmente uno por cada entidad de negocio o área de la misma. Por ejemplo, el programa de abastecimiento de los puntos de venta de una red de distribución, está sincronizado con el programa de producción de cada proveedor de los productos distribuidos, y cada uno de ellos a su vez está sincronizado con los programas de abastecimiento de los materiales utilizados para la producción.

Los sistemas de gestión de la cadena de suministro según su arquitectura pueden ser de gestión centralizada o de gestión descentralizada [Villarreal et al, 2003a, 2003b]. Los enfoques centralizados se caracterizan por un único sistema de información que concentra todas las funciones de gestión de la cadena de suministro. Estos sistemas tienen aceptación en cadenas de suministro donde la coordinación la realiza la entidad de negocio dominante y es la que implementa el sistema y concentra la información y el poder de decisión, o bien el servicio de gestión lo brinda una tercera parte. El mayor inconveniente de este enfoque es la pérdida de autonomía que significa para las entidades de negocio participantes, por lo cual es un enfoque poco aceptado. En contraposición, el enfoque descentralizado supone una arquitectura distribuida con una relación punto-a-punto que permite mantener la autonomía de las entidades de negocio. Esta arquitectura es la que permite implementar una relación de colaboración entre las entidades de negocio en el sentido definido en la sección previa.

Los sistemas de información que implementan las funcionalidades de los procesos de gestión de la cadena de suministro, al igual que los sistemas ERP, separan el software de programación del software de ejecución. Los *Sistemas de Programación* son los responsables de generar el conjunto de programas de abastecimiento sincronizados a lo largo de la cadena de suministro, mientras que los *Sistemas de Ejecución* son los responsables del control de la ejecución de dichos programas.

### 1.3. EVENTOS DISRUPTIVOS

Un *evento disruptivo* se define como un cambio significativo ocurrido durante la ejecución de un programa de abastecimiento que puede afectar el programa y su sincronización con los otros programas, y producir efectos negativos que se propagan a lo largo de la cadena de suministro [Kleindorfer and Saad, 2005; Lee et al, 1997; Radjou et al, 2002]. Dicho cambio puede tener lugar en los valores de especificación de una orden o en los valores programados de la disponibilidad de los recursos. Una

*excepción* se define como una desviación en un programa de abastecimiento que impide el cumplimiento de una o más órdenes.

La incertidumbre inherente en una cadena de suministro es reconocida por el paradigma de programación robusta [Landeghem and Vanmaele, 2002] el cual propone definir holguras (*buffers* de materiales, y de capacidad y tiempo de los recursos) para absorber cambios que puedan tener lugar durante la ejecución de un programa. Estas holguras permiten generar programas robustos, con más probabilidad de permanecer estables durante su ejecución. El objetivo es tratar de evitar que las entidades de negocio que participan del proceso de colaboración, tengan que acordar y sincronizar nuevamente sus programas de abastecimiento; tarea usualmente referida como *re-programación*, que es costosa y requiere tiempo.

Si bien los beneficios de tener un programa de abastecimiento robusto son indiscutibles, los usuarios reconocen que no es una tarea fácil asignar holguras a los recursos para lograr robustez y usarlas de manera sistemática para mantener la ejecución de un programa de abastecimiento acorde a los valores programados.

En principio, un programa de abastecimiento robusto tendría holguras para cubrir el efecto de eventos disruptivos. No obstante, debido a la imposibilidad de predecir con certeza el tiempo, el lugar de ocurrencia y la magnitud de tales eventos disruptivos, es posible que las holguras previstas no puedan absorber todas las variaciones.

Considérese el ejemplo de una empresa de producción en el cual hay definida una orden de producción que toma una cantidad  $q1$  de una materia prima desde un depósito en la fecha  $f1$ , la procesa y entrega una cantidad  $q2$  de un producto en el almacén en una fecha programada  $f2 > f1$ . Este programa está sincronizado con un programa de abastecimiento de una empresa de distribución, en el cual hay una orden de distribución que en la fecha  $f3 > f2$  toma una cantidad  $q3$  de producto del almacén para su distribución. Cuando un evento disruptivo tiene lugar, es necesario analizar la factibilidad del programa de abastecimiento en el punto directamente afectado por la variación ocurrida a efectos de verificar si la misma es absorbida por las holguras. En el ejemplo, supóngase que se redujo la cantidad de la materia prima disponible en el depósito por deterioro imprevisto (evento disruptivo). Si dicha variación no supera el stock de seguridad  $Ss1$  (holgura), la misma es absorbida por la holgura de la materia



prima con lo cual no afecta la orden de producción, de modo que el programa de abastecimiento se mantiene factible. Si la variación es mayor que el stock de seguridad  $Ss1$ , se produce una excepción, el programa ya no es factible y se analiza propagar las modificaciones resultantes tratando de que el efecto del evento disruptivo pueda ser absorbido con las restantes holguras del programa. Continuando con el ejemplo, al no poder cubrir toda la variación ocurrida con el stock de seguridad  $Ss1$ , la cantidad de materia prima disponible  $qm1$  es menor que la cantidad requerida por la orden de producción bajo análisis ( $qm1 < q1$ ), esto implica que la cantidad de producto producido  $qm2$  será menor que la cantidad especificada ( $qm2 < q2$ ). En este caso, si esta variación (reducción) en la cantidad producida del producto es menor que el stock de seguridad  $Ss2$  del mismo en el almacén, la misma es absorbida y el programa de abastecimiento se mantiene factible (la excepción fue resuelta). Si esta variación es mayor que el stock de seguridad  $Ss2$ , el programa ya no es factible puesto que no se podrá cumplir con la cantidad  $q3$  requerida por la orden de distribución del programa de abastecimiento del distribuidor. En este caso, en el marco del proceso de colaboración, se analiza resolver la excepción propagando aguas abajo las modificaciones resultantes tratando de que el efecto del evento disruptivo pueda ser absorbido con las holguras del programa de abastecimiento del distribuidor. Si la variación puede ser finalmente absorbida, ambos programas de abastecimiento se mantienen factibles ante el evento disruptivo. En caso contrario la excepción no fue resuelta y un proceso de re-programación es necesario.

Debido a la compleja relación entre recursos y órdenes que conforman un conjunto de programas de abastecimiento sincronizados, es difícil utilizar de manera apropiada las holguras, por lo cual luego que la disrupción ocurre los usuarios llevan a cabo un proceso “ad-hoc” cuya decisión usualmente es re-programar. Para explotar adecuadamente las holguras de los programas de abastecimiento, dichos procesos de decisión requieren de metodologías que permitan resolver los problemas de manera sistemática. Los programas de abastecimiento deben ser reparados a través de modificaciones enmarcadas dentro de las holguras y “localizadas” a efectos de reducir la propagación de las variaciones y evitar el efecto “nerviosismo” [Villarreal et al, 2003a; Villarreal et al, 2003b]. Para este objetivo deben utilizarse modelos apropiados para verificar factibilidad y reparar programas de abastecimiento teniendo en cuenta la naturaleza distribuida de la cadena de suministro. Esta tarea requiere disponer de información de los eventos disruptivos ya sea a través de la captura en el momento en

que ocurren o mediante la anticipación de la ocurrencia de los mismos, lo cual puede ayudar a tomar mejores decisiones. Para satisfacer este requerimiento, es necesario desarrollar una tarea denominada monitoreo de eventos disruptivos, la cual deberá ejecutarse en modo reactivo y/o en modo predictivo.

Bajo este escenario, surgió como concepto la Gestión de Eventos en la Cadena de Suministro [Knickle and Kemmeler, 2002; Masing, 2003].

## 1.4. GESTIÓN DE EVENTOS EN LA CADENA DE SUMINISTRO

Un evento en la cadena de suministro se puede definir como un cambio de estado [Masing, 2003], siendo posible identificar dos tipos de eventos: los programados y los disruptivos. Para la gestión de eventos en la cadena de suministro, los eventos disruptivos constituyen el objetivo del proceso de monitoreo, ya que ellos son imprevistos y requieren ajustar el o los programas de abastecimiento en ejecución afectados o pueden conducir a una re-programación. Ejemplos de eventos disruptivos que pueden presentarse en los distintos puntos de las cadenas de suministro son los siguientes [Masing, 2003; Knickle and Kemmeler, 2002]: un proveedor adelanta, atrasa, reduce la cantidad o anula una entrega; un cliente solicita adelantar, atrasar, reducir o aumentar la cantidad, anular una orden o incluir una nueva orden de compra; el vencimiento o la rotura de materiales/productos; un recurso asignado al proceso de suministro (producción o distribución) pueden reducir su disponibilidad o quedar fuera de servicio durante un periodo de tiempo.

El objetivo de la gestión de eventos es permitir que la cadena de suministro pueda responder a eventos disruptivos minimizando el impacto de los mismos, tratando de evitar la re-programación. La gestión de eventos en la cadena de suministro se puede definir como *el proceso de negocio en el cual eventos disruptivos son reconocidos a tiempo, acciones reactivas son rápidamente ejecutadas, el flujo, tanto de materiales como de información, es ajustado basándose en reglas de negocios y la notificación a empleados claves es inmediata* [Knickle and Kemmeler, 2002].

La implementación de estos procesos de negocio ha dado origen a un nuevo tipo de sistemas de información denominados Sistemas de Gestión de Eventos de la Cadena

de Suministro. Estos sistemas son conocidos por sus siglas en inglés como Sistemas SCEM (Supply Chain Event Management) [Masing, 2003; Zimmermann, 2006].

## 1.5. SISTEMAS SCEM

Los sistemas SCEM tienen por propósito la localización y el seguimiento del flujo de eventos a lo largo de la cadena de suministro, monitoreando eventos programados y eventos disruptivos, detectando cambios relevantes y notificándolos a los usuarios en tiempo real para que tomen las decisiones que consideren pertinentes. Brindan soporte de simulación para que el usuario pueda analizar alternativas y encontrar soluciones a eventos disruptivos y soporte para analizar y documentar el efecto para subsecuentes procesos. Proveen también soporte para evaluar indicadores, y para analizar y evaluar datos históricos [Knickle and Kemmeler, 2002].

Idealmente un sistema de gestión de eventos en la cadena de suministro debería ser pro-activo y responder a las interrupciones que tienen lugar en un programa de abastecimiento generando e implementando soluciones en forma automática [Alvarenga and Schoenthaler and Alvarenga, 2003; Christopher, 2005]. *Generar una solución* implica modificar adecuadamente el/los programa/s de abastecimiento sincronizados haciendo uso de las holguras para absorber la interrupción. *Implementar la solución* generada implica enviar el/los programa/s modificado/s al/los correspondiente/s Sistema/s de Ejecución.

Debido a la naturaleza autónoma de las entidades de negocio que forman parte de una cadena de suministro, un sistema de gestión de eventos en la cadena de suministro debería ser concebido como un sistema de información distribuido que permita la gestión de las interrupciones que se producen en la cadena de suministro, desempeñando las funciones de monitoreo, de generación e implementación de soluciones y de reporte al personal competente a través de un proceso de colaboración ejecutado de manera descentralizada por cada una de las entidades de negocio participantes.

---

### 1.5.1. NIVELES DE AUTOMATIZACIÓN

Un sistema que implementa un proceso de gestión de eventos en la cadena de suministro puede presentar diferentes niveles de automatización [Bearzotti et al, 2011].

El primer nivel comprende a los *sistemas de monitoreo*, concebidos como una extensión de los sistemas de seguimiento y localización, identificados por sus siglas del inglés como T&T (Tracking and Tracing) [Szirbik et al, 2000; Kärkkäinen et al, 2003]. El segundo nivel corresponde a los *sistemas de alarma* que no solo detectan un cambio sino que analizan si el mismo produce una disrupción, notificando a quien corresponda el suceso [Hoffmann et al, 1999; Teuteberg and Schreber, 2005; Zimmermann, 2006]. El tercer nivel corresponde a *sistemas soporte de decisiones* que no solo notifican sino que proponen cursos de acción [Speyerer and Zeller, 2004; Adhitya et al, 2007; Cauvin et al, 2009]. Por último, el cuarto nivel corresponde a *sistemas correctivos autónomos* que no solo detectan y analizan eventos disruptivos sino que intentan generar e implementar las soluciones pertinentes de manera autónoma.

Un *sistema de monitoreo* de eventos representa el nivel más bajo de automatización. Para estos sistemas la automatización se focaliza en la captura de los eventos que se producen durante la ejecución de un programa sin analizar el impacto que estos eventos tienen ni su relevancia. Todo el proceso relacionado con la interpretación de los eventos y la toma de acciones en caso de que se produzca un desvío queda a criterio de los usuarios que observan el sistema de monitoreo.

Un *sistema de alarmas* automatiza el proceso de monitoreo, capturando y analizando el impacto de los eventos que afectan los programas de abastecimiento en ejecución, detectando eventos disruptivos y notificando los mismos a los usuarios. Las actividades restantes quedan en manos de los usuarios que son los responsables de tomar las decisiones.

Un *sistema soporte de decisión* además de capturar y analizar el impacto de los eventos, analiza y propone al usuario diferentes escenarios que pueden ser factibles para solucionar los desvíos en los programas de abastecimiento en ejecución producidos por eventos disruptivos detectados.

Un *sistema correctivo autónomo* debe tener capacidad de decisión, pudiendo definir soluciones e implementarlas actuando autónomamente. En el caso de no encontrar solución factible el sistema debe notificar al Sistema de Programación para que proceda a re-programar el abastecimiento. Para generar una solución el sistema hará uso de las holguras del programa de abastecimiento. Si los programas de abastecimiento en ejecución fueron generados sin holguras, al no contar con márgenes de acción este

sistema se comportará como un sistema de alarma. Un programa sin holguras no puede adaptarse a los cambios y requerirá frecuentes re-programaciones.

### 1.5.2. SCEM COMO UN SISTEMA DE CONTROL COMPLEJO

La Teoría de Control [Monostori et al, 1998; Zhou and Venkatesh, 1999] brinda un marco conceptual en el cual es posible detectar conceptos claves necesarios a la hora de buscar una solución al problema de gestión de eventos en la cadena de suministro. La gestión de eventos disruptivos puede ser vista como un problema de control, que extiende el campo de las aplicaciones tradicionales de control. Como problema de control es posible identificar los siguientes tipos de variables:

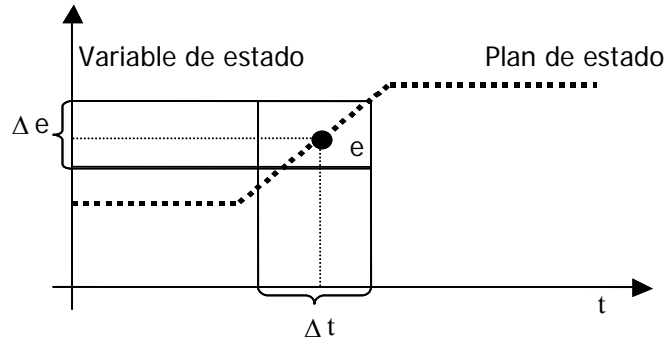
*Variable Observada:* durante la ejecución del programa esta variable es observada con el propósito de detectar la ocurrencia de un evento disruptivo, el cual es generado por un cambio significativo en el valor de la misma.

*Variable Controlada o Variable de Estado:* define el punto de control. Tiene asociado un programa de estados definido con ciertas holguras por el Sistema de Programación. Un *perfil de estado* es una sucesión de valores que la variable de estado puede tomar en su horizonte programado; es representado como una lista de tuplas de la forma  $[e, t]$  donde  $e$  es el valor programado que la variable de estado tiene que tomar en el tiempo  $t$ . Una interrupción es una diferencia significativa entre el valor actual de esta variable (observado o calculado) y el programado para el tiempo actual.

*Variable de decisión:* es la variable independiente cuyo valor puede ser ajustado para mitigar los efectos de una interrupción con el propósito de mantener la factibilidad del programa de abastecimiento en ejecución. Cambiar el valor de esta variable afecta tanto a la variable observada como a la variable controlada.

En la Figura 1.1 se representa la *variable de estado* como una función del tiempo. En la misma se ha resaltado el punto  $[e, t]$  el cual representa la  $n$ -ésima tupla  $[e, t]$  del *perfil de estado* de dicha variable. Representa el valor programado  $e$  que la *variable de estado* tiene que tomar en el tiempo  $t$ .  $\Delta e$  representa la holgura de  $e$  y es la variación permitida por el programa para el valor de la variable de estado y  $\Delta t$  representa la holgura de  $t$ , y es la variación permitida por el programa para el tiempo.  $\Delta e$

$x \Delta t$  define el conjunto de valores que la variable de estado puede tomar sin que su programa pierda factibilidad.



**Figura 1.1: Representación gráfica de la variable de estado**

Una vez definido el problema desde la perspectiva de la Teoría de Control es posible analizarlo desde la perspectiva de la Teoría de los Sistemas Complejos. Un sistema caracterizado como complejo implica el estudio de sus componentes individuales y cómo el sistema cumple con sus objetivos (globales) mediante las interacciones de sus componentes y el comportamiento emergente de esas interacciones. Un sistema complejo es definido como un sistema con un gran número de elementos o bloques de construcción, capaces de intercambiar estímulos unos con otros y con su ambiente donde el comportamiento global es más que la suma de los comportamientos individuales. Con esta característica un sistema complejo puede responder a las condiciones cambiantes externas [Ottino, 2003].

La gestión de eventos de una cadena de suministro constituye un problema de control complejo debido a que una aplicación práctica puede involucrar cientos o quizás miles de variables de estado, cuya interrelación es frecuentemente implícita desde una perspectiva global. Por lo tanto un sistema SCEM es un sistema de control complejo en el que intervienen entidades de negocio distintas con objetivos e intereses individuales no siempre en armonía. Para encontrar una solución a una disrupción producida por un evento, los distintos elementos que componen el sistema deben interactuar entre sí con el objetivo de minimizar el efecto de la misma. Esta búsqueda de solución puede ser interpretada como un comportamiento emergente del sistema ya que no es propio de un elemento en particular, sino que surge de su interacción. Al encontrar solución a las

situaciones cambiantes tanto internas como externas, el sistema de gestión de eventos está presentando adaptabilidad ya que cambia para responder al cambio.

## 1.6. PROBLEMA A RESOLVER

Una cadena de suministro deber ser capaz de responder sistemáticamente a un evento disruptivo minimizando su impacto, evitando la necesidad de re-programación. Este es un problema de control complejo que requiere monitorear cambios en los programas de abastecimiento en ejecución, analizarlos y detectar la ocurrencia de una excepción; generar e implementar de manera automática una solución utilizando las holguras del programa de abastecimiento de la entidad de negocio como primer alternativa; y en caso que la excepción persista, usando las holguras de los diferentes programas sincronizados de las entidades de negocio que participan del proceso de colaboración. Finalmente, en caso que la excepción no sea resuelta, requiere notificar al Sistema de Programación correspondiente para que proceda a re-programar.

## 1.7. TRABAJOS RELACIONADOS: ESTADO DEL ARTE

El problema de sistemáticamente identificar y corregir interrupciones que tienen lugar durante la ejecución de programas de abastecimiento en la cadena de suministro ha sido estudiado tanto en el ámbito industrial como en el académico. Las propuestas más relevantes se discuten a continuación:

ECTL-Monitor, es un sistema basado en agentes de software diseñado para ser embebido en el portal de Internet de la empresa con el propósito de proveer a los clientes con datos necesarios para que los mismos puedan realizar el seguimiento de sus órdenes y suscribirse a un servicio de notificaciones [Hoffmann et al, 1999]. Estas son características propias de un sistema de alarmas. No provee mecanismo alguno que le permita realizar en forma autónoma acciones correctivas de un programa de abastecimiento en ejecución cuando un evento disruptivo ocurre. No es concebido como un sistema de información a ser ejecutado en colaboración por diferentes entidades de negocio de la cadena de suministro.

PROVE, es un sistema prototipo basado en la tecnología de agentes de software móviles focalizado en proveer soporte al proceso de negociación entre empresas necesario para construir y mantener una empresa virtual. Un evento disruptivo tiene

lugar cuando una empresa decide abandonar la empresa virtual. En este caso la propuesta provee soporte para encontrar una nueva empresa que pueda asumir el compromiso [Szirbik et al, 2000]. Dicho soporte se basa en un proceso de decisión semi-estructurado que no puede ser completamente automatizado por lo cual la intervención humana es necesaria. Estas características son propias de un sistema soporte de decisiones. PROVE también provee soporte para monitoreo y seguimiento de órdenes, funciones típicas de un sistema de monitoreo. Si bien fue diseñado como un sistema inter-organizacional, no posee mecanismos para realizar de manera autónoma acciones correctivas cuando ocurre un evento disruptivo diferente al abandono de la empresa virtual por parte de una empresa.

DIALOG, es un sistema basado en agentes de software que comparte datos a efectos de facilitar el seguimiento de una orden, ofreciendo información acerca de su estado [Kärkkäinen et al, 2003]. Esta funcionalidad es característica de un sistema de monitoreo. Fue diseñado como un sistema inter-organizacional pero sin habilidad para realizar acciones de corrección de manera autónoma en colaboración con las empresas participantes de la cadena de suministro.

FORWIN, es un sistema prototipo basado en la web que provee soporte para monitoreo y alarmas a través de la habilidad para detectar desviaciones, mediante un análisis causal a partir de síntomas y proponer potenciales medidas correctivas. Los síntomas surgen de problemas de performance obtenidos a partir del valor de métricas predefinidas [Speyerer and Zeller, 2004]. Sus funcionalidades son características de un sistema soporte de decisiones y fue concebido como un sistema inter-organizacional. No es provisto con habilidades para realizar acciones correctivas de manera autónoma a nivel de órdenes y recursos asociados a un programa de abastecimiento a nivel operativo en ejecución. Está focalizado en un proceso de gestión de alto nivel de abstracción de la cadena de suministro, dejando de lado el monitoreo y control del proceso de ejecución de las órdenes.

CoS.MA, es un sistema de información par-a-par diseñado para integrar datos de cada una de las empresas de la cadena de suministro que integran el proceso de colaboración, de modo que todos tengan una vista de los datos pertinentes. El seguimiento y localización de las órdenes en la cadena de suministro es soportado por tecnologías móviles y Auto-ID [Teuteberg and Schreber, 2005]. Sus funcionalidades



son características de un sistema de alarmas. Fue diseñado como un sistema inter-organizacional, pero no realiza acciones correctivas autónomas para mitigar efectos de eventos disruptivos.

PAMAS, es un sistema SCEM basado en agentes de software que monitorea los órdenes en la medida que las mismas se mueven en la cadena de suministro detectando cuando un evento disruptivo afecta una de ellas. Utiliza perfiles de órdenes para identificar las órdenes a ser monitoreadas porque algunas de sus características las hace más vulnerables [Bodendorf and Zimmermann, 2005; and Zimmermann, 2006]. Sus funcionalidades son características de un sistema de alarmas. Este sistema desarrollado también como un sistema inter-organizacional no presenta capacidades para realizar acciones de control correctivas de manera autónoma.

Una propuesta para minimizar el impacto de eventos disruptivos sobre un sistema de información intra-organizacional es presentada [Cauvin et al, 2009]. La misma se basa en un análisis de eventos disruptivos, la caracterización del proceso de recuperación de errores y un método de reparación cooperativo de sistemas industriales distribuidos. El objetivo de la propuesta es asistir al usuario en el diseño de procesos de recuperación, proponiéndoles soluciones para que tome la decisión final. Estas funcionalidades son características de un sistema soporte de decisiones. El sistema si bien genera soluciones posibles no tiene capacidad de decisión, y no fue diseñado para operar en un ambiente inter-organizacional.

Un sistema de información intra-organizacional basado en el uso de tecnologías de agentes de software fue propuesto para generar programas de producción de procesos de manufactura en forma dinámica y responder de manera flexible a eventos disruptivos generados por cambios en la demanda del mercado [Guo and Zhang, 2009]. Sus funcionalidades son características de un sistema soporte de decisiones. No es provisto con mecanismos para realizar acciones correctivas de manera autónoma para mitigar el efecto de eventos disruptivos, y su arquitectura no es adaptable para trabajar en un contexto de colaboración inter-organizacional.

En resumen, los sistemas de información de seguimiento y localización de órdenes (*Tracking and Tracing*) son el status quo en la mayoría de las empresas (ámbito industrial). Las propuestas académicas analizadas previamente son una evolución de estos sistemas de información, pero la siguiente etapa es un sistema SCEM capaz de

generar e implementar soluciones de manera autónoma cuando un evento disruptivo ocurre durante la ejecución de un programa de abastecimiento definido a nivel operativo. Estos sistemas SCQM deberían estar basados en métodos proactivos y sistemáticos de predicción y reacción a situaciones que son muy diferentes de los típicos reportes de excepción generados por los sistemas ERP. El trabajo presentado en esta tesis tiene por propósito contribuir en esta dirección.

## 1.8. OBJETIVO E HIPÓTESIS DE LA TESIS

Se define como objetivo general de esta tesis desarrollar una propuesta de solución para el problema de la gestión de eventos disruptivos en la cadena de suministro, donde la propuesta debe estar caracterizada por su autonomía en la búsqueda e implementación de acciones de control cuando una excepción se produce, siendo éste el aspecto principal que marca la diferencia con otras propuestas relacionadas.

El desarrollo de la propuesta de solución se fundamenta en las siguientes hipótesis:

- El problema de responder sistemáticamente a un evento disruptivo minimizando su impacto, evitando la necesidad de re-programación puede ser solucionado implementando un adecuado sistema de gestión de eventos de la cadena de suministro.
- El desarrollo de un modelo conceptual sustentado en la Teoría de Control y de la Teoría de los Sistemas Complejos permitirá abordar la complejidad del problema.
- Un mecanismo descentralizado para el monitoreo de eventos disruptivos y detección de excepciones permitirá satisfacer el requerimiento de respetar la naturaleza descentralizada y distribuida del problema.
- El desarrollo de un mecanismo de colaboración para la determinación de acciones correctivas frente a una excepción, permitirá absorber el efecto de la misma utilizando las holguras distribuidas a largo de la cadena de suministro.

- Al contar con un sistema para la gestión de eventos disruptivos, una cadena de suministro podrá detectar los eventos en etapas tempranas mejorando los resultados de las acciones para absorber los mismos.

Se propone como meta comprender los posibles mecanismos de colaboración y desarrollar tecnologías de información clave para hacer posible:

- La coordinación descentralizada de los procesos de monitoreo y captura de eventos disruptivos.
- La coordinación descentralizada de los procesos de control que permiten definir las acciones correctivas.

El resultado de esta tesis es un modelo para la gestión descentralizada de eventos disruptivos, en base al cual se desarrolla un sistema informático que lo implementa. El sistema resultante automatiza el proceso de búsqueda e implementación de una solución y en caso no encontrarla reporta al Sistema de Programación la excepción detectada.

La propuesta de esta tesis introduce dos aspectos novedosos no tratados en los trabajos previos. En primer lugar, el sistema se concibe como un sistema de información distribuido que permite la colaboración entre las entidades de negocio para dar respuesta a la necesidad de la gestión de eventos disruptivos, teniendo en cuenta la naturaleza distribuida de la cadena de suministro, preservando la autonomía de cada una de ellas [Cauvin et al, 2009]. En segundo lugar, se proveen mecanismos para que el sistema pueda llevar a cabo acciones autónomas de control correctivo. Este aspecto está incluido en respuesta al requerimiento de automatización de la búsqueda e implementación de soluciones, no cumplido por las propuestas existentes, las que se centran principalmente en hacer frente a las tareas de monitoreo, captura y notificación de eventos disruptivos. La capacidad de realizar en forma automática las acciones de control correctivas ha sido identificado como un área apenas explorada [Zimmermann, 2006; Pereira, 2009]. En este enfoque, las acciones de control utilizan las holguras previstas en los programas de abastecimiento en la búsqueda de una solución para resolver una excepción. Otro aporte propuesto en este trabajo es el protocolo de red denominado *Double Contract Net Protocol*. El mismo consiste en un protocolo de red que un agente de software con el rol de coordinador comienza a ejecutar con un agente de software con el rol de mediador, el

cual a su vez comienza a ejecutar tantos protocolos de red como agentes se requieran para dar respuesta.

Los resultados de esta tesis dieron origen a las siguientes publicaciones:

**An autonomous multi-agent approach to supply chain event management.**

Bearzotti, Lorena A., Enrique Salomone, Omar J. Chiotti,, International Journal of Production Economics, Available online 12 September 2011, ISSN 0925-5273, 10.1016/j.ijpe.2011.08.023.(<http://www.sciencedirect.com/science/article/pii/S092552731100377X>).

**Supply Chain Event Management System.**

Bearzotti Lorena, Fernandez Erica, Guarnaschelli Armando, Salomone Enrique and Chiotti Omar (2011). In Supply Chain Management - Applications and Simulations, Mamun Habib (Ed.), ISBN: 978-953-307-250-0, InTech, Available from: <http://www.intechopen.com/articles/show/title/supply-chain-event-management-system>

**Collaborative Negotiations in Supply Chain Event Management Multi-Agent**

**System.** Bearzotti, L., Salomone, E., Chiotti, O. Actas de las Jornadas Chilenas de Computación 2008, Workshop en Agentes y Sistemas Colaborativos, Noviembre 2008, Punta Arenas, Chile.

**An Autonomous Multi-Agent Approach to Supply Chain Event Management.**

Lorena Bearzotti, Enrique Salomone, Omar Chiotti. 2008 IEEE International Conference on Service Operations and Logistics, and Informatics, October 2008. Beijing, China.

**Modelando un Sistema de Gestión de Eventos Basado en Agentes desde la**

**Perspectiva Holónica.** Lorena Bearzotti, Enrique Salomone, Omar Chiotti. Actas del XII Encuentro Chileno de Computación ECC 2004, Noviembre 2004, Arica, Chile

**Coordinación en un Sistema de Gestión de Eventos autónomo basado en agentes de**

**software.** Lorena Bearzotti, Enrique Salomone, Omar Chiotti. Actas del IX Congreso Argentino de Ciencias de la Computación, Octubre 2003, La Plata, Argentina (aceptado).

**Autonomous Event Management Systems based on Software Agents.**

Lorena Bearzotti, Enrique Salomone, Omar Chiotti. Proceedings XXIX Conferencia Latinoamericana de Informática CLEI 2003, Septiembre 2003, La Paz, Bolivia

Además se han presentado los reportes técnicos que se detallan:

**An Agent-based approach to Supply Chain Event Management: closing the gap**

**between the planning and the execution.** Lorena Bearzotti, Enrique Salomone, Omar Chiotti.

(aceptado en Doctoral Consortium in 8<sup>th</sup> International Conference on Enterprise Information Systems in Paphos, Cyprus, 2006).

**A Multi-Agent Approach to Supply Event Management.** Lorena Bearzotti, Enrique Salomone, Omar Chiotti. (aceptado en Doctoral Consortium in 12<sup>th</sup> IFAC Symposium on Information Control Problems in Manufacturing in Saint Etienne, France, 2006).

## CAPÍTULO 2 – UN MODELO CONCEPTUAL DE UN SISTEMA PARA LA GESTIÓN DE EVENTOS DISRUPTIVOS EN LA CADENA DE SUMINISTRO

En este capítulo se presentan los modelos planteados desde una perspectiva conceptual que permiten capturar las características de la propuesta de solución para el problema de la gestión de eventos disruptivos en la cadena de suministro.

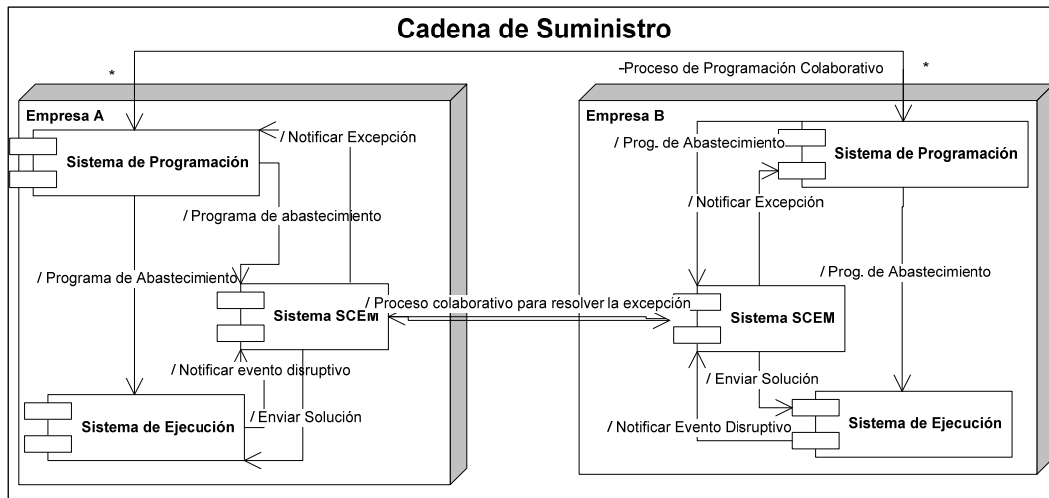
En una primera instancia se presenta un modelo de componentes principales de un sistema SCEM, buscando respetar la autonomía y heterogeneidad de las entidades de negocio que la integran. En base a la idea de sistema distribuido se propone un modelo funcional del proceso de gestión de eventos disruptivos, el cual es la base para el desarrollo de un modelo conceptual. Se identifican las fuentes posibles de eventos, los principales componentes y cómo éstos interaccionan para lograr el comportamiento global esperado de un sistema SCEM.

### 2.1. MODELO DE COMPONENTES PRINCIPALES

Esta tesis presenta una propuesta para automatizar el proceso de absorber el efecto de eventos disruptivos que afectan un programa de abastecimiento en ejecución utilizando las holguras del conjunto de programas de abastecimiento de una cadena de suministro que operan de manera sincronizada. Las entidades de negocio que integran la cadena de suministro llevan a cabo un proceso de decisión en colaboración para “distribuir” la variación en sus programas de abastecimiento. Esta propuesta resulta en un sistema SCEM distribuido que trabaja de manera integrada con los Sistemas de Programación y los Sistemas de Ejecución de cada empresa.

La Figura 2.1 presenta una representación gráfica de los componentes principales de la arquitectura del sistema SCEM, para el caso particular de la interacción entre dos empresas de la cadena de suministro. El sistema propuesto está basado en una arquitectura distribuida donde cada empresa es considerada como un sistema abierto interactuando uno con otro. De esta manera se define la arquitectura como una *red de*

sistemas SCEM, donde cada uno representa una entidad de negocio autónoma (empresa) de la cadena de suministro.

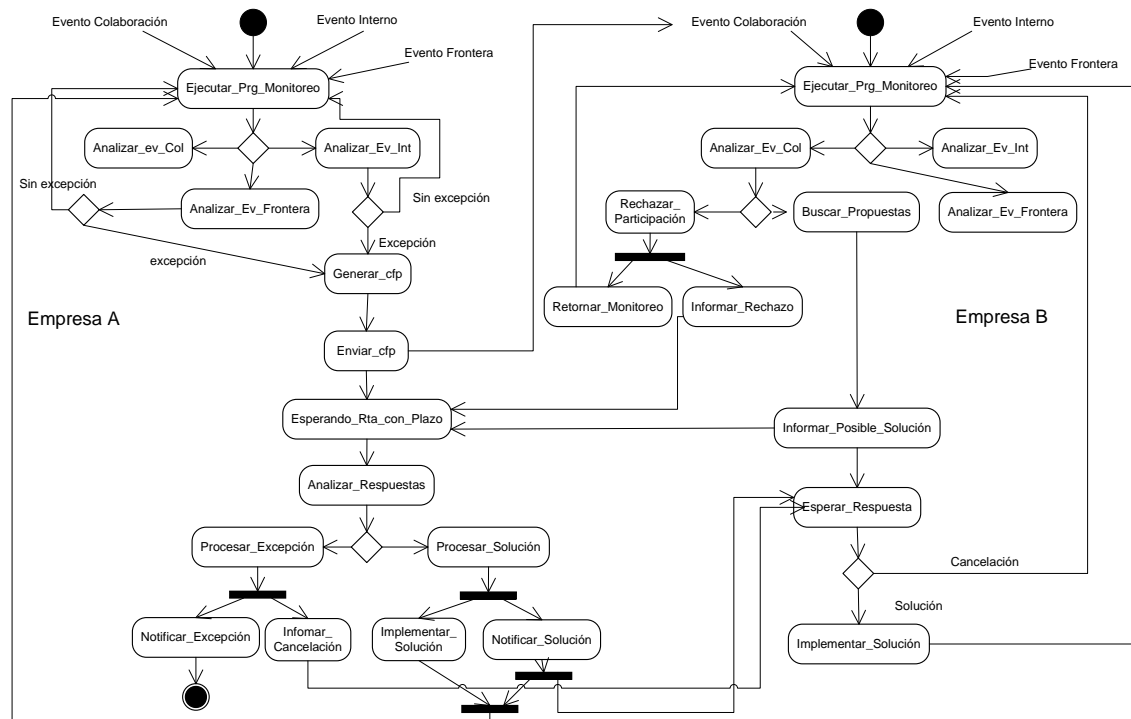


**Figura 2.1: Componentes principales de un sistema SCEM**

Las empresas que integran la cadena de suministro a través de un proceso de programación inter-empresa ejecutado en colaboración acuerdan un plan global de suministro. En base a este plan, el *Sistema de Programación* de cada empresa (por ejemplo la *Empresa A* en la Figura 2.1) genera un *programa de abastecimiento* sincronizado con el correspondiente programa de abastecimiento de la otra empresa y comunica dicho programa al *Sistema de Ejecución* y al *Sistema SCEM*. Durante la ejecución del programa de abastecimiento, el Sistema SCEM de cada empresa monitorea los cambios provenientes del Sistema de Ejecución. Cuando una *excepción* es detectada por el Sistema SCEM de alguna empresa, analiza las holguras del programa de abastecimiento afectado para definir acciones de control que permitan resolver la excepción. Si no puede generar una *solución interna*, esto es, que las variaciones sean absorbidas por las holguras del programa de abastecimiento en ejecución, dicho Sistema SCEM puede iniciar un *Proceso de colaboración inter-organizacional para resolver la excepción*. Para ello se comunica con el Sistema SCEM de otra empresa ejecutando un proceso de búsqueda de solución en colaboración. Si una *solución* es encontrada, el Sistema SCEM de cada empresa la envía al respectivo Sistema de Ejecución a efectos de que la misma sea implementada. Si no se encuentra solución, el Sistema SCEM que detectó la *excepción* la notifica al Sistema de Programación para re-programación.

## 2.2. MODELO FUNCIONAL DEL SISTEMA SCYM

En el modelo de componentes principales presentado en la sección anterior el Sistema SCYM es definido como una red de Sistemas SCYM individuales, cada uno asociado con una empresa de la cadena de suministro que participa del proceso de colaboración. Para especificar este modelo se definieron las funciones que un Sistema SCYM debe proveer para satisfacer el objetivo propuesto, desarrollándose un modelo funcional. El mismo resulta de integrar las funciones de un proceso genérico para la gestión de eventos disruptivos detallando los roles y las relaciones entre las entidades de negocio de una cadena de suministro. Continuando con el ejemplo de la sección previa, la Figura 2.2 presenta el modelo funcional para el caso particular de las dos empresas (*Empresa A* y *Empresa B*).



**Figura 2.2: Modelo Funcional para el proceso SCYM integrado (Empresas A y B)**

Una vez recibido el programa de abastecimiento, enviado por el Sistema de Programación, el Sistema SCYM de una entidad de negocio (por ejemplo *Empresa A* en la Figura 2.2) realiza la primera actividad del proceso (*Ejecutar\_Prg\_Monitoreo*) la cual consiste en sensor el entorno para recibir eventos disruptivos de interés que se puedan



producir. Se han identificado tres tipos de eventos disruptivos: *internos*, producidos dentro de la empresa por lo que provienen del Sistema de Ejecución; *de colaboración*, asociados con solicitudes de colaboración de empresas que participan del proceso de colaboración; y *de frontera*, cuyo origen se encuentra en clientes o proveedores que no participan del proceso de colaboración.

Cuando un evento disruptivo *interno* es capturado (por el Sistema SCEM de la *Empresa A* en la Figura 2.2), el mismo es analizado (*Analizar\_Ev\_Interno*) para determinar si la excepción generada puede ser resuelta por las holguras del programa de abastecimiento en ejecución (Sección 1.3, Capítulo 1). Si puede ser resuelta, genera una *solución interna*, la notifica al Sistema de Ejecución para su implementación y retorna a la tarea de monitoreo. Por otro lado, si no es posible generar una solución interna (el programa resulta no factible), ante la necesidad de resolver la excepción, recurre al proceso de búsqueda de una solución en colaboración con una o más empresas de la cadena de suministro. En una primera instancia realiza un llamado a participar del proceso a las empresas cuyos programas de abastecimiento están sincronizados con el programa afectado para distribuir el efecto de la disrupción (*Generar\_cfp*). Los mensajes generados son enviados (*Enviar\_cfp*) y pasa a un estado de espera de respuestas (*Esperar\_Rta\_con\_Plazo*). En la Figura 2.2 la *Empresa A* invita a participar de un proceso de colaboración a la *Empresa B*. Cuando todas las respuestas han sido recibidas o el plazo límite ha sido alcanzado procede al análisis de las respuestas (*Analizar\_Respuestas*). Esta actividad busca determinar si la combinación de las respuestas recibidas permite o no resolver la excepción detectada.

En el caso de no resolver la excepción se pasa a procesar la misma (*Procesar\_Excepción*), se notifica (*Notificar\_Excepción*) al Sistema de Programación y se informa a todas las empresas que han aportado una posible solución que la solicitud ha sido cancelada (*Informar\_Cancelación*). El proceso de gestión de eventos termina porque una tarea de re-programación debe ser llevada a cabo.

En el caso de haber encontrado una solución, la misma es procesada (*Procesar\_Solución*), informada al Sistema de Ejecución para que la implemente (*Implementar\_Solución*) y se notifica de la solución generada a todas las empresas involucradas (*Notificar\_Solución*). Luego se retorna al estado de monitoreo (*Ejecutar\_Prg\_Monitoreo*).

Cuando el Sistema SCEM recibe un evento disruptivo *de colaboración* proveniente de una empresa que participa del proceso de colaboración (en la Figura 2.2 la *Empresa B* recibe el evento por el llamado a participación de la *Empresa A*) se activa la tarea (*Analizar\_Ev\_Col*) donde se simula cómo ese evento afecta el programa de abastecimiento. Si la disrupción asociada a la excepción puede ser absorbida con las holguras (el programa sigue siendo factible), la colaboración es factible y se realiza una búsqueda de propuestas (*Buscar\_Propuestas*) que posteriormente son informadas al emisor de la solicitud (*Informar\_Posible\_Solución*) y se pasa al estado de esperar respuesta (*Esperar\_Respuesta*). La respuesta que se puede recibir es que se cancela la participación por lo que retorna al monitoreo, o bien se recibe la solución escogida, lo que implica actualizar el correspondiente programa de abastecimiento para reflejar esos cambios (*Implementar\_Solución*). En caso de encontrar que la colaboración no es factible (las holguras del programa de abastecimiento afectado no pueden absorber el cambio propuesto) procede a rechazar la participación (*Rechazar\_Participación*), informa el rechazo (*Informar\_Rechazo*) a la empresa que envió el llamado a participar y retorna al estado de monitoreo (*Ejecutar\_Prg\_Monitoreo*).

El tercer caso ocurre cuando un evento disruptivo *de frontera* es recibido (por el Sistema SCEM de la *Empresa A* en la Figura 2.2). El mismo es analizado (*Analizar\_Ev\_Frontera*) mediante un proceso análogo al caso de recibir un evento disruptivo interno.

El modelo funcional propuesto cubre todas las entidades de negocio de la cadena de suministro que participan de la colaboración. El comportamiento global del sistema SCEM surge de las interacciones de los sistemas SCEM individuales implementados por cada empresa.

## 2.3. MODELO CONCEPTUAL DEL SISTEMA SCEM

La gestión de eventos disruptivos en la cadena de suministro ha sido definida como un problema de control complejo cuyo objetivo es llevar a cabo acciones correctivas individuales o en colaboración entre las unidades de negocio cuando se presenta un desvío en un programa de abastecimiento en ejecución (Capítulo 1, Sección 1.5.2).

El objetivo de esta sección es desarrollar un modelo conceptual para una propuesta de solución al problema de la gestión de eventos disruptivos en la cadena de suministro determinando sus principales características y atributos.

---

### 2.3.1. DEFINICIÓN DEL MODELO CONCEPTUAL

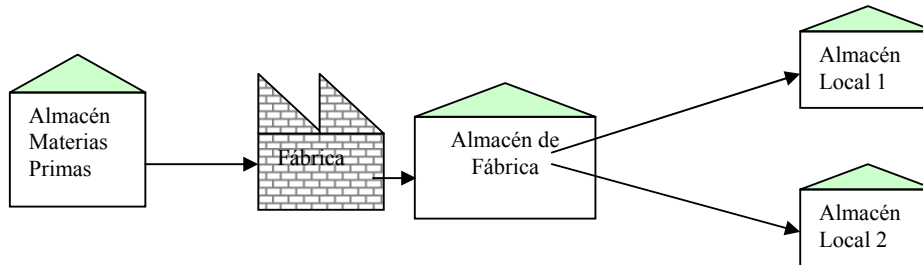
En el Capítulo 1, Sección 1.2, un *programa de abastecimiento* se ha definido como un conjunto de órdenes, donde cada orden representa un proceso de suministro (producción o distribución) que asigna materiales a un lugar, define los recursos necesarios, el periodo de tiempo durante el cual cada recurso es utilizado, su capacidad y el estado requerido.

La ejecución de un programa de abastecimiento así definido implica tomar cada orden que lo compone y realizar las operaciones definidas por el proceso de suministro que la orden representa en el período de tiempo especificado. Durante la ejecución, un evento disruptivo puede afectar los valores especificados de la orden. Los cambios observados sobre una orden son producidos por cambios en los materiales o en los recursos necesarios para su ejecución. Por lo cual, en esta tesis se propone definir un *punto de control* para cada recurso o material relevante involucrado en un programa de abastecimiento. El término relevante indica que el punto de control se definirá sobre aquellos materiales y recursos que se consideren críticos para la ejecución del programa de abastecimiento, en el sentido que sus holguras son limitadas y la variación de los mismos puede producir una excepción.

Un *punto de control* constituye un sub-sistema de control en el cual se identifican las correspondientes variables de control (decisión), de estado (controlada) y monitoreada (observada) definidas en el Capítulo 1, Sección 1.5.2. Estos sub-sistemas de control están conectados entre sí a través de los procesos de suministro. De este modo, se define *el modelo conceptual del Sistema SCCEM como una red de puntos de control definidos sobre los recursos o materiales conectados entre sí mediante procesos de suministro*.

Considérese el ejemplo de la Figura 2.3 en la que se muestra una cadena de suministro constituida por dos empresas, la Empresa A propietaria del Almacén de Materias Primas (AMP), la Fábrica y el Almacén de Fábrica (AF), y la Empresa B

responsable de la distribución de los productos desde al Almacén de Fábrica a los Almacenes Locales 1 y 2 (AL1 y AL2) de su propiedad.



**Figura 2.3: Cadena de suministro. Empresas A (producción) Empresa B (distribución y venta)**

En la Tabla 2.1 se muestra la Orden\_Producción\_01 del programa de producción en ejecución de la Empresa A y las Orden\_Distribución\_01 y Orden\_Distribución\_02 del programa de distribución en ejecución de la Empresa B. Cada orden especifica el punto de almacenamiento, representados por la sigla ISL (*Inventory Stock Location*), en el cual el material es ingresado.

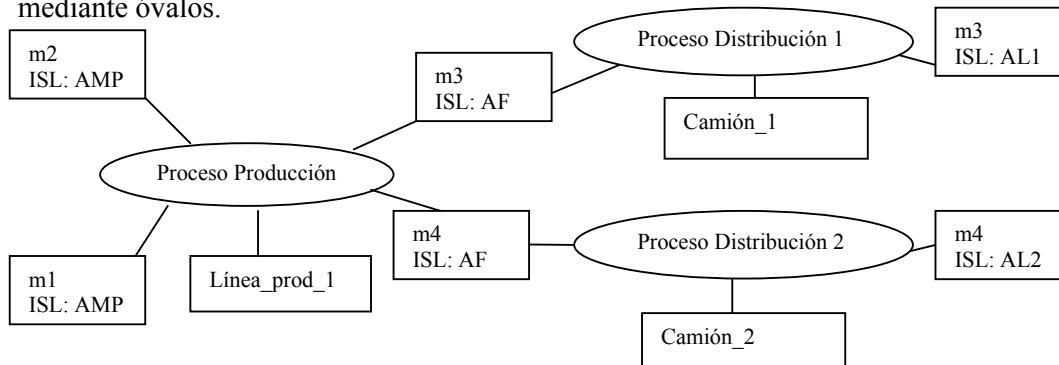
Empresa A		Empresa B			
Programa de producción		Programa de distribución			
Orden Producción 01		Orden Distribución 01		Orden Distribución 02	
Material	m3	material	m3	material	m4
Cantidad	q3	cantidad	q4	cantidad	q5
fecha Inicio	t1	fecha Inicio	t2	fecha Inicio	t3
Duración	d1	duración	d2	duración	d3
Recurso	Línea prod 1	recurso	Camión 1	recurso	Camión 2
capacidad requerida	c1	capacidad requerida	c2	capacidad requerida	c3
		estado inicial	ei2=AF	estado inicial	ei3=AF
		estado final	ef2=AL1	estado final	ef3=AL2
ISL	AF	ISL	AL1	ISL	AL2

**Tabla 2-1: Programas de abastecimiento de las Empresas A y Empresa B**

La lista de materiales, conocida usualmente como BOM (*Bill of Materials*) correspondiente al proceso de producción es:  $2.m1+3.m2=1.m3+2.m4$  esto es, con dos unidades del material m1 y tres unidades del material m2 se producen una unidad del material m3 y dos unidades del material m4.

Según la definición dada, el modelo conceptual del Sistema SCQM para este ejemplo quedará definido por la red de puntos de control de la Figura 2.4. En dicha

figura los puntos de control se representan mediante rectángulos y los procesos de suministro representados por las órdenes, que actúan de conectores, se representan mediante óvalos.



**Figura 2.4: Red de puntos de control del ejemplo**

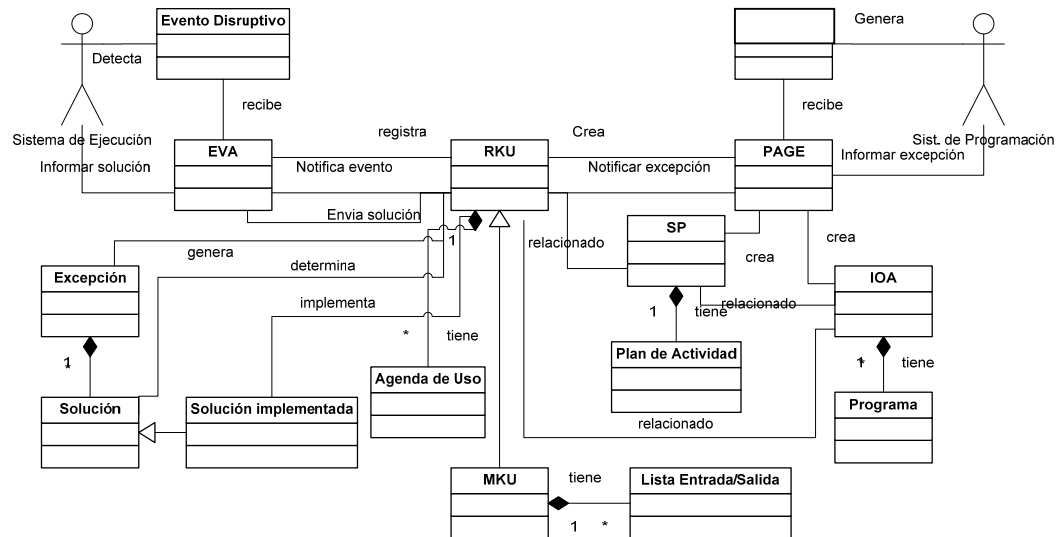
### 2.3.2. LOS ELEMENTOS DEL MODELO CONCEPTUAL

Los puntos de control definidos sobre recursos reciben el nombre de *RKU* (*Resource Keeping Unit*). De los recursos se diferencia al recurso material, el cual es modelado como un punto de control denominado *MKU* (*Material Keeping Unit*). Por otro lado los procesos de suministro que vinculan los distintos puntos de control se denominan *SP* (*Supply Process*) y son definidos a partir de las órdenes de producción o de distribución de los respectivos programas de abastecimiento. En la Figura 2.5 se presenta el modelo conceptual.

La función de un sub-sistema *RKU* es la de gestionar los eventos disruptivos que tienen lugar en el recurso que representa y que puedan alterar la ejecución del programa de abastecimiento al cual está asignado dicho recurso. Para llevar a cabo esta función, los atributos que definen un *RKU* determinan la disponibilidad del recurso que el mismo representa.

Un *RKU* monitorea la disponibilidad del recurso que representa; para ello tiene un programa de requerimientos del recurso que recibe el nombre de *agenda de uso*, en ella se detallan las órdenes que hacen uso del recurso. Esta agenda puede ser representada por la 5-tupla [id\_orden, SP, tiempo-inicio, duración, capacidad/cantidad requerida], donde id\_orden identifica la orden, SP señala cuál es el proceso de suministro relacionado, tiempo de inicio cuándo comienza el procesamiento de esa

orden, duración señala por cuánto tiempo será procesada y cantidad/capacidad cuánto del recurso será empleado.



**Figura 2.5: Definición del Modelo Conceptual**

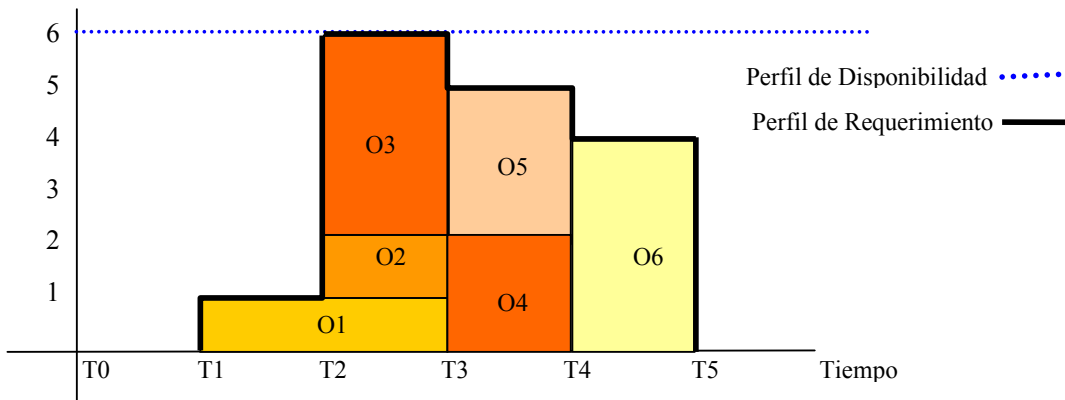
La capacidad disponible de un RKU define su *perfil de disponibilidad*, el cual puede variar en distintos períodos a lo largo del horizonte de programación. En base a la agenda de uso un RKU puede generar su *perfil de requerimiento* que indica el valor programado de uso de la capacidad del recurso en cada período de tiempo. La diferencia entre ambos perfiles define el *perfil de factibilidad*, que representa la holgura del recurso en los diferentes períodos a lo largo del horizonte de tiempo. Cuando la demanda es mayor a la disponibilidad se está en presencia de una excepción ya que al menos un requerimiento no puede ser satisfecho completamente.

En la Figura 2.6 se representa el perfil de disponibilidad y el perfil de requerimiento de un RKU que representa un centro de envasado cuya agenda de uso se detalla en la Tabla 2-2. El perfil de disponibilidad programado es de 6 unidades de envasado iguales para todos los períodos. Como puede observarse en dicha figura, durante el período de tiempo T0-T1 no habrá requerimientos por lo que la holgura del recurso será de 6 unidades; durante el período de tiempo T1-T2 el requerimiento estará dado por la orden 1 y es de 1 unidad, de modo que en ese período la holgura del recurso será de 5 unidades; durante el período de tiempo T2-T3, la capacidad requerida, resultante de atender en paralelo las órdenes 1, 2 y 3 alcanzará su capacidad disponible,

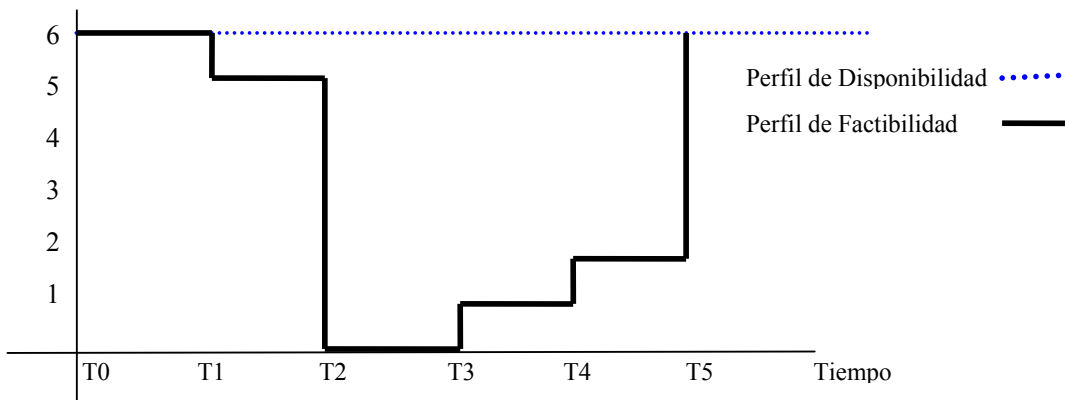
por lo cual no tendrá holgura. En la Figura 2.7 se representa el perfil de factibilidad que define las holguras del recurso en cada período de tiempo. Dado que el perfil de factibilidad no presenta segmentos negativos, el programa para este recurso es factible ya que puede cumplir con todas las órdenes de su agenda de uso.

id-Orden	SP-id	T-Inicio	Duración	Capacidad Requerida
1	SP-1	T1	2	1
2	SP-2	T2	1	1
3	SP-1	T2	1	4
4	SP-3	T3	1	2
5	SP-2	T4	1	3
6	SP-3	T4	1	4

**Tabla 2-2: Agenda de uso para el RKU**



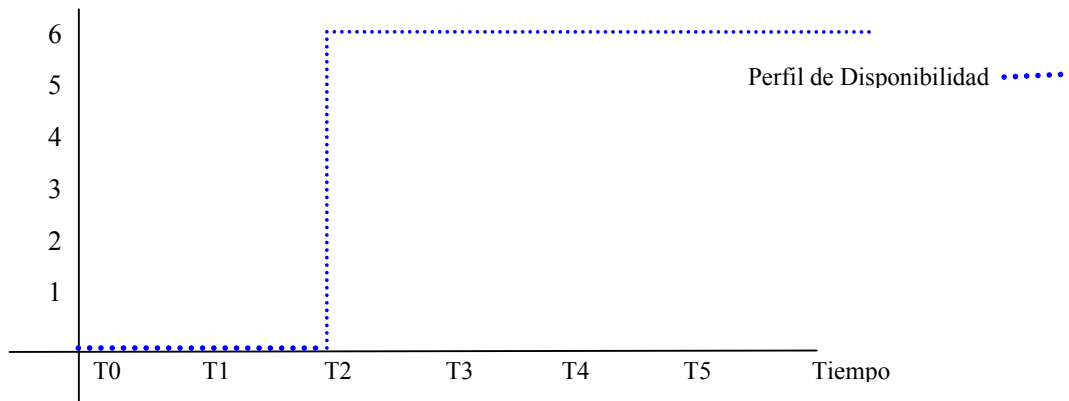
**Figura 2.6: Perfil de Requerimientos y Perfil de Disponibilidad para el RKU**



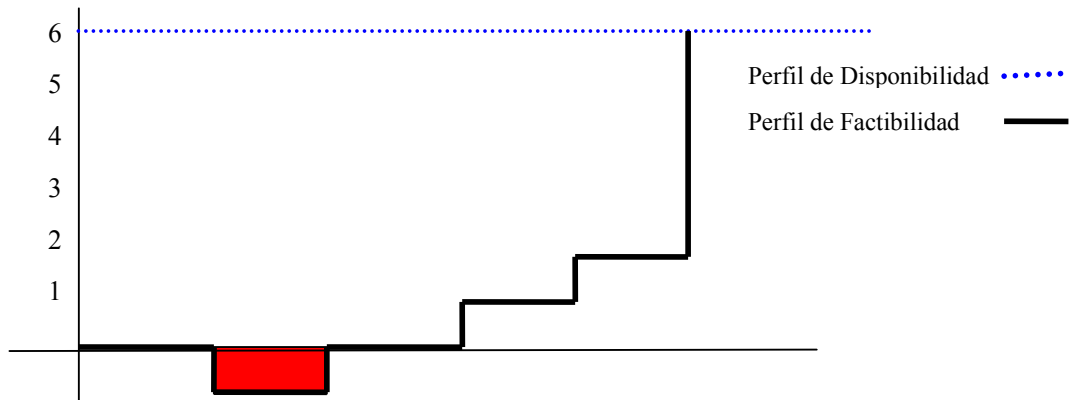
**Figura 2.7: Perfil de Factibilidad para el RKU**

Supóngase que por algún problema el recurso pasa a estar indisponible durante el período T0-T2. Ante este evento disruptivo el RKU que representa al recurso

actualiza su perfil de disponibilidad, el cual se representa en la Figura 2.8. Luego redefine su perfil de factibilidad, el que se muestra en la Figura 2.9. Analizando el perfil de factibilidad actualizado se observa que el mismo se vuelve no factible durante el período T1-T2. Esto indica que existe al menos una orden cuyo requerimiento el recurso no puede satisfacer completamente. En el ejemplo la orden 1 solo puede ser cumplida parcialmente. Esto indica que se produjo una excepción.



**Figura 2.8: Perfil de Disponibilidad actualizado.**



**Figura 2.9: Perfil de Factibilidad actualizado**

Un MKU monitorea la disponibilidad del material que representa en el ISL; para ello tiene un programa de requerimientos del material que recibe el nombre de *lista entrada/salida*, en ella se detallan todas las órdenes de entrada y salida del material al ISL. En base a esta lista un MKU puede generar su *perfil de inventario* que indica la función de su estado de disponibilidad.

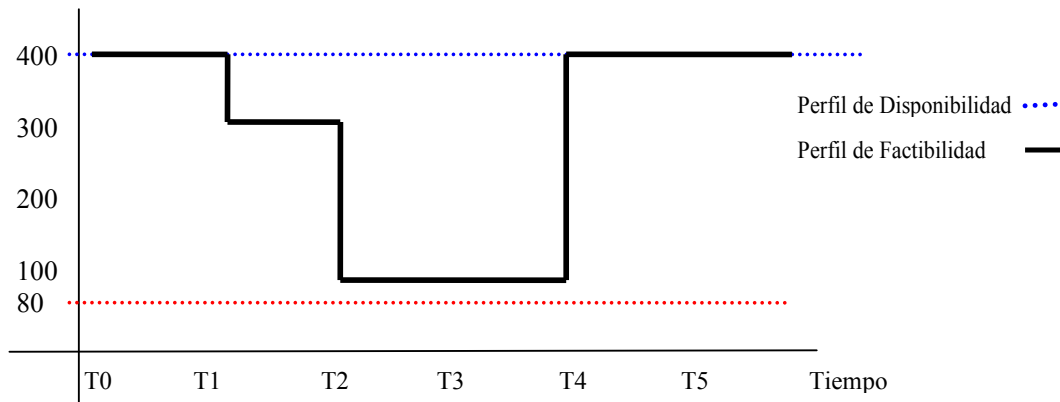


En la Figura 2.10 se representa el perfil de inventarios del MKU-m1g-silo que representa el material m1 almacenado a granel en un silo, cuya lista entrada/salida se detalla en la Tabla 2-3.

id-Orden	SP-id	T-Inicio	Material	Cantidad
1	SP-1	T1	m1	-100
2	SP-1	T2	m1	-200
3	SP-5	T4	m1	+300

**Tabla 2-3: Lista de entrada/salida para el MKU-m1g-silo**

Como puede observarse en la Figura 2.10, el inventario inicial es de 400 unidades que corresponden a su valor máximo. En el tiempo T1 la orden de salida 1 para atender el proceso de suministro SP-1 producirá una disminución de 100 unidades y en T2 la orden de salida 2 reducirá el inventario en 200 unidades, quedando en el silo 100 unidades, 20 unidades por arriba del stock de seguridad que es de 80 unidades. En el tiempo T4 ingresarán 300 unidades, llevando nuevamente el inventario a su nivel máximo.



**Figura 2.10: Perfil de inventario del MKU-m1g-silo**

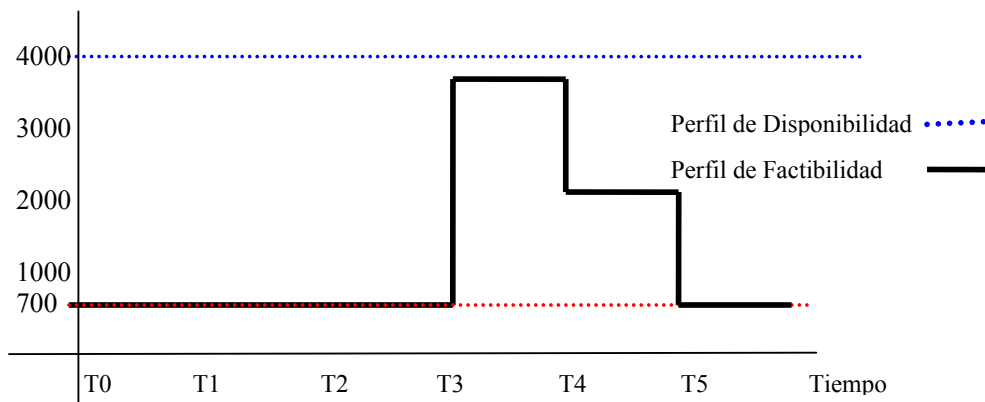
Id-Orden	SP-ID	T-Inicio	Material	Cantidad
1	SP-1	T3	m1e	+1000
2	SP-1	T3	m1e	+2000
3	SP-4	T4	m1e	-1500
4	SP-4	T5	m1e	-1500

**Tabla 2-4: Lista entrada/salida para el MKU-m1e-almacén**

En la Figura 2.11 se representa el perfil de inventarios del MKU-m1e1-almacén que representa el material m1 envasado en el envase e almacenado en el almacén de

fábrica, cuya lista entrada/salida se detalla en la Tabla 2-4. Como puede observarse, 1 unidad del material m1 a granel produce 10 unidades envasadas de dicho material, mientras que una unidad de envasado (envasadora) produce 500 unidades de m1e por período de tiempo  $T_i-T_{i+1}$ .

Como puede observarse en la Figura 2.11, el inventario inicial en el almacén de fábrica del material m1e es de 700 unidades, que corresponde a su stock de seguridad. En el tiempo  $T_3$  las órdenes de entrada 1 y 2 programadas para ser producidas por el proceso de suministro SP-1 incrementarán el inventario a 3700 unidades. En los tiempos  $T_4$  y  $T_5$  las órdenes de salida 3 y 4 reducirán nuevamente el inventario a su valor mínimo de 700 unidades.



**Figura 2.11: Perfil de inventario del MKU-m1e-almacén**

Un MKU posee tres atributos básicos, material (m), empaque (p) y lugar (l), representados por la 3-tupla  $[m, p, l]$ , los cuales permiten su identificación unívoca. Debido a que un SP representa una transición entre MKUs, tres tipos básicos de transiciones pueden definirse a partir de los atributos básicos del MKU: cambio de material ( $\Delta m$ ) como resultado de una operación de transformación física o química; cambio de empaque ( $\Delta p$ ) como resultado de una operación de empaque; y cambio de lugar ( $\Delta l$ ) como resultado de una operación de transporte. Estas transiciones básicas pueden ser combinadas para obtener siete tipos de transformaciones: cambio de material =  $\Delta m$ ; cambio de empaque =  $\Delta p$ ; cambio de lugar =  $\Delta l$ ; cambio de material y lugar =  $\Delta m \cup \Delta l$ ; cambio de material y empaque =  $\Delta m \cup \Delta p$ ; cambio de empaque y lugar =  $\Delta p \cup \Delta l$ ; cambio de material, empaque y lugar =  $\Delta m \cup \Delta p \cup \Delta l$ . Estos siete tipos de

transformaciones representan todas las transformaciones que un SP pueden producir sobre un MKU involucrado en una orden.

Es necesario destacar que los puntos de control RKU o MKU no se relacionan directamente entre ellos sino a través de los respectivos procesos de suministro (SPs), los cuales ellos conocen por medio de su agenda de uso o lista entrada/salida respectivamente. En el ejemplo previo, el RKU a través de su agenda de uso (Tabla 2.1) identifica al SP-1 como el proceso de suministro asociado a la orden 1 afectada por el evento disruptivo.

Un SP no es un punto de control sino que representa un balance de materiales definiendo cómo un conjunto de MKUs se relacionan y el conjunto de RKUs requeridos. Para ello, un SP cuenta un *plan de actividad* (Figura 2.5) el cual se puede representar por la 4-tupla [id-orden, fecha\_inicio, duración, L\_recurso] donde L\_recurso es representado por la 5-tupla [id-recurso, fecha\_inicio, duración, cantidad, modo] donde modo puede tomar tres valores posibles indicando si el recurso es consumido, producido o usado.

Para el ejemplo, SP-1 produce una transición  $\Delta p \cup \Delta l$ , ya que realiza un cambio de empaque (el material m1 pasa de granel a un envase) y de lugar (m1 pasa del silo al almacén de fábrica). El plan de actividades del SP-1 es definido en la Tabla 2-5. En dicha tabla una duración 0 indica que es instantánea.

id-Orden	T-Inicio	Duración	Lista-id
1	T1	2	L-1
2	T2	1	L-2

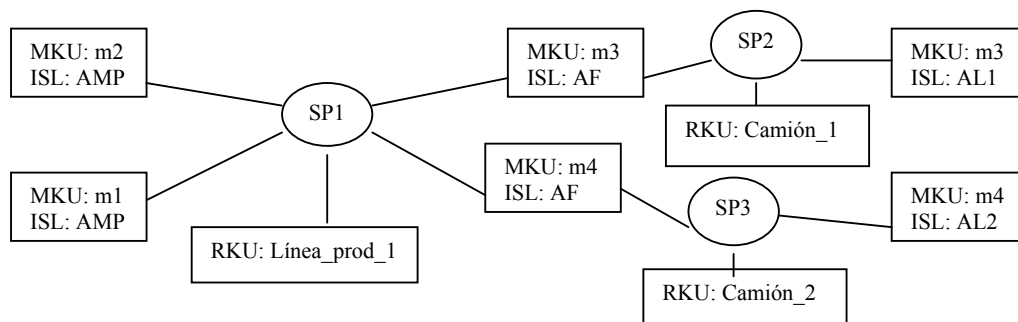
Lista-id	Recurso	T-Inicio	Duración	Cantidad	Modo
L-1	m1 g-silo	T1	0	100	Consumo
	m1 e-almacén	T3	0	1000	producción
	CEnvasado	T1	2	1	Uso
L-2	m1 g-silo	T2	0	200	Consumo
	m1 e-almacén	T3	0	2000	producción
	CEnvasado	T2	1	4	Uso

**Tabla 2-5: Plan de actividades para el SP-1**

De este modo, *el modelo conceptual del Sistema SCEM se define como una red de MKUs conectados entre si mediante SPs que usan RKUs para ejecutar sus tareas de transformación o transferencia.*

Para el ejemplo de la sección previa, el modelo conceptual quedaría definido por la red de puntos de control representada en la Figura 2.12. En dicha figura se han definido seis puntos de control sobre materiales, cada uno representado por un MKU en el respectivo punto de almacenamiento ISL, tres puntos de control definidos sobre los recursos, cada uno representado por un RKU y tres procesos de suministro, el proceso de producción SP1 que la Orden\_Producción\_01 representa, y los procesos de distribución SP2 y SP3 que las Orden\_Distribución\_01 y Orden\_Distribución\_02 representan.

Se puede observar en la figura que el SP2 se relaciona con MKU:m3 en ISL:AF, con el MKU:m3 en ISL:AL1 y con el RKU:Camión\_1. El SP2 puede identificar estas relaciones recurriendo a su lista de recursos.



**Figura 2.12: Red de MKUs, RKUs y SP del ejemplo**

En la próxima sección se detalla la interacción entre los elementos conceptuales identificados, con el fin de describir el comportamiento colectivo en la búsqueda de una solución cuando ocurre un evento disruptivo.

### 2.3.3. LA INTERACCIÓN EN EL MODELO CONCEPTUAL

La interacción entre los elementos del modelo conceptual se da en dos niveles: Un interacción entre los componentes principales del modelo, y una interacción entre los elementos que definen la red del modelo. Esta segunda interacción está relacionada con el proceso de colaboración invocado y coordinado por un punto de control de la red cuando detecta una excepción.

### 2.3.3.1 LA INTERACCIÓN ENTRE COMPONENTES

En el modelo de componentes principales (Figura 2.1) es posible identificar dos tipos de interacciones entre componentes: intra-organizacional e inter-organizacional.

La interacción intra-organizacional se produce cuando el Sistema SCEM de una empresa se vincula con el Sistema de Programación y con el Sistema de Ejecución. Se propone definir entidades dedicadas para llevar a cabo esta función; así se especifican la *Interfaz de Programación* (PAGE) y la *Interfaz de Ejecución* (EVA) (Figura 2.5).

La interfaz PAGE recibirá del Sistema de Programación el programa de abastecimiento a monitorear y comunicará al mismo aquellas excepciones para las cuales no ha podido encontrar una solución. El programa de abastecimiento contiene toda la información necesaria para crear la red de puntos de control y procesos de suministro correspondiente (MKUs, RKUs y SPs).

La interfaz EVA es responsable de recibir los *eventos internos* informados por el Sistema de Ejecución y los *eventos de frontera*, y notificarlos al RKU o MKU correspondiente. Cuando una solución es generada, ésta es notificada a EVA para que la informe al Sistema de Ejecución con la finalidad de actualizar el programa de abastecimiento en ejecución (implementación de la solución).

La interacción inter-organizacional se presenta cuando la empresa recibe de otra empresa de la cadena de suministro una solicitud de búsqueda de solución en colaboración para resolver una excepción. Para este proceso se define una *Interfaz Inter-organizacional* (IOA) dedicada a llevar a cabo esta función. La interfaz IOA recibe el *evento de colaboración* de parte del Sistema SCEM de la empresa solicitante y a partir de ese momento es responsable de todas las interacciones requeridas por el proceso de colaboración inter-organizacional, según el modelo funcional descrito en la Sección 2.2. Para la ejecución de esta tarea IOA tiene un *programa* de órdenes de suministro acordado con la otra empresa.

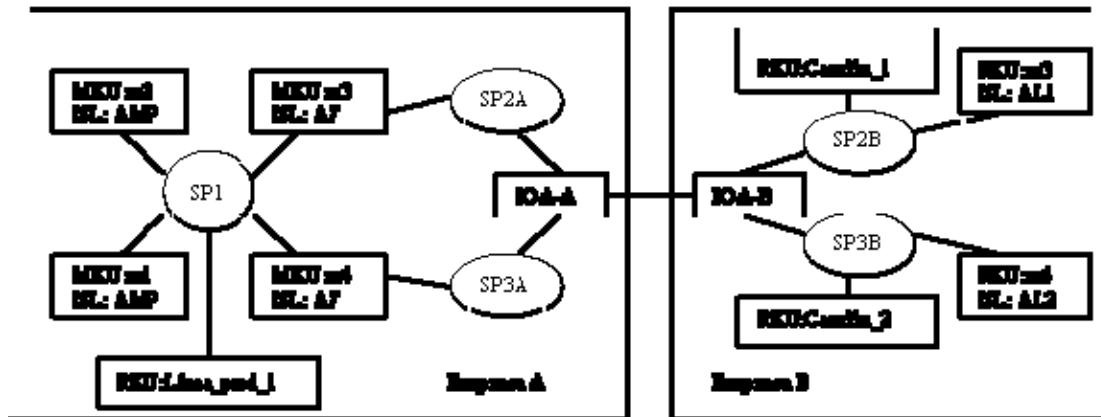


Figura 2.13: Red con IOA del ejemplo

Con la definición de IOA la red de puntos de control para el ejemplo de la sección previa queda representada como se muestra en la Figura 2.13. El programa de órdenes de IOA-A contiene las órdenes de despacho (procesos de despacho SP2A y SP3A) que la empresa A acordó con la empresa B, las cuales se corresponden con las órdenes de distribución Orden\_Distribución\_01 y Orden\_Distribución\_02 (procesos de distribución SP2B y SP3B) contenidas en el programa de órdenes de IOA-B.

### 2.3.3.2 LA INTERACCIÓN ENTRE ELEMENTOS DE LA RED

Cada RKU se define como un sub-sistema de control donde: la variable de estado es el *estado de su disponibilidad*; la variable monitoreada es su perfil de requerimiento; y las variables de control son el *tiempo* en su agenda de uso que puede ser usado para extender, adelantar y/o retrasar el uso del recurso, y la *disponibilidad* del recurso, la que puede ser utilizada dentro de los límites definidos por la holgura del programa. La función de un RKU es la gestión de los eventos disruptivos que podrían alterar la ejecución del programa de uso del recurso que representa. Por esa razón, debe controlar la disponibilidad de este recurso. Para llevar a cabo esta tarea, cada RKU se registra en EVA, la cual notifica los cambios significativos recibidos.

Los eventos disruptivos que pueden afectar al recurso que un RKU representa se clasifican en eventos disruptivos *directos* e *indirectos*. Un evento disruptivo *directo* es detectado por un cambio en la disponibilidad del recurso. Cuando se produce un cambio, el RKU genera el nuevo perfil de requerimiento del recurso y lo analiza para detectar el efecto de la variación. En este caso, el RKU trata de encontrar una solución mediante el uso de sus variables de control (tiempo y disponibilidad del recurso). Si

encuentra una solución, la cual no afecta las órdenes asociadas al recurso, decide implementarla. Dicha solución es enviada al Sistema de Ejecución a través de EVA para su puesta en ejecución. Cuando el evento disruptivo afecta al menos a una de las órdenes asociadas al recurso, se produce una excepción. El RKU debe buscar una solución recurriendo a un comportamiento colectivo. Si no se encuentra una solución, el RKU es responsable de notificar la excepción producida al Sistema de Programación a través de PAGE.

Un evento disruptivo *indirecto* para un RKU tiene su origen en un SP que requiere su uso. Este evento disruptivo es una consecuencia de un evento disruptivo que afectó a alguno otro recurso asociado al SP y que el RKU que representa a dicho recurso no puede resolver con las holguras de su agenda de uso. En otras palabras, es una disrupción causada por el comportamiento colectivo en la búsqueda de una solución.

Dado que es una especialización del RKU, un MKU también puede recibir dos tipos de eventos disruptivos: directos, relacionados con un cambio de su inventario, por ejemplo daño o vencimiento del material y actualización del nivel de inventario; indirectos, relacionados a cambios en el tiempo y/o en la cantidad de una orden de su lista de entrada/salida. Ambos tipos de cambios modifican su perfil de inventario, que es la función de su estado de disponibilidad. La variable monitoreada es el inventario y sus variables controladas son el *tiempo* y *cantidad* de cada orden de la lista de entrada/salida. Cuando la modificación no es absorbida por la holgura del material una excepción se produce. La misma es detectada analizando el perfil de inventario modificado. Una vez detectada la excepción el MKU procede según se ha descrito para el RKU.

Un tipo especial de eventos disruptivos indirectos son los *de frontera*. Estos afectan únicamente a los MKUs que determinan los límites de la cadena de suministro. Cuando un MKU recibe este evento, debe comportarse como si éste fuera un evento disruptivo directo, según se describió previamente. Un evento disruptivo de frontera afecta una orden de la lista de entrada/salida del MKU que lo relaciona con una empresa que no forma parte de la cadena de suministro, por lo que no se establece un vínculo de colaboración. Eso significa que no es posible negociar de manera sistemática los valores de las especificaciones de las órdenes de suministro que los vincula.

Un SP solo recibe eventos disruptivos indirectos provenientes de un punto de control, RKU o su especialización MKU, que llama a participar de un proceso de colaboración para la búsqueda de una solución. Ante el evento, la función del SP es identificar empleando su plan de actividades, cuáles son los puntos de control (RKU y MKU) afectados para convocarlos a participar del proceso. A su vez recibe las respuestas de éstos, las unifica y genera una respuesta al punto de control que generó la convocatoria. Los detalles del comportamiento colectivo son descriptos en la siguiente sección.

---

### 2.3.3.3 EL PROCESO DE COORDINACIÓN

La interacción entre RKUs, MKUs, SPs y eventualmente IOA cuando se necesita colaborar con otros miembros de la cadena, siempre se produce dentro de un marco de coordinación bien definido. Los responsables de llevar a cabo tareas de coordinación son los sub-sistemas de control RKUs y sus especializaciones MKUs.

El proceso de coordinación es llevado a cabo por el RKU o MKU que detecta la excepción en tres etapas:

Etapa 1: se genera el conjunto de órdenes de análisis incluyendo toda orden de la agenda de uso con requerimiento asignado al período donde se produjo la excepción. Para el ejemplo de la Figura 2.9, corresponde a la orden 1. Si una solución no es hallada y el tiempo límite no ha sido alcanzado se avanza a la siguiente etapa.

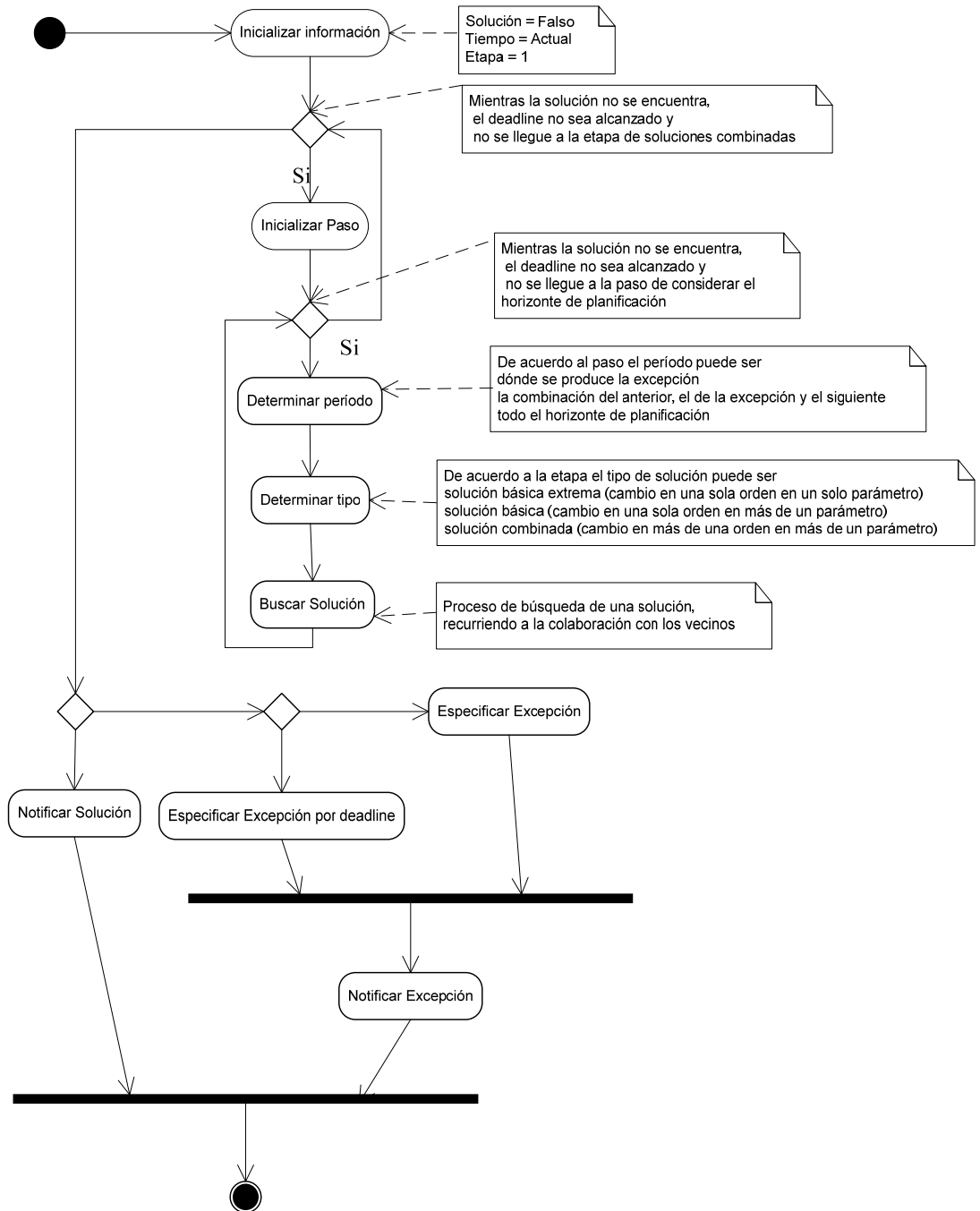
Etapa 2: se genera el conjunto de órdenes de análisis incluyendo toda orden de la agenda de uso con requerimiento asignado a los períodos inmediatamente anterior y posterior junto con el período en el que se produjo la excepción. Para el ejemplo de la Figura 2.9, corresponde a las órdenes 1, 2 y 3. De no hallarse una solución se pasa a la etapa siguiente.

Etapa 3: se genera el conjunto de órdenes de análisis incluyendo toda orden de la agenda de uso con requerimiento asignado a todo el horizonte del programa de abastecimiento. Para el ejemplo de la Figura 2.9, corresponde a las 6 órdenes.

El límite de la expansión está determinado por el tiempo. Una vez alcanzado el tiempo límite, si no ha sido encontrada una solución, la excepción es reportada.



En cada etapa los siguientes pasos son ejecutados antes de evolucionar a la próxima etapa de expansión:



**Figura 2.14: Lógica del proceso de coordinación**

Paso 1: *solución básica extrema*, consiste en generar propuestas de solución modificando uno solo de los parámetros de una única orden del conjunto de órdenes de análisis.

Paso 2: *solución básica*, consiste en generar propuestas de solución en las cuales para una única orden del conjunto de órdenes de análisis se proponen cambios en más de un parámetro.

Paso 3; *solución combinada*, consiste en generar propuestas de solución en las cuales para más de una orden del conjunto de órdenes de análisis se proponen cambios en más de un parámetro.

La lógica del proceso de coordinación descrito se representa en el diagrama de la Figura 2.14.

En cada uno de estos pasos se emplea un protocolo de interacción en el que se define claramente la secuencia de mensajes esperados a producirse cuando la conversación es iniciada. Dicho protocolo se especifica en el siguiente capítulo.

En el ejemplo de la Sección 2.3.2, el RKU que representa el recurso centro de envasado (CEnvasado), ante la ocurrencia del evento disruptivo recalculó el perfil de factibilidad, Figura 2.9, y detectó que la orden 1 resultó afectada. Ante la excepción el RKU debe constituirse en coordinador. En su agenda de uso, Tabla 2-1, identifica al SP-1 como proceso de suministro asociado a dicha orden a quien debe enviar la solicitud de colaboración. Dicha solicitud debe ir acompañada de propuestas de cambios viables. Ejemplo de propuestas de cambios viables para el RKU podrían ser:

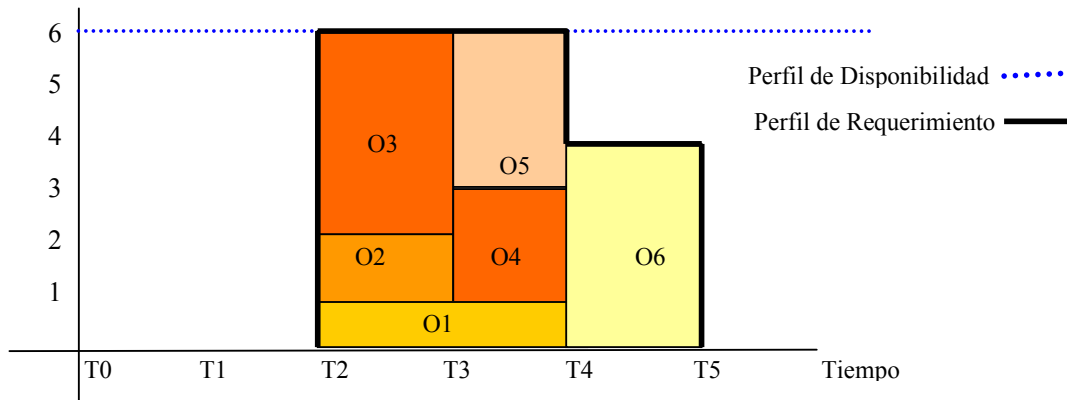
PRO1: Retrasar el inicio de la orden 1 al tiempo T2 (solución básica extrema).

PRO2: Retrasar el inicio de la orden 1 al tiempo T2 y reducir la cantidad de la orden 1 en 500 unidades (solución básica).

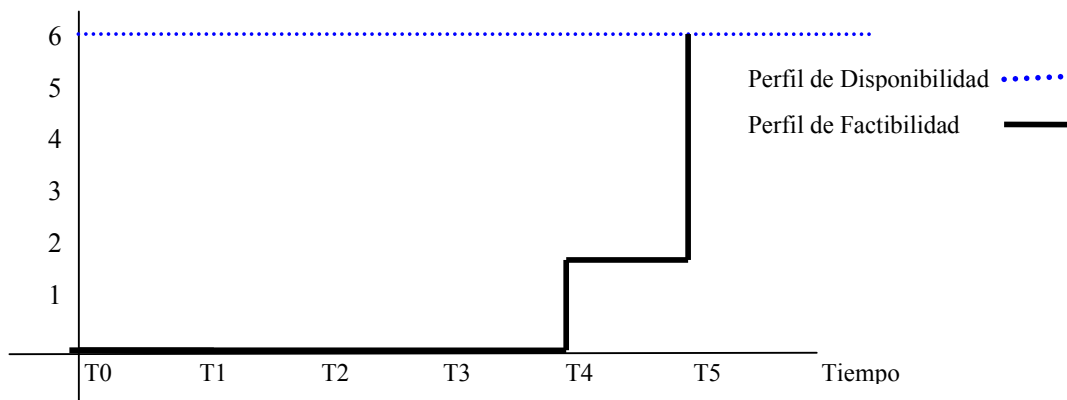
PRO3: Retrasar el inicio de la orden 1 al tiempo T2 y asignar dos unidades del recurso de envasado para que mantenga la fecha de finalización en T3, y reducir en 500 unidades la orden 3 (solución combinada).

PRO4: Retrasar el inicio de la orden 1 al tiempo T2 y asignar dos unidades del recurso de envasado para que mantenga la fecha de finalización en T3, y retrasar el inicio de la orden 2 al tiempo T3 (solución combinada).

Análisis de la propuesta PRO1: En la Figura 2.15 se representa el perfil de requerimientos y de disponibilidad para el RKU correspondiente a la PRO1, y en la Figura 2.16 se representa el perfil de factibilidad resultante. Como se observa, dicha solución es viable para el RKU.



**Figura 2.15: PRO1: Perfil de Requerimientos y de Disponibilidad para el RKU**



**Figura 2.16: PRO1: Perfil de Factibilidad para el RKU**

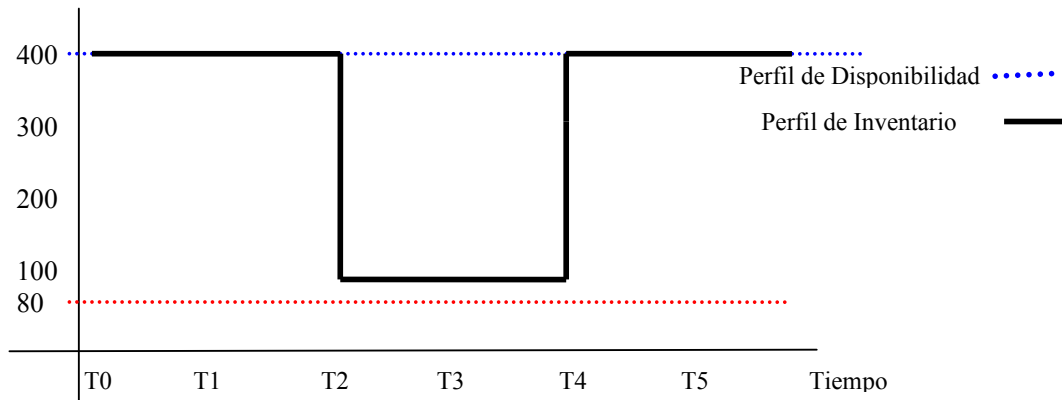
Supóngase que el RKU envía esta propuesta de solución al SP-1 anexa a la solicitud de colaboración. El SP-1 mediante su plan de actividades, Tabla 2-4, identifica a MKU-m1g-silo y MKU-m1e-almacén como participantes de la colaboración. El SP-1 debe transformar utilizando el plan de actividades la propuesta para cada MKU y enviarla con la solicitud de colaboración. La propuesta enviada a cada uno será:

PRO1- MKU-m1g-silo: Retrasar el inicio de la orden 1 al tiempo T2.

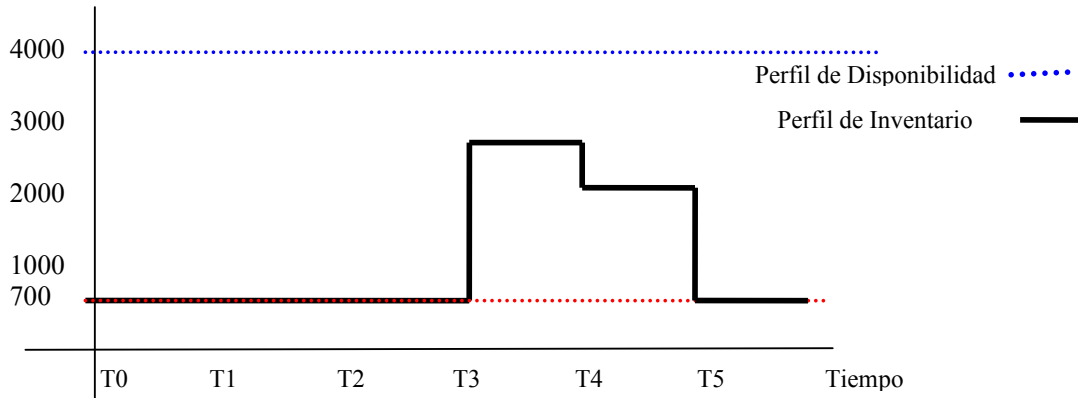
PRO1- MKU-m1e-almacén: Retrasar el inicio de la orden 1 al tiempo T4.

Al recibir la propuesta, cada MKU recalcula su perfil de inventario, los que se muestran en las Figuras 2.17 y 2.18. Como puede verse en ambas figuras, la propuesta

resulta factible, por lo cual ambos MKUs podrían responder aceptando la propuesta, con lo cual se resuelve la excepción.

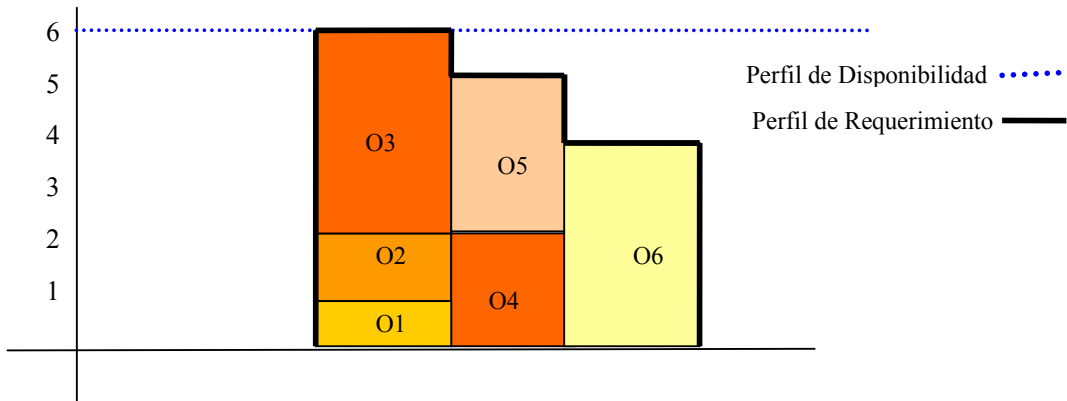


**Figura 2.17: PRO1: Perfil de inventario del MKU-m1g-silo**

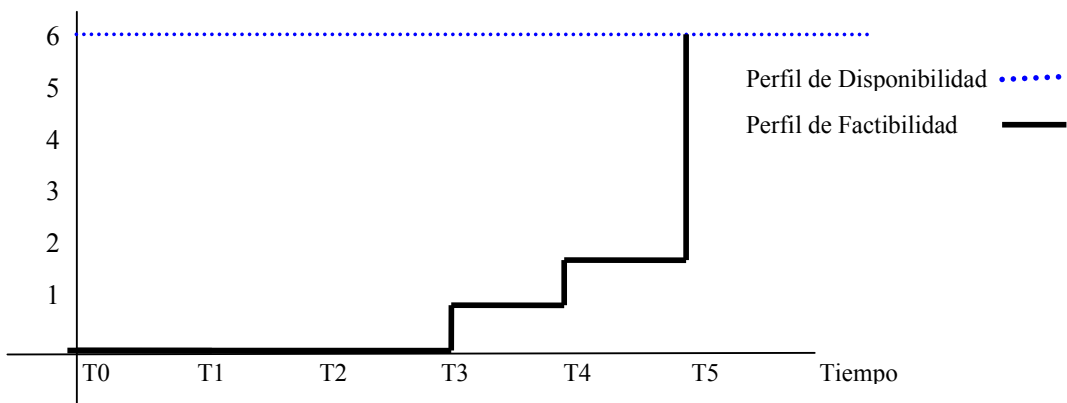


**Figura 2.18: PRO1: Perfil de inventario del MKU-m1e-almacén**

Análisis de la propuesta PRO2: En la Figura 2.19 se representa el perfil de requerimientos y de disponibilidad para el RKU correspondiente a la PRO2, y en la Figura 2.20 se representa el perfil de factibilidad resultante. Como se observa, dicha solución es viable para el RKU.



**Figura 2.19: PRO2: Perfil de Requerimientos y de Disponibilidad para el RKU**



**Figura 2.20: PRO2: Perfil de Factibilidad para el RKU**

Supóngase que el RKU envía esta propuesta de solución al SP-1 anexa a la solicitud de colaboración. El SP-1 mediante su plan de actividades, Tabla 2-4, identifica a MKU-m1g-silo y MKU-m1e-almacén como participantes de la colaboración. El SP-1 debe transformar utilizando el plan de actividades la propuesta para cada MKU y enviarla con la solicitud de colaboración. La propuesta enviada a cada uno será:

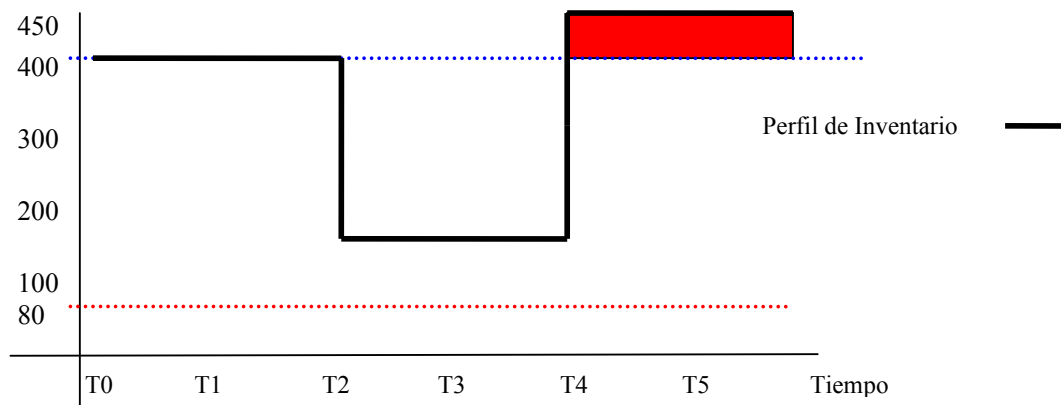
PRO2- MKU-m1g-silo: Retrasa el inicio de la orden 1 al tiempo T2 y reduce la cantidad en 50 unidades.

PRO2- MKU-m1e-almacén: Reduce la cantidad de la orden 1 en 500 unidades.

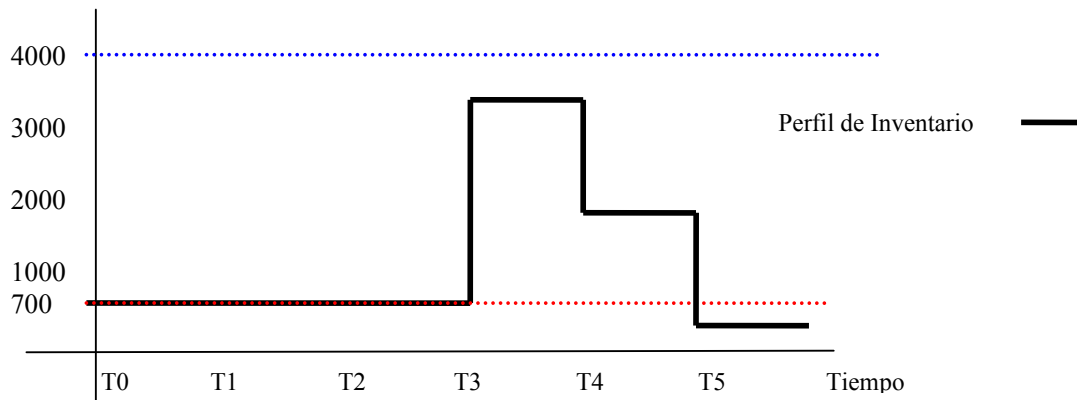
Al recibir la propuesta, cada MKU recalcula su perfil de inventario. El perfil de inventario del MKU-m1e-almacén se muestra en la Figura 2.22. Como puede observarse, si bien insume parte del stock de seguridad resulta factible, por lo cual podría responder aceptando la propuesta. En el caso del MKU-m1g-silo, el perfil de inventarios se representa en la Figura 2.21, en la cual se observa que se vuelve no

factible a partir del tiempo T4, siendo afectada la orden 3 de su lista de entrada/salida (Tabla 2-2). Ante esta situación el MKU-m1g-silo puede rechazar la propuesta o bien enviar al SP-5 una solicitud de colaboración, enviando como propuesta reducir la cantidad de la orden 3 en 50 unidades. Esto implicaría expandir el proceso de colaboración a un *segundo nivel*. La expansión del proceso de colaboración podría extenderse a más niveles. En esta tesis se limita la expansión sólo a mecanismos de colaboración de *primer nivel*.

Trabajando en el marco de un mecanismo de primer nivel, el MKU-m1g-silo debe rechazar la propuesta, con lo cual la propuesta PRO2 resultaría no factible.



**Figura 2.21: PRO2: Perfil de inventario del MKU-m1g-silo**

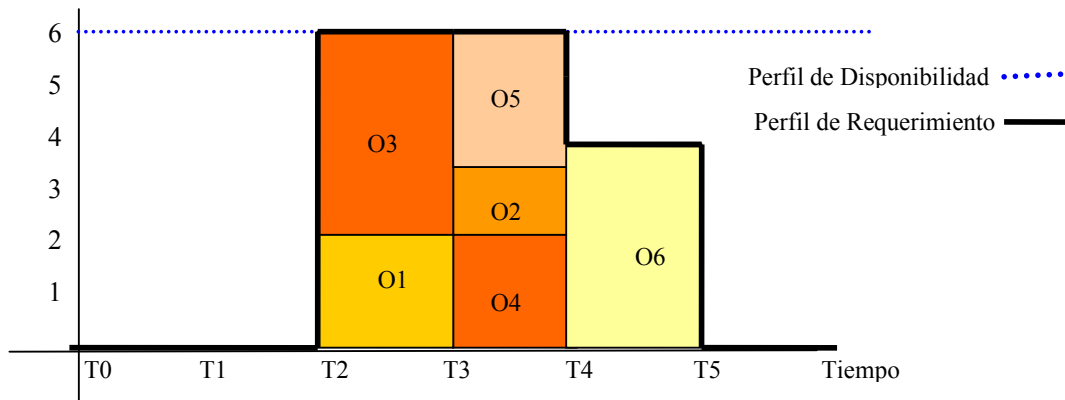


**Figura 2.22: PRO2: Perfil de inventario del MKU-m1e-almacén**

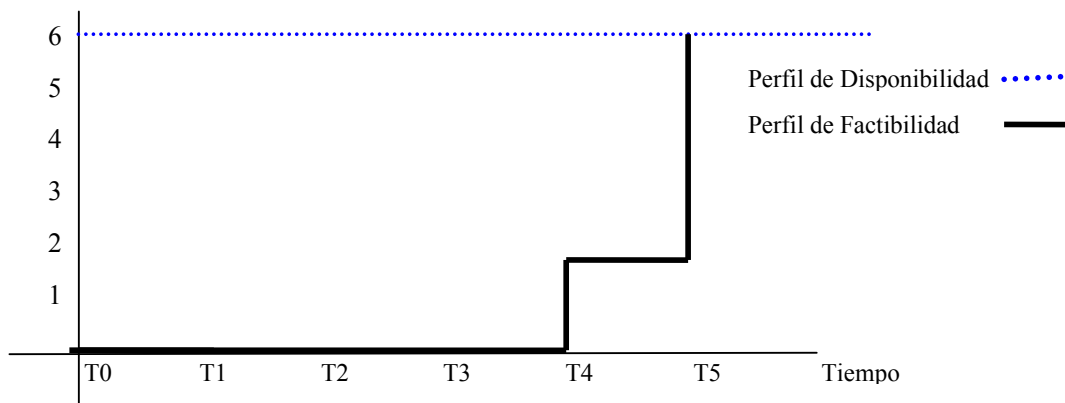
Un análisis equivalente para la propuesta PRO3 también indicaría que la solución no es factible para el MKU-m1g-silo, por lo cual resultaría no factible.

Análisis de la propuesta PRO4: En la Figura 2.23 se representa el perfil de requerimientos y de disponibilidad para el RKU correspondiente a la propuesta PRO4, y

en la Figura 2.24 se representa el perfil de factibilidad resultante. Como se observa, dicha solución es viable para el RKU. Para iniciar un proceso de colaboración en base a esta propuesta el RKU encontrará en su agenda de uso (Tabla 2-1) que: por la orden 1 debe enviar una solicitud de colaboración al SP-1 proponiendo retrasar el inicio de la misma al tiempo T2 y asignar dos unidades del recurso de envasado para que mantenga la fecha de finalización en T3; mientras que por la orden 2 debe enviar una solicitud de colaboración al SP-2 proponiendo retrasar el inicio de la orden 2 al tiempo T3. En este caso el RKU deberá recibir la respuesta favorable de ambos SPs para que la propuesta PRO4 sea factible y pueda ser implementada para resolver la excepción.



**Figura 2.23: PRO4: Perfil de Requerimientos y de Disponibilidad para el RKU**



**Figura 2.24: PRO4: Perfil de Factibilidad para el RKU**

Volviendo a la propuesta PRO1, el RKU podría plantear dicha propuesta de manera más flexible, ofreciendo alternativas. Por ejemplo:

PRO5: Retrasar el inicio de la orden 1 al tiempo T2, T3 o T4 (solución básica extrema).

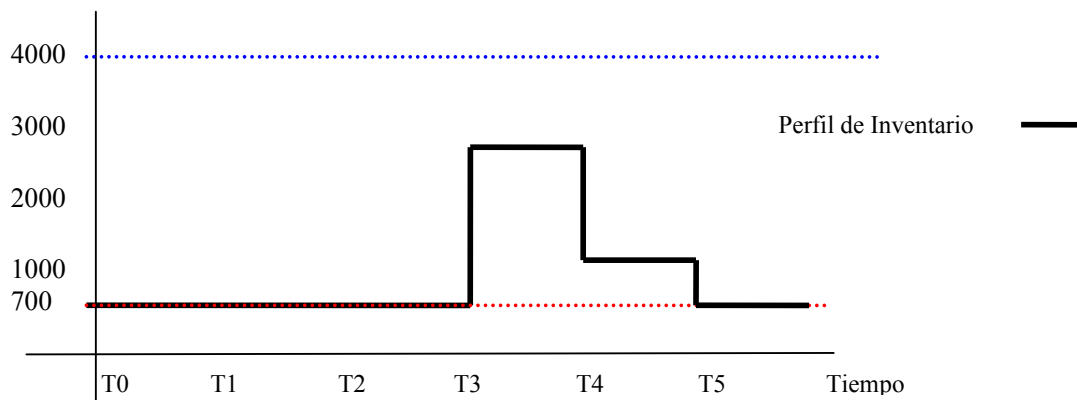
Es fácil verificar que los tres valores alternativos del cambio propuesto en el retraso del inicio de la orden 1 generan perfiles de factibilidad del RKU que resuelven la excepción, por lo cual PRO5 es una propuesta viable para el RKU. También es de destacar que sigue siendo una solución básica extrema.

Cuando el RKU envía esta propuesta de solución al SP-1 anexa a la solicitud de colaboración. El SP-1 mediante su plan de actividades, Tabla 2-4, identifica a MKU-m1g-silo y MKU-m1e-almacén como participantes de la colaboración. El SP-1 debe transformar utilizando el plan de actividades la propuesta para cada MKU y enviarla con la solicitud de colaboración. La propuesta enviada a cada uno será:

PRO5- MKU-m1g-silo: Retrasar inicio de la orden 1 al tiempo T2, T3 o T4

PRO5- MKU-m1e-almacén: Retrasar inicio de la orden 1 al tiempo T4, T5 o T6

Es fácil verificar que los tres valores del cambio propuesto en el retraso del inicio de la orden 1 generan perfiles de inventario factibles para el MKU-m1g-silo. Por lo cual podrá responder enviando como propuesta la PRO5 tal cual la recibió.



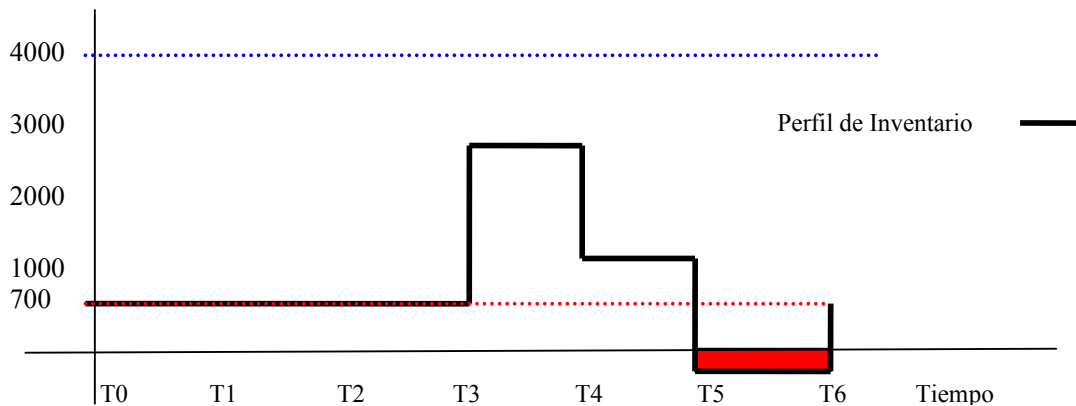
**Figura 2.25: PRO5: Perfil de inventario del MKU-m1e-almacén**

Para el MKU-m1e-almacén, los perfiles de inventario resultantes para cada uno de los tres valores de la PRO5 se muestran en las Figuras 2.18, 2.25 y 2.26. Como puede observarse, los dos primeros valores, que corresponden a retrasar el inicio de la orden 1 a los tiempos T4 y T5 generan perfiles de inventario factibles; mientras que el



tercer valor, que es retrasar el inicio de la orden 1 al tiempo T6 genera un perfil de inventarios negativo. Por esta razón, el MKU-m1e-almacén podrá responder enviando como propuesta:

PRO5r- MKU-m1e-almacén: Retrasar el inicio de la orden 1 al tiempo T4 o T5



**Figura 2.26: PRO5: Perfil de inventario del MKU-m1e-almacén**

Cuando el SP-1 recibe las propuestas enviadas por los MKUs en respuesta a la solicitud de colaboración, debe unificarlas y transformarla para generar la propuesta a enviar al RKU en respuesta a la solicitud de colaboración. La propuesta resultante enviada como respuesta será:

PRO5r: Retrasar el inicio de la orden 1 al tiempo T2 o T3.

Mediante el análisis realizado a esta última propuesta se puede observar que las propuestas recibidas en respuesta a una solicitud de colaboración pueden tener varias alternativas. El caso analizado fue el de una *solución básica extrema*, la cual propone cambiar un único parámetro de una orden. En el caso de una *solución básica* la propuesta es cambiar varios parámetros de una orden, y en el caso de una *solución combinada* la propuesta es cambiar varios parámetros de varias órdenes. Dado que cada propuesta de cambio puede tener vario valores alternativos, la cantidad máxima de alternativas que puede incluir una propuesta recibida por el RKU que envió la solicitud de colaboración resulta de realizar el producto cartesiano de dichos cambios. La cantidad real de alternativas surge de las combinaciones factibles que definirá el proponente.

Independientemente de la cantidad, cuando una propuesta tiene más de una alternativa, requiere tomar una decisión, para lo cual es necesario definir algún criterio. En esta tesis se propone utilizar una *función utilidad*. Esto implica que cada RKU o MKU proponente debe acompañar cada alternativa con un valor de utilidad definido en el dominio continuo  $[0,1]$ . El valor de cada alternativa es definido por el proponente con algún criterio que le permita ponderar la utilidad desde el punto de vista del recurso que representa. A su vez el RKU o MKU receptor de la propuesta, que es el coordinador del proceso de colaboración y quien debe tomar la decisión, también con algún criterio debe definir el valor de utilidad de cada alternativa desde el punto de vista del recurso que representa, para finalmente elegir la alternativa que mejor equilibra las utilidades proponente-receptor.

## 2.4. CONCLUSIONES

Los modelos desarrollados permiten capturar las características de la propuesta de solución para el problema de la gestión de eventos disruptivos en la cadena de suministro. El modelo de componentes principales del sistema SCEM distribuido respeta la autonomía de las entidades de negocio que integran la cadena de suministro y la heterogeneidad de los sistemas que componen el soporte informático de las mismas.

El modelo funcional del proceso de gestión de eventos disruptivos describe el comportamiento autónomo de cada entidad de negocio que compone la cadena de suministro en la gestión de un evento disruptivo y el comportamiento frente a la necesidad de una acción en colaboración con otra entidad de negocio. En base al mismo se desarrolló un modelo conceptual que permite representar el problema en cuestión como una red de puntos de control definidos sobre los recursos o materiales conectados entre si mediante procesos de suministro. Finalmente se definió el proceso de integración entre componentes, el proceso de identificación de una excepción y el proceso de coordinación para llevar a cabo una búsqueda de solución a una excepción en colaboración.

A diferencia de las propuestas existentes, las cuales centran el monitoreo de eventos disruptivos en las órdenes de abastecimiento, esta propuesta centra el monitoreo y las acciones de control en los recursos. Esto representa un aspecto innovativo en la forma de modelar el problema cuya ventaja radica en que los recursos son los

directamente afectados por los eventos disruptivos, mientras que los eventos disruptivos que afectan a una orden de abastecimiento son consecuencia de los que afectan los recursos asociados a la misma. Es decir las órdenes son afectadas de manera indirecta.

El enfoque propuesto permite anticipar la ocurrencia de una excepción, que afecta una o más órdenes, producida por un evento disruptivo sobre un recurso. Esto permite tener más y mejores alternativas de decisión a la hora de buscar una solución a la disrupción.

## CAPÍTULO 3 – ARQUITECTURA DE UN SISTEMA SCHEM BASADO EN AGENTES

En este capítulo se presenta una arquitectura multi-agente para la gestión de eventos disruptivos en la cadena de suministro desarrollada en base al modelo conceptual presentado en el capítulo anterior.

La finalidad de este capítulo es describir la arquitectura propuesta y detallar cómo a partir de las interacciones de sus componentes surge el comportamiento global de la agencia para satisfacer su objetivo, que es el de absorber con las holguras los desvíos producidos por las disrupciones en un programa de abastecimiento que se encuentra en ejecución.

### 3.1. ARQUITECTURA MULTI-AGENTE PARA SCHEM

El modelo conceptual del Sistema SCHEM propuesto en la sección anterior implica una compleja red de sub-sistemas de control autónomos y distribuidos, definidos cada uno de ellos sobre un recurso (RKU) o sobre un material (MKU), que llevan a cabo acciones en colaboración, valiéndose de los vínculos definidos por los procesos de suministro (SP) que hacen uso de dichos materiales y recursos. Para implementar este modelo, se ha elegido la tecnología de agentes de software porque es una solución innovadora para sistemas constituidos por entidades autónomas y distribuidas donde el comportamiento global emerge a partir de la interacción de sus entidades [Anosike and Zhang, 2002; Bussmann et al, 2004; Mas, 2005].

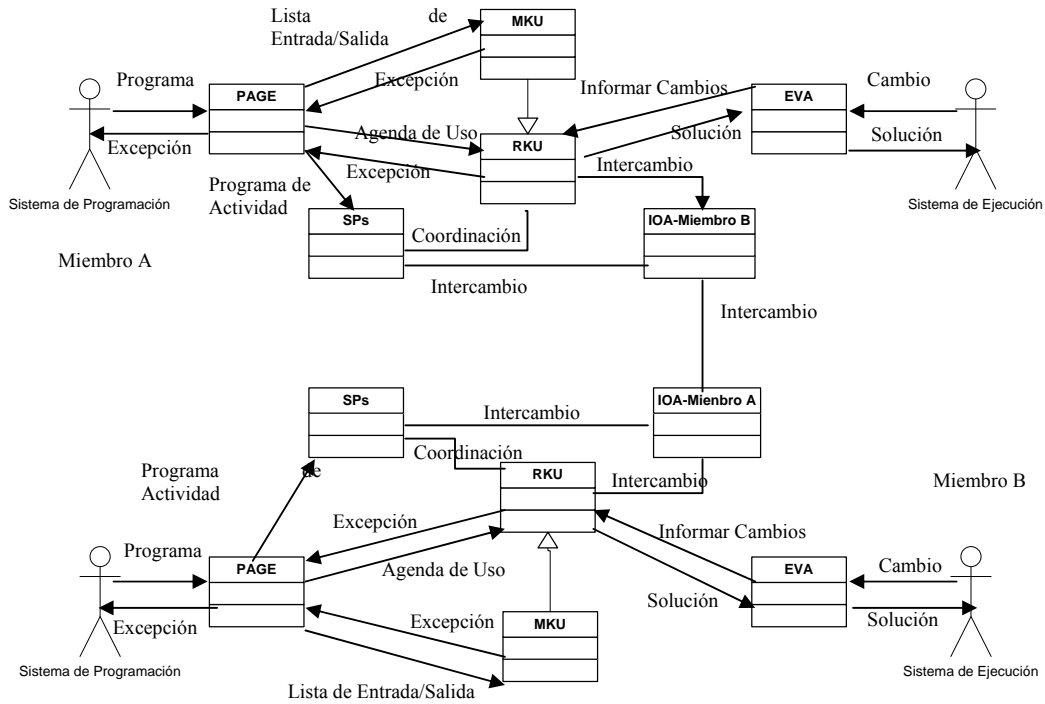
En base a esta tecnología, el Sistema SCHEM de cada empresa es modelado como un sistema-multiagente denominado *agencia*. Los agentes de software definidos para integrar la agencia pueden ser agrupados en dos conjuntos, los agentes de interfaz responsables de la interacción del Sistema SCHEM con los otros sistemas intra-organizacionales e inter-organizacionales, y los agentes responsables de representar los recursos y los materiales de la organización y cómo dichos recursos se relacionan. Esta clasificación se define a partir de los roles identificados en el modelo conceptual.

En la Figura 3.1 se representan gráficamente los componentes e interacciones de la arquitectura multi-agente propuesta para el caso particular del Sistema SCEM de las Empresas A y B de una cadena de suministro que participan del proceso de colaboración. Los agentes identificados son: El agente *EVA*, responsable de la interfaz con el Sistema de Ejecución de la empresa; el agente *PAGE* que tiene a su cargo la interacción con el Sistema de Programación de la empresa; el agente *IOA*, responsable de la interacción entre las agencias de cada empresa que participa del proceso de colaboración; el agente *SP* representando al proceso de suministro de cada orden del programa de abastecimiento, que vincula un conjunto de recursos y de materiales; el agente *RKU* que desempeña las funciones del sub-sistema de control definido sobre un recurso; y el agente *MKU* que desempeña las funciones del sub-sistema de control definido sobre un material.

En una agencia existirá un agente *RKU* por cada recurso y un agente *MKU* por cada material relevantes para el programa de abastecimiento en ejecución bajo monitoreo, y un agente *SP* por cada orden que integra dicho programa. Cada orden representa un proceso de suministro, según la definición de programa de abastecimiento dada el Capítulo 1, Sección 1.2. Los agentes han sido diseñados siguiendo el modelo BDI [Rao and Georgeff, 1995].

En base a la arquitectura multi-agente propuesta, un Sistema SCEM se puede definir como una red de agentes *RKU* (sub-sistemas de control definido sobre recursos) y agentes *MKU* (subsistemas de control definido sobre materiales) interconectados entre sí por medio de agentes *SP* (cada uno representando a uno de los siete tipo de transformaciones que puede definir una orden, definidas en el Capítulo 2, Sección 2.3.2). Para su operatoria, estos agentes cuentan con agentes especializados en funciones de servicio de interfaz: agente *PAGE*, agente *EVA* y agente *IOA*.

Así definido, el comportamiento global del Sistema SCEM emerge del comportamiento individual autónomo de cada agente y el comportamiento colectivo dentro de cada agencia y de las interacciones entre las agencias, sin que exista un elemento central responsable de llevar a cabo este proceso. El propósito es que cada empresa preserve su autonomía, pero a su vez colabore con otras empresas para el bien común.



**Figura 3.1: Arquitectura basada en agentes de un sistema SCEM: Componentes e Interacciones**

## 3.2. ARQUITECTURA DE LOS AGENTES

### 3.2.1. AGENTE PAGE

Las características estructurales del agente PAGE se presentan en la Figura 3.2 mediante un diagrama de clases AUML [Bauer, 2001]. PAGE es un agente de servicio responsable de la interfaz de la agencia con el Sistema de Programación. Un rol importante, *Instanciador\_de\_Agencia*, es el de crear las instancias de los restantes agentes de la agencia. El agente PAGE recibe del Sistema de Programación el programa de abastecimiento a gestionar; determina los puntos de control necesarios para representar la dinámica del programa; por cada recurso crea el agente RKU definiendo su agenda de uso; por cada material crea el agente MKU definiendo su lista de entradas/salidas; y para cada proceso de suministro crea el agente SP definiendo su programa de actividades. Crea también los agentes IOA y EVA que brindarán los correspondientes servicios de interfaz.

Su otro rol, *Gestor\_de\_Excepción*, es en sentido inverso, cuando un agente RKU o MKU no puede encontrar una solución a una disrupción producida por un evento, notifica la excepción al agente PAGE y éste es el responsable de enviar un mensaje al Sistema de Programación indicando la excepción. Este escenario conduce a la necesidad de una re-programación, lo cual implica que la agencia espera recibir un nuevo programa de abastecimiento para gestionar, por esta razón, luego de notificada la excepción el elimina a todos los agentes creados para gestionar el programa de abastecimiento previo.

Para el desempeño de los roles asignados, el agente PAGE ha sido diseñado como un agente reactivo ya que frente a los estímulos de su medio determinará las actividades a ser desarrolladas como respuestas a dichos estímulos. De este modo la función de monitoreo se ha limitado a la espera de un mensaje enviado por el Sistema de Ejecución (estímulo).

El *módulo de comunicación* del agente PAGE es el responsable de: recibir los mensajes entrantes y enviarlos al *módulo motor* para su evaluación y toma de decisión; y de enviar mensajes tanto a los otros agentes de la agencia como al Sistema de Programación. Los mensajes de entrada válidos a ser recibidos por el *módulo de comunicación* son: la notificación de un nuevo programa de abastecimiento y la notificación de una excepción (Figura 3.1). Los mensajes de salida que puede producir el agente PAGE luego de llevar a cabo sus procesos de decisión en respuesta a los mensajes de entrada son: los mensajes vinculados con la creación/eliminación de los agentes y la notificación de excepción al Sistema de Programación.

En el diagrama de estados, el primer estado es *EsperandoMensaje*, que pueden ser de dos tipos: recibir un nuevo programa de abastecimiento para gestionar; o recibir una excepción. En el primer caso, pasa al estado *InstanciandoAgencia*, luego de crear la agencia retorna al estado *EsperandoMensaje*. Si el mensaje recibido es una excepción, pasa al estado *ProcesandoExcepción*, lo cual implica eliminar los agentes que constituyen la agencia, luego pasa al estado *NotificandoExcepción* y finaliza su ciclo.

Los principales recursos necesarios para satisfacer su objetivo están relacionados con los conocimientos requeridos para crear a los otros elementos de la agencia, para ello recibe el *Programa\_de\_Abastecimiento* y conoce la *Política\_de\_Instanciación* que

le permite obtener la información para crear los agentes requeridos para representar el programa de abastecimiento recibido. Los datos de la excepción son informados a través de un *Informe\_de\_Excepción* por el agente RKU o MKU responsable de encontrar una solución a una disrupción producida por un evento. Esta información es notificada al Sistema de Programación

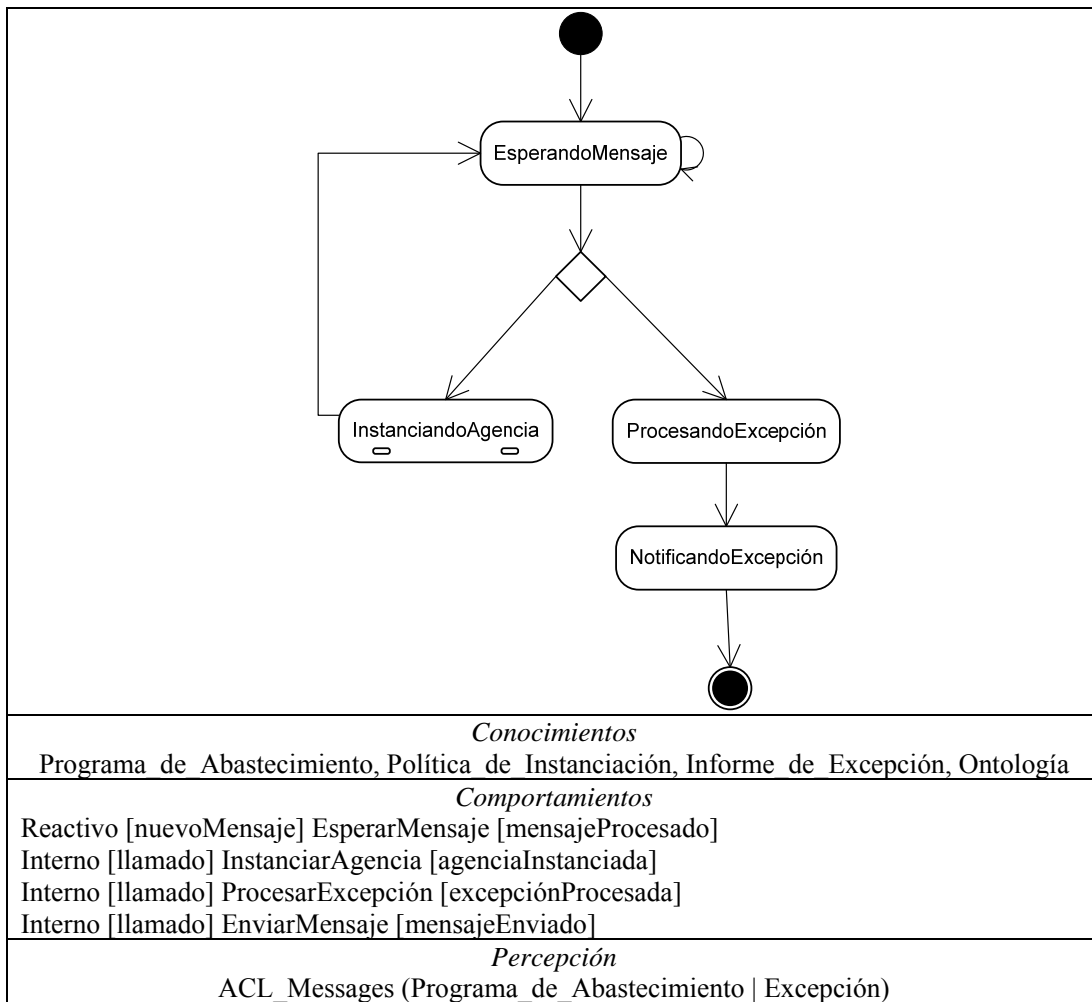
Mediante el comportamiento reactivo *EsperarMensaje* recibe los mensajes de otros agentes o sistemas, analiza su contenido y lo hace disponible a los otros comportamientos del agente. El comportamiento interno *EnviarMensaje* es llamado por otros comportamientos, puede modificar el ambiente enviando mensajes al entorno (al Sistema de Programación).

El rol de *Instanciador\_de\_Agencia* está vinculado con el comportamiento interno *InstanciarAgencia* que es iniciado cuando recibe un nuevo *Programa\_de\_Abastecimiento* a ser gestionado. El rol de *Gestor\_de\_Excepción* está vinculado con el comportamiento interno *ProcesarExcepción* que es iniciado cuando recibe la excepción, esto implica recibir la excepción con la información asociada, eliminar los agentes que componen la agencia y llamar al comportamiento interno *EnviarMensaje*. El mensaje de notificación de excepción, además de ser enviado al Sistema de Programación, de acuerdo al tipo de excepción podría ser enviado a usuarios por otros canales tales como correo electrónico.

Los dos tipos de mensajes percibidos por el agente PAGE, *Programa\_de\_Abastecimiento* o *Excepción*, son basados en el lenguaje de comunicación de agentes ACL [Mas, 2005].

<<agente>> PAGE
<i>Roles</i> Instanciador de Agencia, Gestor de Excepción
<i>Diagrama de Estados</i>





**Figura 3.2: Diagrama de clases AUML de un agente PAGE**

### 3.2.2. AGENTE EVA

Las características estructurales del agente EVA se presentan en la Figura 3.3. Es un agente de servicio de la agencia responsable de la interfaz con el Sistema de Ejecución. Un rol importante es el de *Gestor\_de\_Eventos* cuya función es recibir desde el Sistema de Ejecución los cambios ocurridos (eventos disruptivos internos) y los eventos disruptivos de frontera y enviarlos a los agentes RKU o MKU correspondientes. Un segundo rol es el de actuar como *Despertador* de dichos agentes, los cuales se registran en él para “dormir”. El despertar se puede producir porque: un evento disruptivo de su interés ha acontecido; o el agente PAGE solicita al agente EVA despertar a todos los agentes RKU y MKU porque debe eliminar la actual agencia. Esto tiene lugar cuando se

produce una excepción que implicará una operación de re-programación o se ha completado el programa de abastecimiento.

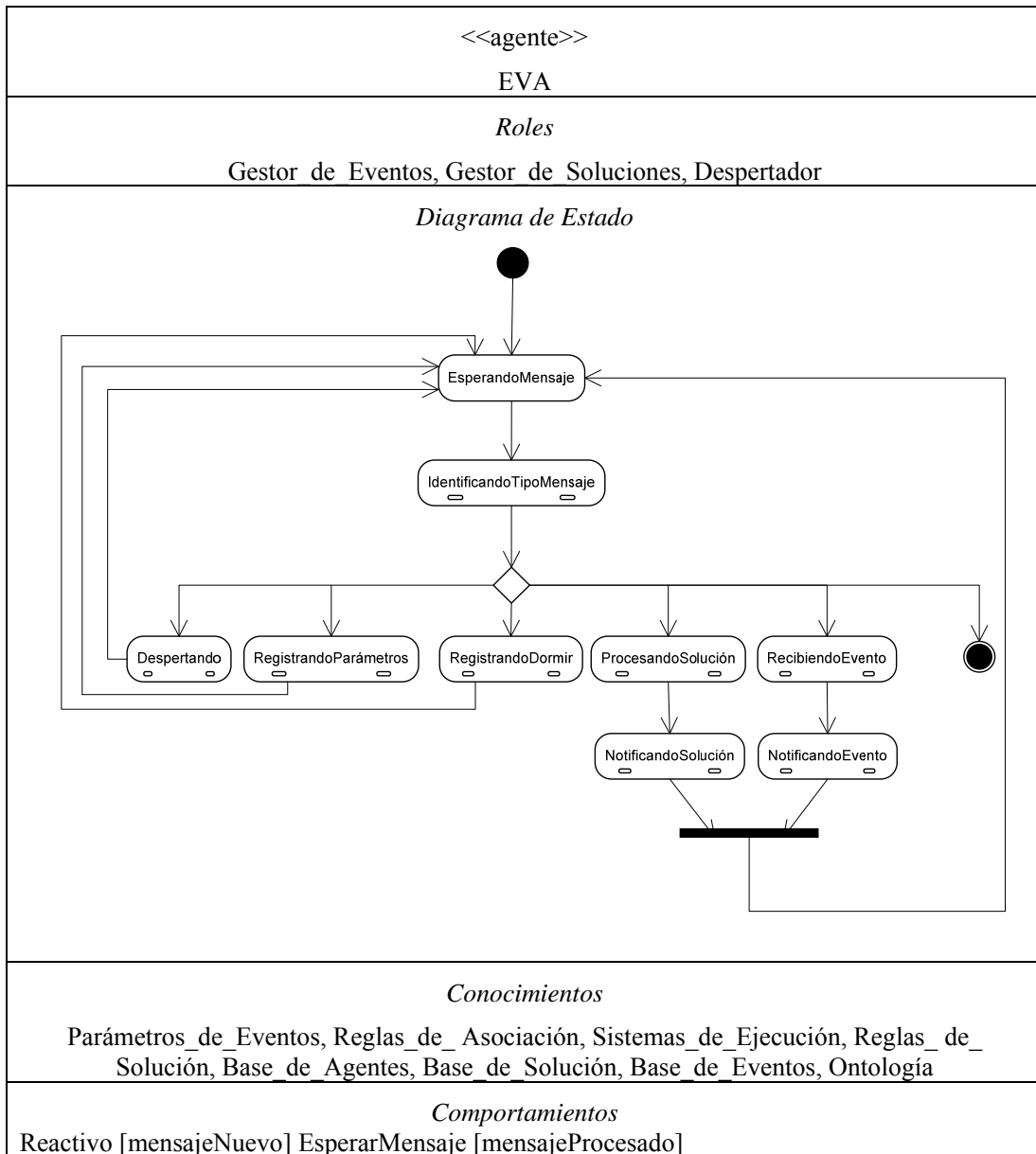
Un tercer rol, *Gestor\_de\_Soluciones*, es en sentido inverso, cuando una solución a un evento disruptivo ha sido hallada, el agente RKU o MKU que ha llevado a cabo el proceso de búsqueda de la solución notifica la misma al agente EVA. Al recibir esta información, el agente EVA debe notificar al Sistema de Ejecución las modificaciones realizadas al programa de abastecimiento para resolver la disrupción generada, a efectos de que dicho sistema las implemente.

Para el desempeño de los roles asignados, la arquitectura propuesta para el agente EVA corresponde al tipo reactiva y es similar a la detallada para el agente PAGE. Su *módulo de comunicación* es responsable de: recibir los mensajes entrantes y enviarlos al *módulo motor* para su evaluación y toma de decisión; y de enviar mensajes a los agentes RKU, MKU y al Sistema de Ejecución. Los mensajes de entrada válidos a ser recibidos por el *módulo de comunicación* son: la notificación de un evento disruptivo interno o de un evento disruptivo de frontera; la notificación de una solución; y los mensajes relacionados con la solicitud de despertar a los agentes RKU y MKU. Los mensajes de salida que puede producir el agente EVA luego de ejecutar sus procesos de decisión en respuesta a los mensajes de entrada son: la notificación de una solución al Sistema de Ejecución; la notificación de un evento disruptivo al agente RKU o MKU directamente afectado por el mismo, el cual asumirá la responsabilidad de gestionar dicho evento; y los mensajes para despertar a los agentes.

Los mensajes de entrada disparan las acciones del agente, las que son seleccionadas de la librería de acciones. Además de la librería de acciones el agente EVA debe almacenar cuál es la variable monitoreada para cada agente RKU o MKU, información que utiliza para “mapear” un evento disruptivo a una de ellas. Otra información asociada con estos agentes es referente a si se encuentran inactivos (dormidos).

En el diagrama de estados, el primer estado es *EsperandoMensaje*, el cual será de cinco tipos: recibir un evento disruptivo; recibir una solución; recibir parámetros de agentes RKU y MKU; agentes a dormir; y despertar agentes. Cada vez que recibe un mensaje pasa al estado *IdentificandoTipoMensaje*. Si el mensaje es del primer tipo, pasa al estado *RecibiendoEvento*, procesa el mensaje recibido y define a qué agente RKU o

MKU debe enviar el evento disruptivo, luego pasa al estado *NotificandoEvento*, si estaba durmiendo procede a despertarlo y envía el mensaje, luego retorna al estado de espera de mensajes. Si el mensaje recibido es una solución, pasa al estado *ProcesandoSolución*, generando el mensaje a enviar, luego pasa al estado *NotificandoSolución* para luego retorna al estado de espera de mensajes. Los otros tres tipos de mensaje recibidos implican los estados: *Despertando*, cuando lo requiere el agente PAGE; *RegistrandoDormir*, cuando lo solicita el propio agente RKU; y *RegistrandoParámetros* informados por el agente PAGE al momento de recibir un nuevo programa de abastecimiento, y luego de haberlo creado.



Interno [mensajeProcesado] RegistrarParámetroEvento [baseEventoActualizada]
Interno [mensajeProcesado] NotificarEvento [eventoProcesado]
Interno [mensajeProcesado] RecibirSolución [baseSoluciónActualizada]
Interno [solución] NotificarSolución [soluciónProcesada]
Interno [llamado] EnviarMensaje [mensajeEnviado]
Interno [mensajeProcesado] RegistrarDormir [baseAgentesActualizada]
Interno [mensajeProcesado  evento] Despertar [baseAgentesActualizada]
<i>Percepción</i>
Mensajes ACL (Informa_Parámetros_Evento  Informa_Solución   Informa_Periodo_Dormir   Informa_Evento   Solicita_Despertar)

**Figura 3.3: Diagrama de clases AUML para el agente EVA**

Para poder cumplir con su objetivo los recursos que el agente EVA necesita están relacionados con los conocimientos requeridos para cumplir sus roles, para ello recibe al momento de su creación por parte del agente PAGE: *Parámetro\_de\_Eventos*, *Reglas\_de\_Asociación*, *Sistema\_de\_Ejecución* y las *Reglas\_de\_Solución*. A su vez gestiona la *Base\_de\_Agentes*, *Base\_de\_Solución* y la *Base\_de\_Eventos*.

El comportamiento *EsperarMensaje* es del tipo reactivo y le permite recepcionar los mensajes provenientes de los agentes con los que se relaciona y desde el Sistema de Ejecución. La función de este comportamiento es recibir los mensajes, analizar su contenido y hacerlo disponible para sus otros comportamientos. Por otro lado el comportamiento interno *EnviarMensaje* es invocado por otros comportamientos. A través de ellos puede modificar su entorno.

El rol *Gestor\_de\_Eventos* se encuentra relacionado con los comportamientos *RegistrarParametroEvento* y *NotificarEvento*. El comportamiento *RegistrarParametroEvento* es iniciado al recibir un mensaje del agente PAGE cuando un nuevo programa de abastecimiento ha sido recibido, este comportamiento registra en su base de conocimientos cuál es la variable monitoreada por cada RKU/MKU, también actualiza los *Parámetros\_de\_Eventos* las *Reglas\_de\_Asociación* y la *Base\_de\_Agentes*.

El rol de *Gestor\_de\_Soluciones* se relaciona con los comportamientos internos *RecibirSolución* y *NotificarSolución*. El comportamiento *RecibirSolución* es iniciado cuando recibe un mensaje desde un RKU o MKU notificándole que una solución a un evento ha sido hallada, frente a esto actualiza la *Base\_de\_Soluciones* y la *Base\_de\_Eventos*, luego invoca el comportamiento *NotificarSolución*, el cual empleando el conocimiento *Sistema\_de\_Ejecución* y *Reglas\_de\_Solución*, comunicará

al Sistema de Ejecución sobre los cambios realizados en el programa de abastecimiento en ejecución.

Los comportamientos *RegistrarDormir* y *Despertar* están asociados con el rol *Despertador* y hacen uso del conocimiento *Base\_de\_Agentes*. El comportamiento *RegistrarDormir* es iniciado por un mensaje recibido desde un agente RKU o MKU indicando que estará inactivo. El comportamiento *Despertar* puede tener su origen en un mensaje recibido de parte del agente PAGE solicitando despertar a todos los agentes, o por un evento recibido relacionado con el agente RKU o MKU.

La percepción del agente EVA se basa en la recepción de mensajes ACL, estos mensajes son: *Informa\_Parametros\_Evento*, *Informa\_Período\_Dormir*, *Informa\_Solución*, *Informa\_Evento* y *Solicita\_Despertar*.

---

### 3.2.3. AGENTE IOA

El comportamiento global del Sistema SCEM surge de las interacciones del conjunto de Sistemas SCEM implementados en cada una de las empresas de la cadena de suministro que participan del proceso de colaboración (Capítulo 2, Sección 2.1). IOA es un agente de servicio cuyo rol es el de *Gestor\_de\_Comunicaciones\_inter\_Organizacionales*. Las características estructurales del agente IOA se presentan en la Figura 3.4. El desempeño de su rol implica comportarse como si fuese un agente MKU y transferir los mensajes de coordinación con otra agencia. Entre dos agentes IOA relacionados solo se transfieren mensajes sin mayor análisis ni decisiones al respecto. Cada agencia define un agente IOA por cada agencia con la que tiene una relación de colaboración. El agente IOA es creado por el agente PAGE, el cual le provee la identidad del agente IOA de la agencia de la empresa relacionada y el programa de órdenes de suministro acordado con dicha empresa (Capítulo 2, Sección 2.3.3).

La arquitectura del agente IOA es reactiva, al recibir un mensaje proveniente de un agente SP de su agencia se comporta como si fuese un agente MKU de la otra agencia, encapsulando el mensaje y enviándolo al agente IOA de la agencia relacionada correspondiente. En sentido inverso, si recibe un mensaje proveniente de un agente IOA de una agencia relacionada, el cual inicialmente corresponde a un evento disruptivo *de colaboración*, selecciona el agente SP destinatario, envía el mensaje y gestiona

posteriormente la secuencia de mensajes a intercambiar (Capítulo 2, Sección 2.2) pero no participa de los procesos de decisión.

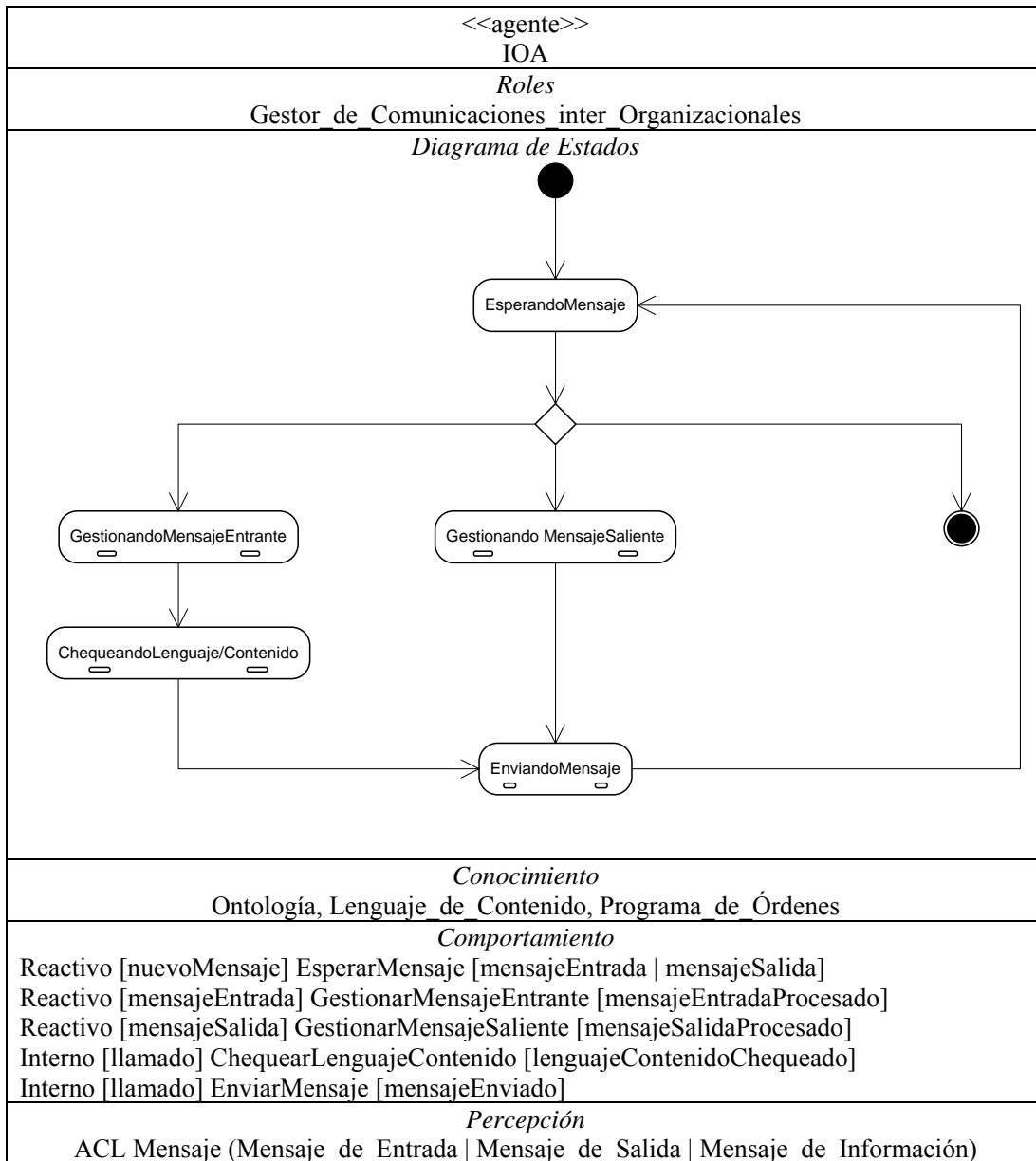
En el diagrama de estado, el primer estado es *EsperandoMensaje*. Si el mensaje recibido es enviado por un SP de la agencia pasa al estado *GestionandoMensajeSaliente*, valiéndose del programa de órdenes de suministro identifica la orden afectada y pasa al estado *EnviandoMensaje* para proceder a enviar el mensaje al agente IOA de la agencia relacionada, volviendo al estado de espera de un nuevo mensaje. Si el mensaje recibido es de un agente IOA de la agencia relacionada, pasa al estado *GestionandoMensajeEntrante*, en base a la información de la orden afectada identifica el SP al que debe enviar el mensaje y pasa al estado *ChequeandoLenguaje/Contenido* donde verifica la validez del mensaje en cuanto al lenguaje intercambiado y el contenido que presenta, posteriormente pasa al estado *Enviando Mensaje* para proceder a enviar el mensaje al agente SP, para luego volver al estado de espera de un nuevo mensaje.

Los principales recursos necesarios para satisfacer su objetivo están relacionados con los conocimientos requeridos para gestionar el proceso de comunicación con un agente externo perteneciente a otra agencia. Para ello recibe el *Programa\_de\_Órdenes* y conoce la *Ontología* y el *Lenguaje\_de\_Contenido*.

Mediante el comportamiento reactivo *EsperarMensaje* recibe los mensajes de otros agentes de la agencia (salientes) y los hace disponible al comportamiento reactivo *GestionarMensajeSaliente* para su procesamiento; y los mensajes de agentes de otra agencia (entrantes) y los hace disponible al comportamiento reactivo *GestionarMensajeEntrante* para su procesamiento. Una vez procesados son puestos a disposición de los otros comportamientos internos del agente: *EnnviarMensaje* o *ChequearLenguajeContenido*. El primero es llamado por los comportamientos de gestión de mensajes, mientras que el último es llamado por el comportamiento que gestiona los mensajes entrantes.

Las dos percepciones principales de un agente IOA son: *Mensaje\_de\_Entrada* originado en la agencia relacionada, con destino a un agente interno de la agencia; y *Mensaje\_de\_Salida* propinado en un agente interno de la agencia, dirigido a la agencia relacionada. Una tercera percepción, *Mensaje\_de\_Información*, está relacionada con la

información inicial recibida del agente PAGE. Los mensajes son basados en el lenguaje de comunicación de agentes ACL.



**Figura 3.4: Diagrama de clases AUML para el agente IOA**

### 3.2.4. AGENTE RKU

El agente RKU cumple la función de sub-sistema de control autónomo responsable de gestionar los eventos disruptivos que afectan al recurso que representa. Es creado por el agente PAGE, recibiendo como entrada la agenda de uso y el perfil de

disponibilidad (Sección 2.3.2). Por medio de la agenda de uso conoce el conjunto de órdenes que utilizan el recurso, lo cual le proporciona el conocimiento de su entorno.

Las características estructurales del agente RKU se presentan en la Figura 3.5. En su rol de *Gestor\_de\_Eventos*, gestiona dos tipos de eventos: directos (asociados con cambios en la disponibilidad del recurso que representa) e indirectos (provenientes de una solicitud de colaboración en la búsqueda de solución emitida por un agente SP relacionado). En el caso de un evento directo asume el rol de *Iniciador\_Coordinación*, mientras que en el caso de un evento indirecto asume el rol de *Participante\_Coordinación* a efectos de llevar a cabo las funciones pertinentes según se definió en la Sección 2.3.3.

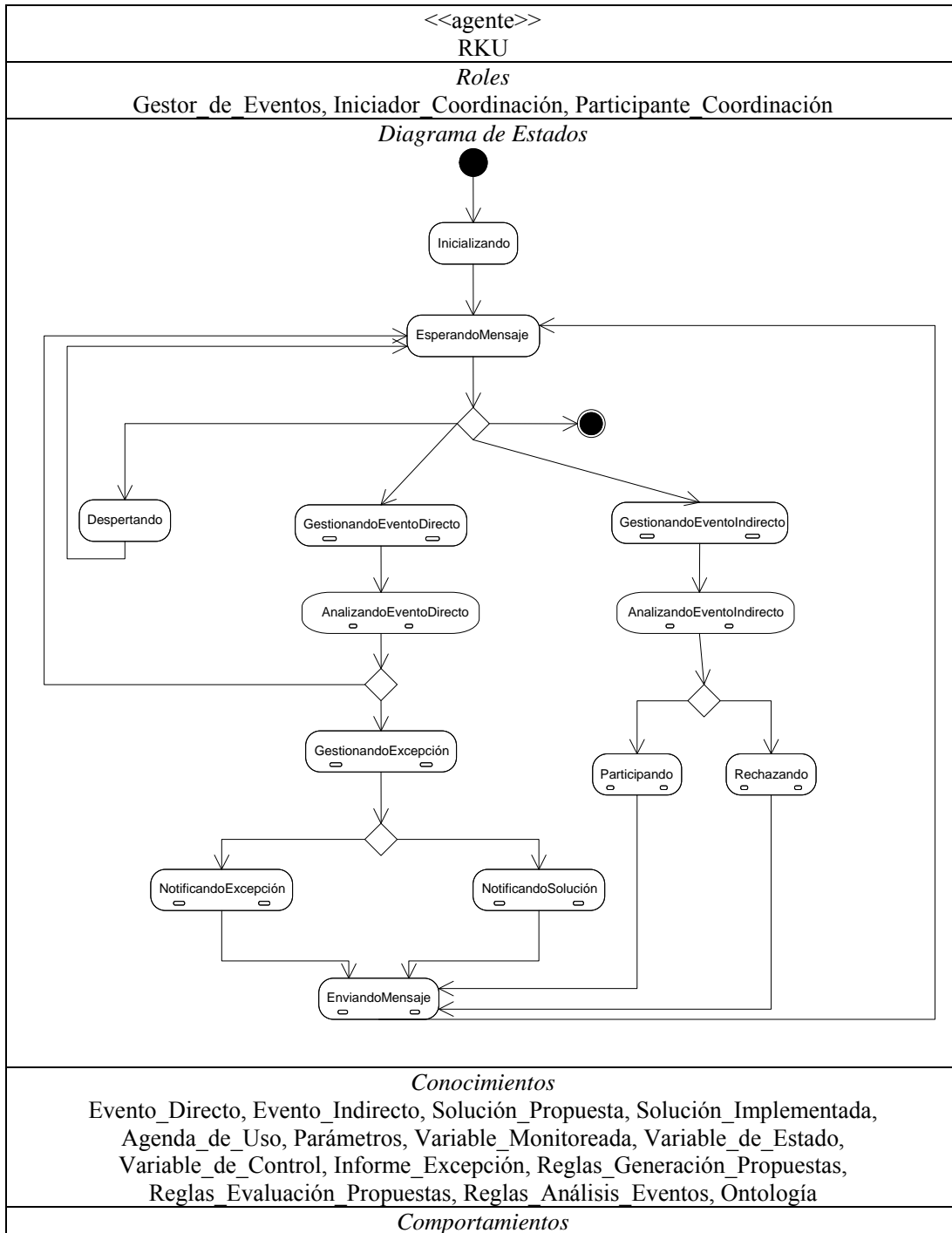
El agente RKU ha sido diseñado en base a la arquitectura BDI (*Belief, Desire and Intention*) [Rao and Georgeff, 1995]. Las creencias están determinadas por sus *Conocimientos*, para ello recibe al momento de su creación la *Agenda\_de\_Uso*, *Parámetros*, *Variable\_Monitoreada*, *Variable\_de\_Estado* y *Variable\_de\_Control*. La información sobre un evento directo es proporcionada por el agente EVA y registrada en *Evento\_Directo*, en tanto que la información sobre un evento indirecto es almacenada en *Evento\_Indirecto*. Los datos sobre *Solución\_Implementada* son enviados al agente EVA y si corresponde a los agentes SP involucrados, en forma análoga información sobre *Solución\_Propuesta* es enviada al agente SP que solicitó la participación. También informa al agente PAGE cuando una solución a una disrupción no ha sido hallada a través de un *Informe\_de\_Excepción*. El conjunto de conocimientos le permiten conocer su entorno y los agentes a él asociados.

Las distintas reglas que forman parte del conocimiento del agente, *Reglas\_Generación\_Propuestas*, *Reglas\_Evaluación\_Propuestas*, *Reglas\_Análisis\_Eventos* son parte de la definición del agente, que tiene lugar durante su creación.

El objetivo de un agente RKU es absorber la disrupción causada por un evento utilizando las holguras del recurso que representa o recurriendo a un llamado a colaboración si fuese necesario. Para ello una librería de planes especifica las acciones que puede realizar, mientras que el motor de inferencia define cómo y cuándo los planes son considerados en función del estado actual.



En el diagrama de estados, el primer estado es *Inicializando*, en el cual el agente RKU es creado por el agente PAGE; luego pasar al estado *EsperandoMensaje*. Los mensajes a recibir pueden ser de cuatro tipos: despertarse (enviado por el agente EVA), evento directo (enviado por el agente EVA), evento indirecto (enviado por un agente SP), o eliminarse (enviado por el agente PAGE).



Reactivo [nuevoMensaje] EsperarMensaje [mensajeProcesado] Reactivo [eliminarse] EliminarAgente [eliminadoRKU] Reactivo [eventoDirecto] GestionarEventoDirecto [solución   excepción] Reactivo [eventoIndirecto] GestionarEventoIndirecto [participa   noParticipa] Reactivo [despertarse] DespertarRKU [rkuActivo] Cíclico [ ] AnalizarAgenda [continua   duerme] Interno [eventoDirecto] CoordinarBúsquedaSolución [solución   excepción] Interno [eventoIndirecto] ParticiparEventoIndirecto [soluciónPropuesta] Interno [dormirse] NotificarDormir [rkuDormido] Interno [llamado] EnviarMensaje [mensajeEnviado]
<i>Percepción</i> Mensajes-ACL (Informa_Instanciación   Informa_Evento_Directo   Cfp_Evento_Indirecto   Informa Despertar   Informa Eliminar)

**Figura 3.5: Diagrama de clases AUML del agente RKU**

Dependiendo del mensaje evoluciona al estado *Despertando*, *GestionandoEventoDirecto*, *GestionandoEventoIndirecto* o estado final. En el estado *Despertando* el agente se vuelve activo, retomando su funcionamiento en el estado *EsperandoMensaje*. *GestionandoEventoDirecto* es un estado en el cual registra en su base de conocimiento el evento recibido para pasar luego al estado *AnalizandoEventoDirecto*, donde el evento disruptivo es analizado para determinar en qué medida afecta al programa en ejecución. Si el evento es absorbido por las holguras del recurso, retorna al estado *EsperandoMensaje*. En caso contrario pasa al estado *GestionandoExcepción* en el cual el agente RKU asume el rol de coordinador del proceso de búsqueda de una solución en colaboración con otros agentes del entorno. Al finalizar pasa al estado *NotificandoSolución* donde genera los mensajes a enviar a los participantes del proceso informando la solución a implementar y al agente EVA para que comunique las modificaciones al Sistema de Ejecución; o bien pasa al estado *NotificandoExcepción* donde genera el mensaje de excepción a enviar al agente PAGE notificando que el programa en ejecución se ha vuelto no factible. Concreta el envío de mensajes en el estado *EnviandoMensaje*, y luego retorna al estado *EsperandoMensaje*.

*GestionandoEventoIndirecto* es un estado al cual arriba como consecuencia de un llamado a colaboración por parte de un SP de su entorno. Pasa al estado *AnalizandoEventoIndirecto* en el cual realiza simulaciones de potenciales cambios en la agenda de uso producto de este evento; si algunos de los cambios son factibles pasa al estado *Participando* en el que formaliza la participación en el proceso de colaboración. Pasa luego al estado *EnviandoMensaje* para confirmar su participación. Si las simulaciones no arrojan un resultado positivo pasa al estado *Rechazando* y

posteriormente al estado *EnviandoMensaje* para informar al SP el rechazo a participar. Por último, si el mensaje recibido solicita su eliminación el agente finaliza su ciclo de vida.

El comportamiento reactivo *EsperarMensaje* recibe los mensajes provenientes de los otros agentes, analiza su contenido y lo deja disponible a los otros comportamientos del agente. El comportamiento interno *EnviarMensaje* es llamado por otros comportamientos, pudiendo modificar su ambiente por medio de mensajes a otros agentes relacionados.

Los comportamientos *GestionarEventoDirecto* y *GestionarEventoIndirecto* están relacionados con el rol *Gestor\_de\_Eventos*. El rol *Iniciador\_Coordinación* se vincula con el comportamiento *CoordinarBúsquedaSolución*, mientras el rol *Participante\_Coordinación* se asocia al comportamiento *ParticiparEventoIndirecto*.

Además de los comportamientos mencionados, posee otros relacionados con la funcionalidad del agente: *EliminarAgente*, *DespertarRKU*, *AnalizarAgenda*, *NotificarDormir*. El comportamiento *AnalizarAgenda* es del tipo cíclico y tiene por finalidad identificar períodos de inactividad durante los cuales puede estar “dormido”; cuando identifica este período llama al comportamiento *NotificarDormir*, mediante el cual el agente RKU notifica al agente EVA su período de inactividad y cuándo debe ser despertado. Mediante el comportamiento *DespertarRKU* el agente vuelve a un estado activo.

El agente RKU percibe su medio ambiente por medio de mensajes tipo ACL, estos mensajes son: *Informar\_Instanciación* (recibido desde el agente PAGE conteniendo todos los datos de inicialización), *Informar\_Evento\_Directo* (proveniente del agente EVA en el que se notifica al agente de un evento de su interés), *Cfp\_Evento\_Indirecto* (originado en un SP relacionado solicitando participar de un proceso de colaboración), *Informar\_Despertar* (enviado por el agente EVA para volver operativo al agente RKU) y por último el *Informar\_Eliminar* (originado por el agente PAGE en el que notifica la baja del agente).

---

### 3.2.5. AGENTE MKU

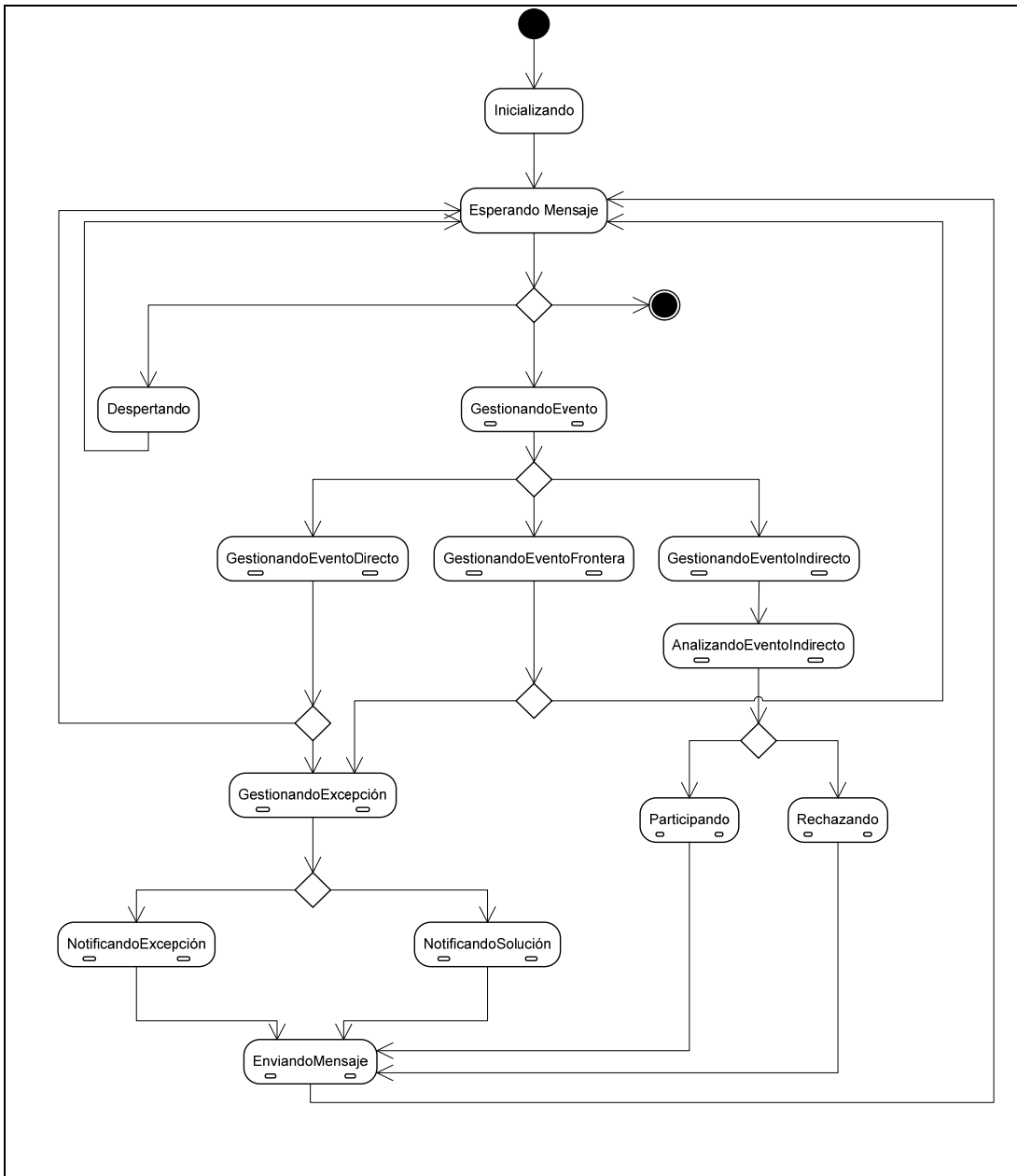
El agente MKU es una especialización del agente RKU dedicado a la gestión de eventos disruptivos que afectan el inventario del material que representa. Es creado por el agente PAGE recibiendo su lista de entrada/salida y como parámetros los valores para los tres atributos básicos que definen el material que representa: material, envase y lugar. También recibe como parámetros el stock inicial y el stock de seguridad, los cuales junto a la lista de entrada/salidas le permiten calcular el perfil de inventario programado.

Las características estructurales del agente MKU se presentan en la Figura 3.6. En su rol de *Gestor\_de\_Eventos*, gestiona tres tipos de eventos: directos (asociados con cambios en la disponibilidad del material que representa), indirectos (provenientes de una solicitud de colaboración emitida por un agente SP relacionado) y de frontera (asociados con cambios en órdenes de su lista de entrada/salida vinculadas a entidades de negocio que no participan del proceso de colaboración). En el caso de un evento directo o de frontera asume el rol de *Iniciador\_Coordinación*, mientras que en el caso de un evento indirecto asume el rol de *Participante\_Coordinación* a efectos de llevar a cabo las funciones pertinentes según se definió en la Sección 2.3.3.

El agente MKU también ha sido diseñado en base a la arquitectura BDI. Los *Conocimientos* que definen sus creencias son los mismos que el de un agente RKU, a diferencia de la *Lista\_Entrada/Salida*, que reemplaza a la agenda de uso, y la información sobre un evento de frontera enviada por el agente EVA que es registrada en *Evento\_Frontera*.

El diagrama de estados comienza en el estado *Inicializando*, en el cual el agente MKU es creado por parte del agente PAGE. Una vez creado pasa al estado *EsperandoMensaje*, en el que permanece hasta que recibe uno de los cinco mensajes posibles: despertarse (enviado por el agente EVA), evento directo (enviado por el agente EVA), evento indirecto (enviado por un agente SP), evento de frontera (enviado por el agente EVA) o eliminarse (enviado por el agente PAGE).

<<agente>> MKU
<i>Roles</i> Gestor_de_Eventos, Iniciador_Coordinación, Participante_Coordinación
<i>Diagrama de Estados</i>



*Conocimientos*

Evento\_Directo, Evento\_Indirecto, Evento\_Frontera, Solución\_Propuesta,  
 Solución\_Implementada, Lista\_Entrada/Salida, Parámetros, Variable\_Monitoreada,  
 Variable\_de\_Estado, Variable\_de\_Control, Informe\_Excepción,  
 Reglas\_Generación\_Propuestas, Reglas\_Evaluación\_Propuestas, Reglas\_Análisis\_Events,  
 Ontología

*Comportamientos*

Reactivo [nuevoMensaje] EsperarMensaje [mensajeProcesado]  
 Reactivo [eliminarAgente] EliminarAgente [eliminadoMKU]  
 Reactivo [eventoDirecto] GestionarEventoDirecto [solución | excepción]  
 Reactivo [eventoFrontera] GestionarEventoFrontera [solución | excepción]  
 Reactivo [eventoIndirecto] GestionarEventoIndirecto [participa | no Participa]  
 Reactivo [despertarse] DespertarMKU [mkuActivo]  
 Cíclico [] AnalizarListaE/S [continua | duerme]

Interno [eventoDirecto] CoordinarBúsquedaSolución [solución   excepción]
Interno [eventoIndirecto] ParticiparEventoIndirecto [solucionesPropuestas]
Interno [dormirse] NotificarDormir [mkuDormido]
Interno [llamado] EnviarMensaje [mensajeEnviado]
<i>Percepción</i>
Mensajes-ACL (Informa_Instanciación  Informa_Evento_Directo  Informa_Evento_Frontera  Cfp_Evento_Externo   Informa_Despertar   Informa_Eliminar)

**Figura 3.6: Diagrama de clases AUMML para el agente MKU**

Dependiendo del tipo de mensaje recibido se activa uno de los siguientes estados: *Despertando*, *GestionandoEvento* o realiza la transición al estado final. En el estado *Despertando* retorna a la actividad pasando al estado *EsperandoMensaje*. En el estado *GestionandoEvento* determina el tipo de evento a efectos de definir el próximo estado al cual evolucionar. *GestionandoEventoDirecto* y *GestionandoEventoIndirecto* son estados análogos a los descriptos para el agente RKU. En el estado *GestionarEventoFrontera* actualiza la lista entrada/salida y analiza si el evento de frontera produce una excepción, en cuyo caso se convierte en agente coordinador pasando al estado *GestionandoExcepción*. Si no genera excepción, pasa al estado *EsperandoMensaje*.

En cuanto a los comportamientos, la única diferencia en relación al agente RKU es el comportamiento reactivo *GestionarEventoFrontera*, asociado con el rol *Gestor\_Eventos*, que es iniciado cuando ha percibido un evento de frontera.

### 3.2.6. AGENTE SP

El agente SP representa un proceso de suministro (producción o distribución) que define la transición de uno o más agentes MKU a uno o más agentes MKU, pudiendo requerir uno o más agentes RKU para su ejecución. Es decir, su función es actuar como un mediador entre agentes MKU y RKU relacionados.

Los agentes SP no son puntos de control, por lo cual solo pueden ser afectados por eventos indirectos con origen en alguno de los agentes RKU o MKU relacionados que requieren de su mediación.

El agente SP es creado por el agente PAGE recibiendo en ese momento el *Plan\_de\_Actividad* y las *Reglas\_de\_Transformación*, que forman parte de su conocimiento.

Las características estructurales del agente SP se detallan en la Figura 3.7. Es diseñado como un agente reactivo, donde el estímulo es un mensaje notificando un evento, identificado como *cfp* (*Call for Proposal*). El módulo de comunicación es el responsable de recibir los mensajes para luego enviarlo al módulo motor, y de enviar mensajes a los agentes RKU y MKU relacionados. Los mensajes de entrada y de salida se encuentran definidos en el protocolo de interacción que sustenta el proceso de colaboración entre agentes RKU y MKU de búsqueda e implementación de una solución. Dicho protocolo, identificado como DCNP (*Double Contract Net Protocol*) se describe posteriormente en la Sección 3.3.1.

El agente SP lleva a cabo el rol de *Mediador*, el que se compone de dos roles: *Iniciador* y *Respondedor*. Los detalles de estos roles se definen en el protocolo DCNP.

El funcionamiento del agente se describe en el diagrama de estados. El estado *Inicializando* corresponde al proceso de creación del agente por parte del agente PAGE; luego pasar al estado *EsperandoMensaje*. Los mensajes a recibir pueden ser de tres tipos: *cfp* (enviado por un agente MKU o RKU solicitándole mediar en un proceso de colaboración para la búsqueda de una solución); propuesta (enviada por un agente RKU o MKU en respuesta a un llamado a colaborar enviado); o eliminarse (enviado por el agente PAGE). Al recibir un *cfp* pasa al estado *RecibiendoSolicitud* donde recibe un documento de negocio con los detalles del llamado a colaboración. Pasa al estado *Enviando\_CFPs*, en el cual mediante el *Plan\_de\_Actividad* que forma parte de sus conocimientos identifica los agentes RKU y MKU que deben participar del proceso de colaboración. Mediante las *Reglas\_de\_Transformación* que también forman parte de sus conocimientos, realiza la conversión del documento de negocio recibido generando el correspondiente documento de negocio para cada agente participante, el cual envía anexo a un mensaje *cfp*. Una vez enviados pasa al estado *RecibiendoPropuestas*.

En el flujo normal de una negociación exitosa el agente SP permanece en este estado hasta recibir los mensajes con la *propuesta* de todos los agentes RKU y MKU participantes. Cada propuesta recibida es guardada en la base de *Propuestas* que formarán parte de sus conocimientos. Una vez recibidas todas las propuestas, pasa al estado *ProcesandoPropuestas* donde intenta combinar las propuestas en una única propuesta para el RKU iniciador. Si esta combinación es exitosa, guarda la respuesta en la base de *Respuestas* que formará parte de sus conocimientos, genera el

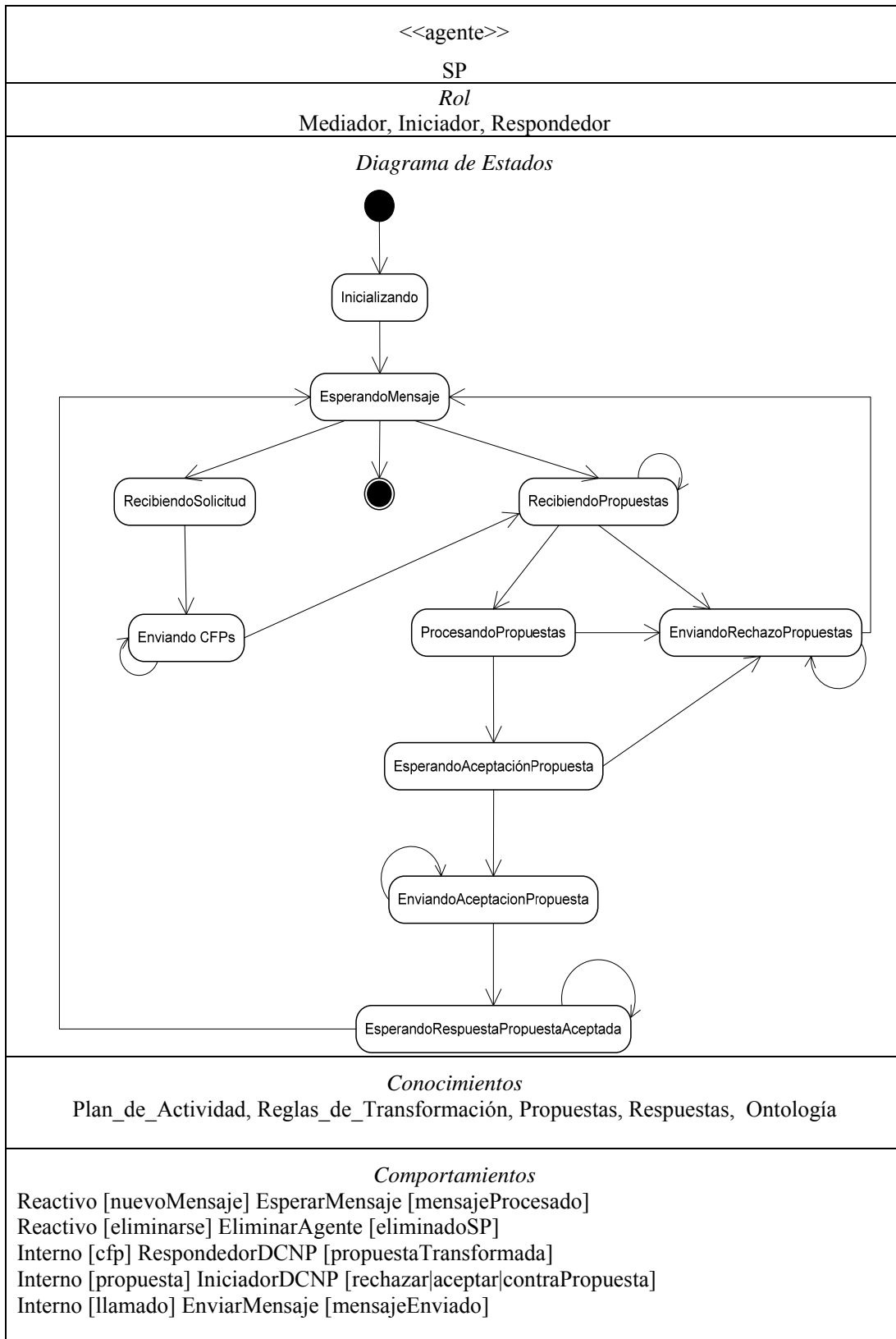
correspondiente documento de negocio utilizando las *Reglas\_de\_Transformación* y lo envía al agente RKU o SKU iniciador, luego pasa al estado *EsperandoAceptaciónPropuesta*. Si la propuesta es aceptada pasa al estado *EnviandoAceptaciónPropuesta* en el cual, usando las *Reglas\_de\_Transformación* convierte el documento de negocio enviado por el agente iniciador generando el correspondiente documento de negocio para cada agente participante con la propuesta aceptada, y lo envía anexo a un mensaje *cfp* a cada uno de los agentes RKU y SKU participantes. Luego pasa al estado *EsperandoConfirmaciónRespuestaAceptada*, donde espera la confirmación de recepción de la respuesta aceptada por parte de cada uno de los agentes participantes. Recibidas todas las confirmaciones, envía un mensaje de confirmación al agente RKU o SPU iniciador y da por finalizada de manera exitosa la negociación volviendo al estado *EsperandoMensaje*.

El flujo de una negociación no exitosa, puede transcurrir por tres caminos diferentes: Camino 1: encontrándose el agente SP en el estado *RecibiendoPropuestas*, si algún agente RKU o SKU participante envía un mensaje rechazando el *cfp*, pasa al estado *EnviandoRechazoPropuestas*. En esta estado envía en primer lugar un mensaje rechazando el *cfp* al RKU iniciador, y luego un mensaje rechazando la propuesta a cada uno de los agentes participantes que ya enviaron su propuesta. Permanece en dicho estado hasta recibir las propuestas de los restantes participantes, cada una de las cuales es automáticamente rechazada, para finalmente dar por terminada la negociación volviendo al estado *EsperandoMensaje*.

Camino2: encontrándose en el estado *ProcesandoPropuestas*, si al combinar las propuestas no logra generar una combinación factible, pasa al estado *EnviandoRechazoPropuestas*. En esta estado envía en primer lugar un mensaje rechazando el *cfp* al RKU iniciador, y luego un mensaje rechazando la propuesta a cada uno de los agentes participantes, para finalmente dar por terminada la negociación volviendo al estado *EsperandoMensaje*.

Camino3: encontrándose en el estado *EsperandoAceptaciónPropuesta*, si la propuesta es rechazada por el agente RKU o SKU iniciador, pasa al estado *EnviandoRechazoPropuestas* y envía un mensaje rechazando la propuesta a cada uno de los agentes participantes, para finalmente dar por terminada la negociación volviendo al estado *EsperandoMensaje*.





<i>Percepciones</i> Mensajes-ACL(Cfp   Rechaza_Propuesta   Acepta_Propuesta   Contra_Propuesta   Implementa_Solución)
---

**Figura 3.7: Diagrama de clases AUML de un agente SP**

El agente SP presenta los comportamientos *EsperarMensaje*, *EnviarMensaje*, además de los comportamientos asociados a su rol de mediador: *ParticipanteDCNP* e *IniciadorDCNP*. El comportamiento *EsperarMensaje* es el encargado de recibir los mensajes provenientes de los agentes RKU y MKU relacionados, o del agente PAGE notificando que debe eliminarse, disparando el comportamiento *EliminarAgente*. El comportamiento *EnviarMensaje* es invocado por los otros comportamientos para el envío de mensajes a los agentes RKU y MKU relacionados, pudiendo así modificar su entorno.

Al igual que en los otros agentes los mensajes percibidos por el agente SP son basados en el lenguaje ACL y constituyen sus mecanismos de percepción. Los mensajes *Cfp*, *Rechaza\_Propuesta*, *Acepta\_Propuesta*, *Contra\_Propuesta* e *Implementa\_Solución* provienen de su relación con los agentes RKU y MKU a través del protocolo DCNP.

### 3.3. PROCESO DE COORDINACIÓN

La agencia propuesta se caracteriza por su naturaleza distribuida y por la autonomía de sus miembros. El comportamiento global deseado surge del comportamiento individual de cada agente en la interacción con los otros agentes para la búsqueda de una solución a una excepción. En este contexto el concepto de coordinación se torna relevante.

La coordinación requiere definir la información a intercambiar y el proceso de interacción. La estructura del proceso de interacción entre dos agentes generalmente se corresponde con patrones típicos de secuencias de mensajes denominados protocolos de interacción. Un protocolo define las posibles secuencias de mensajes entre dos agentes, de modo que, cuando un agente desea interactuar con otro miembro de su comunidad conoce los mensajes a enviar y su formato, y también conoce las posibles respuestas a esperar y su formato.

FIPA (*Foundation for Intelligent Physical Agents*) ha especificado diferentes protocolos de interacción de agentes, de éstos por sus características particulares

interesan el *FIPA CNIP (Contract Net Interaction Protocol)* y el *FIPA ICNP (Iteration Contract Net Interaction Protocol)*. Estos protocolos se utilizan para permitir que un agente iniciador solicite la ejecución de una tarea a uno o más agentes, la diferencia entre ambos es que *FIPA ICNP* permite múltiples rondas iterativas. Ambos protocolos están basados en el protocolo *CN (Contract Net)* desarrollado por Smith y Davis [Smith, 1980], incorporando los mensajes de rechazo y de confirmación.

Por las características propias del dominio y la forma en que el problema fue modelado la interacción entre los agentes es de naturaleza compleja y distribuida. El llamado a colaboración para resolver una excepción implica coordinar dos o más interacciones entre pares de agentes. Los protocolos propuestos por FIPA no cubren este tipo de interacción por lo cual fue necesario desarrollar un protocolo específico. Dicho protocolo se describe a continuación.

---

### 3.3.1. PROTOCOLO DE INTERACCIÓN

La comunicación entre los agentes que integran una agencia se lleva a cabo a través de protocolos de interacción, los cuales son definidos mediante una secuencia permitida de mensajes entre los agentes participantes y la estructura del contenido de dichos mensajes.

El protocolo *CN (Contract-Net)* [Wellman, 1994] es el más usado para la asignación de recursos y tareas entre un grupo de agentes [Oprea, 2003]. Varios protocolos han sido propuestos en base al protocolo *CN*, entre ellos, el protocolo *MCN (Mediated Contract Net)*, cuyo objetivo es facilitar la creación dinámica de “clusters” de agentes y proveer transacciones de colaboración [Leitao, 2001]. En este protocolo la colaboración puede ser directa (*recruiting mechanism*) o indirecta (*brokering mechanism*).

Para dar soporte al proceso de colaboración que deben llevar a cabo los agentes de la agencia propuesta fue necesario extender el protocolo *CN*, en base al mecanismo de colaboración indirecta del protocolo *MCN*, el que se ha denominado protocolo *DCN (Double Contract Net)*. Este protocolo fue diseñado para dar soporte a una interacción *CN* entre un agente *RKU* o *MKU* iniciador y un agente *SP*. El agente *SP* a su vez inicia varias interacciones *CN* con otros agentes *RKU* o *MKU* denominados participantes. En base a las respuestas obtenidas de los participantes, el agente *SP* genera las respuestas a

enviar al agente iniciador. Este proceso es denominado *negociación simple*. Corresponde al caso de la propuesta PRO1 en el ejemplo de la Sección 2.3.3.3 del Capítulo 2.

Dependiendo de la estrategia de generación de propuestas, el agente iniciador puede necesitar interactuar con más de un agente SP. En este caso debe ejecutar el protocolo DCN  $n$  veces, una por cada agente SP. Este proceso es denominado *negociación compleja*. Corresponde al caso de la propuesta PRO4 en el ejemplo de la Sección 2.3.3.3 del Capítulo 2.

En ambos tipos de negociación los cambios en las órdenes son propuestos y acordados entre agentes RKU y MKU. Para esto se valen del *documento de negocio* intercambiado, anexo a los mensajes del protocolo, los cuales contiene las propuestas. Los cambios propuestos son expresados por cada agente iniciador o participante en términos de las órdenes locales (cada agente RKU o MKU conoce sus propios números de orden, cantidad y tiempos; no sabe cómo se relaciona con las órdenes de los otros agentes), es el agente SP el responsable de realizar las transformaciones correspondientes, valiéndose para ello del plan de actividades y las reglas de transformación definidas en base al mismo. En la Sección 2.3.3.3 del Capítulo 2 se presentaron ejemplos de estas transformaciones.

En el protocolo DCN propuesto el agente SP actúa como mediador, no participa en las decisiones en sí, pero realiza ciertas funciones que son vitales para que los agentes RKU y MKU puedan llevar a cabo el proceso de colaboración. Mediante el plan de actividades el agente SP debe realizar las conversiones del documento de negocio, en cuanto a número de orden, cantidad y tiempos. Ante la llegada de un mensaje cfp desde el agente iniciador el agente SP debe:

- ❖ Identificar los agentes que deben participar del proceso de colaboración, con los cuales debe iniciar una interacción.
- ❖ Con cada mensaje recibido del agente iniciador, debe realizar la transformación del documento de negocio para enviarlo al correspondiente agente participante.
- ❖ Mantener la sincronía entre las diferentes interacciones en las que interviene.
- ❖ Recibir los mensajes de los agentes participantes, combinar las propuestas contenidas en cada documento de negocio y generar una propuesta única a enviar al agente iniciador. En la Sección 2.3.3.3 del Capítulo 2 se presentó un

ejemplo de esta transformación para el caso de la propuesta PRO5.

El protocolo DCN propuesto se representa mediante un diagrama de secuencias en la Figura 3.8. Por simplicidad el agente iniciador y los agentes participantes son referidos como RKU, pero pueden ser agentes MKU. El agente RKU que inicia el protocolo DCN, ejecuta un protocolo CN con el agente SP, el que a su vez inicia un protocolo CN con cada agente RKU que debe participar del proceso de colaboración. El agente cuyo rol es iniciar el protocolo DCN recibe el nombre de *RKU-Initiator* (*rkuInit*) y los agentes contactados por el agente SP, cuyo rol es responder, reciben el nombre de *RKU-Responder* (*rkuResp*). El agente SP desempeña ambos roles.

El protocolo DCN comienza cuando el agente *RKU-Initiator* envía un *cfp* a un agente SP. Cuando el agente SP (rol *Respondedor*) recibe el *cfp*, define los participantes y transforma el documento de negocio. Luego, el agente SP (rol *Iniciador*) inicia un protocolo CN con cada participante enviando un mensaje *cfp*. Cuando el agente *RKU-Responder* recibe el *cfp* puede responder con un *refuse* o un *propose*.

Si el agente SP recibe algún *refuse* o finaliza el tiempo de espera sin que haya recibido la respuesta de todos los participantes, envía un *refuse* al agente *RKU-Initiator* y un *reject-proposal* a cada agente *RKU-Responder* que envió un *propose*. El protocolo finaliza.

Si el agente SP recibe un *propose* de todos los *RKU-Responder* que participan, combina las propuestas con el objetivo de generar una respuesta para el agente *RKU-Initiator*.

Si el agente SP no consigue armar una propuesta envía un *refuse* al agente *RKU-Initiator* y un *reject-proposal* a cada agente *RKU-Responder* participante. El protocolo finaliza.

Si el agente SP consigue armar una propuesta envía un *proposal* al agente *RKU-Initiator*, el cual la evalúa y decide si envía al agente SP un *reject-proposal* o un *accept-proposal*.

Si el agente SP recibe del agente *RKU-Initiator* un *reject-proposal* envía un *reject-proposal* a cada uno de los agentes *RKU-Responder* participantes. El protocolo finaliza.

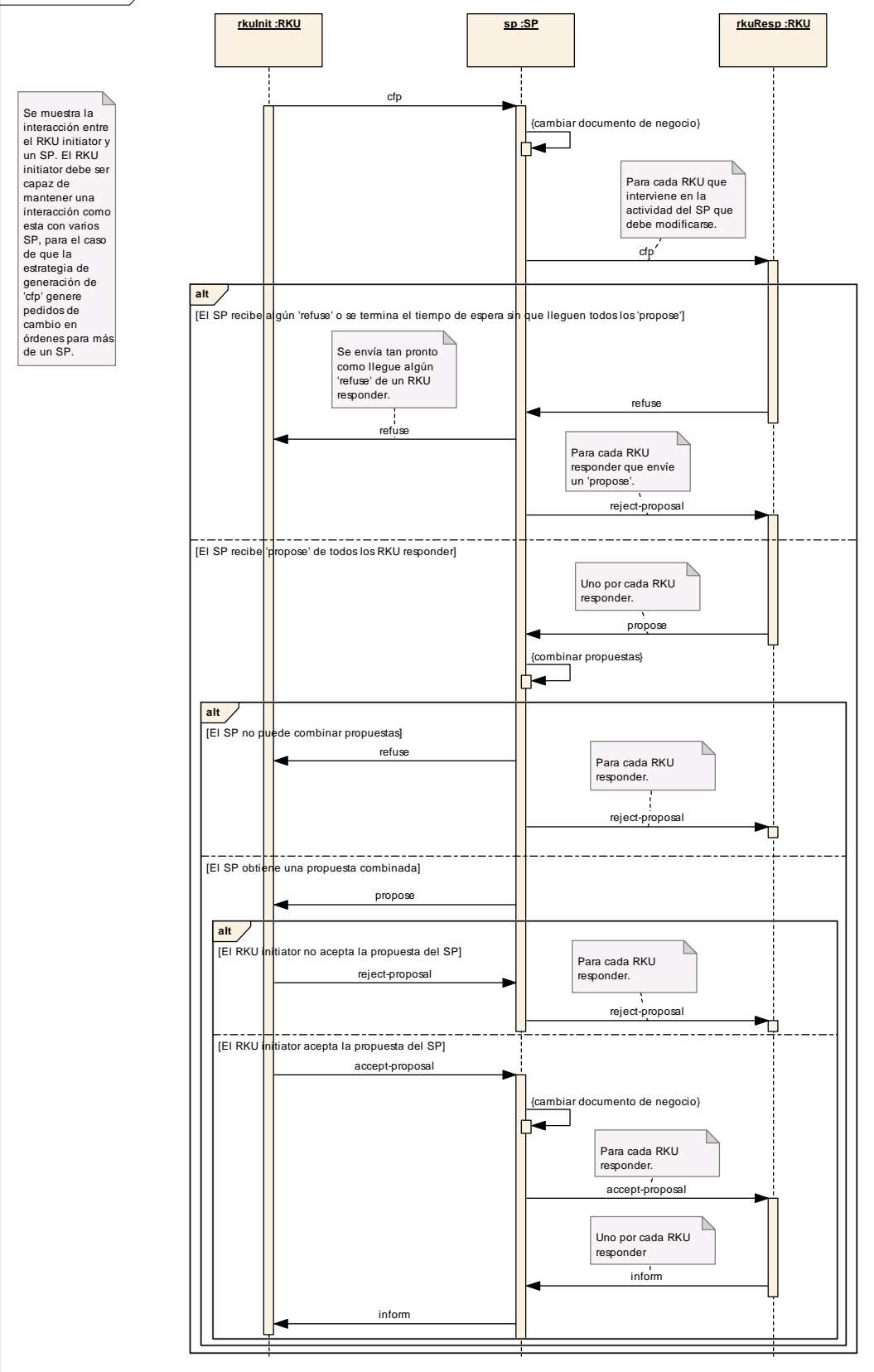


Figura 3.8: Diagrama de Secuencia para el protocolo DCN

Si el agente SP recibe un *accept-proposal*, transforma el documento de negocio con la propuesta aceptada y envía un *accept-proposal* a cada agente *RKUs-Responder* participante. Cuando los agentes *RKUs-Responder* reciben el *accept-proposal* deben confirmar al agente SP con un *inform* que pudieron realizar la acción correctamente. Recibidas todas estas confirmaciones, el agente SP envía el *inform* final al agente *RKU-Initiator*. El protocolo finaliza.

---

### 3.3.2. MODELO DE NEGOCIACIÓN

Un aspecto importante en la especificación de un proceso de coordinación tendiente a determinar una conducta emergente, es definir la información que los agentes deben intercambiar. Esta tarea ha sido inspirada en la Teoría de Mercado Computacional [Weiss, 1999; Mas, 2005; Wellman, 1994; 1995; 1996; Wurman, 1999a; 1999b]. Esta teoría define la existencia de dos tipos de agentes, los consumidores y los productores, y requiere especificar los bienes a ser intercambiados, las funciones de utilidad para los agentes consumidores y las funciones de tecnología para los agentes productores. También se debe especificar el comportamiento de los agentes en la generación de las distintas acciones dentro del marco de un proceso de colaboración.

Los *bienes de intercambio* son los elementos que serán intercambiados y producidos.

Los *agentes consumidores* son aquellos que intercambian los bienes. Son definidos por una asignación inicial de bienes y por su función de utilidad, la cual especifica su preferencia por consumir un conjunto de bienes.

Los *agentes productores* no consumen los bienes de intercambio sino que llevan a cabo un proceso de transformación de bienes de entrada en otros bienes de salida. Son definidos por su función de tecnología, la cual realiza la conversión o transformación.

Desde la perspectiva de esta teoría, en la agencia definida, las holguras son identificadas como los bienes de intercambio en un proceso de coordinación, que son utilizadas por los agentes para responder a los cambios, tratando de evitar la necesidad de una reprogramación. Por ello, las holguras son los elementos a negociar entre los agentes cuando buscan una solución a una excepción, siendo objetivo mantener su holgura en el mejor nivel posible.

Un agente RKU es identificado como consumidor. Consume y negocia bienes de intercambio pero no los transforma.

Un agente SP es identificado como productor. Establece los balances y conversiones de los bienes de intercambio entre agentes consumidores. No fija utilidades ni busca beneficio, solo lleva a cabo la conversión que él representa.

Los bienes de intercambio en la agencia propuesta son las holguras. Las mismas están implícitamente definidas en los programas de los agentes RKU o MKU. Estas holguras pueden ser programadas como una función de variables que el agente conoce y puede manipular. Estas variables dependen del tipo de agente consumidor.

Para el agente consumidor RKU la holgura se encuentra implícitamente definida en su agenda de uso, siendo los parámetros tiempo y capacidad los que permiten especificarla. De este modo un agente RKU puede intercambiar tiempos y/o capacidades proponiendo adelantamientos o atrasos en la fecha de inicio, extensiones o reducciones en la duración de los requerimientos, y/o incrementos o disminuciones en la capacidad solicitada del recurso por parte de las órdenes de su agenda de uso.

Para el caso de los agentes consumidores MKU la holgura se encuentra especificada en los parámetros de las órdenes de su lista de entrada/salida. De este modo un agente MKU puede intercambiar tiempos y/o cantidades proponiendo adelantamientos o atrasos en la fecha de inicio, extensiones o reducciones en la duración, y/o incrementos o disminuciones en las cantidades de las órdenes de su lista de entrada/salida.

Una vez identificados los bienes de intercambio, se pueden definir las funciones asociadas a cada tipo de agentes. Los agentes consumidores tienen asociada una función de utilidad que puede ser interpretada como la función que indica cuán importante es para el agente tener una parte del recurso que se intercambia. La función de utilidad de un agente RKU buscará mantener su nivel de servicio, siendo éste un índice de preferencia relativa. De modo que el valor de la holgura ofertada está asociado al nivel de servicio en el que quedaría el agente RKU oferente de aceptarse esa propuesta.

Los agentes productores SP tienen asociada una función tecnológica, la cual realiza la conversión o transformación de las holguras propuestas y se define a partir de su plan de actividades.



Ambas funciones deben cumplir una serie de requisitos de forma tal que la convergencia a un punto de equilibrio pueda ser esperada, en general se pide que estas funciones sean bien comportadas, es decir deben presentar características de monotonidad, convexidad y continuidad; además la función de utilidad debe verificar completitud y transitividad [Wellman, 1996; Wellman, 1994; Wurman, 1999; Wurman, 1999 and Wellman].

### 3.4. CONCLUSIONES

La arquitectura de la agencia fue definida como una red de puntos de control sobre inventarios a cargo de agentes MKU cuyas relaciones son modeladas con puntos de enlaces gestionados por agentes SP, los cuales para su funcionamiento emplean recursos cuyo control está a cargo de agentes RKU. Además de estos agentes que conforman el núcleo principal de la agencia se han definido agentes de servicio con funciones específicas de interfaz.

El protocolo *Double Contract Net* propuesto proporciona a la agencia un mecanismo de coordinación sustentado en el uso de agentes que actúan de mediadores entre el agente coordinador y los agentes participantes de la colaboración, lo cual facilita la escalabilidad de la agencia.

En el contexto del problema a resolver el proceso de coordinación principal se ha pensado como un mecanismo de reasignación de recursos [Bartschi, 1996]. Los agentes RKU o MKU cuentan con una asignación inicial de holguras (*buffers* de recursos). Cuando un evento ocurre deben usar las holguras para poder hacer frente a la variación producida. Es decir los agentes intercambiarán holguras con el fin de reestablecer el equilibrio. Desde esta perspectiva para plantear una solución al problema de la coordinación se ha tomado como base la propuesta de Mercado Computacional [Wellman, 1996] la cual ofrece un paradigma de control descentralizado y ha inspirado nuevas formas para implementar esquemas de asignación de recursos. La idea de aplicar mecanismos de mercado para resolver problemas distribuidos de coordinación no es nueva, pudiéndose encontrar la metáfora de mercado en la propuesta de *Contract-Net Protocol* [Wellman, 1994].

En la agencia propuesta la capacidad de llevar a cabo un comportamiento autónomo en la gestión de eventos disruptivos en la cadena de suministro para la

detección de una excepción y la posterior búsqueda e implementación de una solución a la misma surge de la interacción entre agentes que llevan a cabo comportamientos relativamente sencillos. Cada agente se especializa en realizar una tarea, pero ninguno de ellos tiene la visión global del problema que se está resolviendo ni conoce el objetivo global del sistema. A partir de la interacción coordinada entre pares de agentes RKU o MKU y SP surge el comportamiento tendiente a resolver una excepción. Además de esta interacción principal tienen lugar otras interacciones simples pero igualmente importantes relacionadas con los agentes de servicios.

El rol de coordinador es asumido por el agente RKU o MKU que modela el punto de control donde se generó el evento. Una vez que un agente detecta una excepción, necesita encontrar una solución llamando a colaborar a otros agentes para distribuir el desvío y de esta manera resolver la excepción. Dado que no conoce quienes son los agentes responsables de recursos relacionados debe interactuar con el agente SP correspondiente y éste es quien conoce los agentes responsables de estos recursos.

En un momento dado pueden coexistir varios coordinadores buscando hallar una solución localmente. Este tipo de propuesta de coordinación es denominada mediada multi-centrica, donde los agentes participantes no solo buscan su propio bienestar sino también el bien común. Además es necesario minimizar la cantidad de agentes participantes, por lo cual el proceso es planteado por niveles pudiéndose definir la cantidad de niveles a involucrar o bien limitar esto por tiempo en el cual una solución debe ser hallada (Sección 2.3.3.3 del Capítulo 2).

En una primera implementación de esta tesis solo se considerarán expansiones de primer nivel. Por lo cual los agentes involucrados en una colaboración serán: un agente RKU o MKU en el rol de coordinador, los agentes SP asociados y los agentes RKU y MKU en el rol de participantes.

## CAPÍTULO 4 – PROTOTIPO DE UN SISTEMA SCEM MULTI-AGENTE

En este capítulo se presenta un prototipo que implementa la agencia desarrollada en el capítulo anterior. El prototipo fue desarrollado con el objetivo de efectuar una validación inicial de las principales ideas y conceptos propuestos para el problema de automatizar el proceso de utilizar las holguras de las órdenes y los recursos para absorber los desvíos producidos por las interrupciones en un programa de abastecimiento que se encuentra en ejecución.

En la arquitectura propuesta (Sección 2.1, Capítulo 2) el sistema SCEM es un componente que interactúa con el Sistema de Programación y el Sistema de Ejecución. El alcance de verificación de esta tesis fue limitado al proceso de coordinación descrito en el Capítulo 2, por lo cual el prototipo fue implementado para simular la operatoria del componente SCEM con independencia de los otros componentes. Para ello se modificó la arquitectura del agente EVA incluyendo el rol de simular la ocurrencia de eventos durante la ejecución de un programa de abastecimiento.

Se describe la implementación de los conceptos previos en un prototipo detallando: la plataforma utilizada, la implementación de los agentes, la ontología que ellos utilizan y las interfaces gráficas desarrolladas.

### 4.1. PLATAFORMA DE DESARROLLO

El desarrollo de un sistema multi-agente requiere la implementación de características no soportadas por los lenguajes de programación convencionales, tales como transporte de mensajes, codificación, servicios de páginas blancas y amarillas, ontologías y gestión del ciclo de vida del agente. Por lo cual, el uso de plataformas de desarrollo de agentes permite reducir considerablemente el esfuerzo de programación. Las plataformas de agentes más populares están basadas en los estándares FIPA y Java, por ejemplo: JADE [Bellifemine et al, 1999], JACK [Busetta et al, 1999] [Howden et al, 2001] o FIPA-OS [FIPA-OS 2006]. Referencias a más herramientas de desarrollo de agentes pueden encontrarse en *Agent Builder* (<http://www.agentbuilder.com/AgentTools/>).

Para el desarrollo del prototipo se seleccionó JADE porque al momento que se tomó la decisión era la única plataforma que satisfacía los siguientes criterios: plataforma libre, con buena documentación y soporte disponible, uso de estándares y características que soportan la gestión de la comunidad de agentes. Dado que JADE está basado en los estándares de FIPA, las interacciones entre los agentes se basan en el intercambio de mensajes en el formato FIPA-ACL y estructurado de acuerdo a los protocolos de interacción FIPA.

JADE provee una plataforma *middleware* y un *framework* para el desarrollo y ejecución de agentes. Esto permite concentrarse en funciones específicas de la solución. La plataforma de agentes provee una GUI para la gestión remota de la plataforma, permitiendo monitorear y controlar el estado de los agentes.

## 4.2. IMPLEMENTACIÓN DE LOS AGENTES

Cada agente es una clase simple que extiende la clase *Agent* provista por JADE, heredando las funcionalidades básicas, tales como servicio de registro, gestión remota y enviar/recibir mensajes ACL [Bellifemine et al, 1999]. Estas funcionalidades fueron extendidas con características que representan el comportamiento específico de cada agente.

El comportamiento de cada agente usa programación multi-hilo permitiendo realizar múltiples acciones en paralelo. Cuando cada agente es creado las primeras acciones son su inicialización (leer los archivos de configuración) y el registro en la federación de acuerdo a la estructura organizacional. Después de esto los componentes del agente son iniciados. En las siguientes secciones se presentan detalles de la implementación de cada uno de los agentes, incluyendo sus principales clases y los comportamientos que implementan el proceso de coordinación descrito en el Capítulo 2.

---

### 4.2.1. AGENTE PAGE

El agente PAGE, implementado en el prototipo con el nombre de agente AP (*Agente de Planificación*), es el responsable de crear la instancia de la agencia. Esto implica crear cada uno de los agentes que conforman la agencia para un determinado programa de abastecimiento. La implementación actual permite realizar dicha tarea

mediante la configuración de los agentes en archivos XML. El agente AP (Figura 4.1) posee un constructor mediante el cual recibe dos rutas absolutas del sistema de archivos: la primera de ellas corresponde a la ubicación del archivo XML con la configuración de los agentes RKU y SP; y la segunda corresponde a la ubicación del archivo XML con los eventos.

En base a la información brindada por los archivos XML, el agente AP instancia los agentes RKU, SP y AE (*Agente de Eventos*). El agente AE es el agente EVA con el rol incorporado de simulación de eventos. Al crear la instancia del agente AE le proporciona la información de los eventos que dicho agente debe ir generando de acuerdo a lo configurado para el proceso de simulación. En la medida que dichos eventos van ocurriendo, afectan a diferentes agentes RKU de la agencia. Cada uno de los cuales deberá en su momento activar su mecanismo para identificar excepción primero y luego de coordinación para buscar una solución, o bien informar que el problema no puede ser resuelto según se describió en el Capítulo 2.

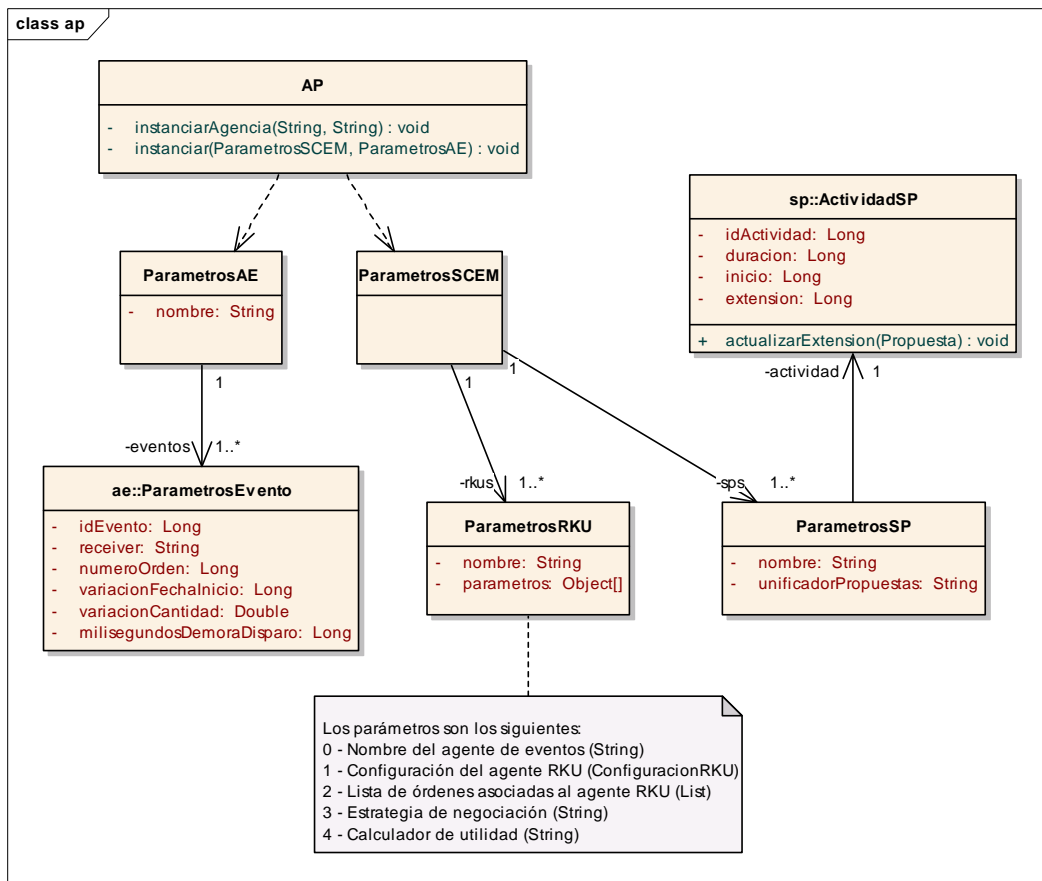


Figura 4.1: Diagrama de clases para el agente AP (PAGE)

Por cada ejemplo que se quiera simular en el prototipo se deberán escribir los dos archivos: el que contiene la configuración de cada uno de los agentes que conforman la agencia y el que contiene cada uno de los eventos a simular.

La especificación de los agentes se realiza mediante un archivo XML cuya estructura se define en el esquema “agentes.xsd” (Figura 4.2). Por esta razón todo archivo que contenga definiciones de agentes debe incluir las mismas en un *tag* “scem” como se muestra a continuación:

```
<scem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="agentes.xsd">
</scem>
```

Dentro del *tag* “scem” se deben incluir, por única vez, dos *tags*: “rkus” y “sps”. Dentro de “rkus” se definen los agentes RKU que conforman la agencia para el ejemplo simulado. Dentro de “sps”, se definen los agentes SP.

Cada agente RKU se define dentro un *tag* “rku”. Para cada agente se debe configurar la siguiente información:

- ❖ *nombre: nombre del agente RKU.*
- ❖ *ae:* agente de eventos que informará al agente RKU de los eventos ocurridos que lo afectan.
- ❖ *configuracion – valorMinimoVariable:* el valor mínimo que podría tomar la variable monitoreada por el agente RKU.
- ❖ *configuracion – valorMaximoVariable:* el valor máximo que podría tomar la variable monitoreada por el agente RKU.
- ❖ *configuracion – nivelServicioObjetivo:* el nivel de servicio objetivo del agente RKU.
- ❖ *ordenes:* configuración de las órdenes de entrada y salida del agente RKU.
- ❖ *estrategia:* la estrategia de negociación utilizada.
- ❖ *calculadorUtilidad:* el calculador de utilidad utilizado.

Por cada orden se debe configurar la siguiente información:

- ❖ *numeroOrden:* el número de la orden.
- ❖ *inicio:* el tiempo en el cual inicia la orden.
- ❖ *duracion:* la duración de la orden.

- ❖ *cantidad*: la cantidad afectada por la orden.
- ❖ *tipoCambioVariable*: la forma en que la orden afecta a la variable monitoreada.
- ❖ *nombreSp*: el nombre del agente SP que relaciona la orden con el agente RKU.

Cada agente SP se define dentro de un *tag* “sp”. Para cada agente se debe configurar la siguiente información:

- ❖ *nombre*: nombre del agente SP.
- ❖ *idActividad*: identificador de la actividad realizada por el agente SP.
- ❖ *inicio*: el tiempo en el cual comienza la actividad del agente SP.
- ❖ *duracion*: la duración de la actividad del agente SP.
- ❖ *extension*: la extensión utilizada en las órdenes del agente SP.
- ❖ *unificadorPropuestas*: el unificador de propuestas que utiliza el agente SP cuando recibe varias propuestas a una llamada a propuestas.
- ❖ *items*: los ítems que conforman el plan de actividades con el cual trabaja el agente SP.

Por cada ítem se debe configurar la siguiente información:

- ❖ *nombreRku*: el nombre del RKU al que afecta la orden involucrada.
- ❖ *numeroOrdenRKU*: el número de la orden asociada al ítem.
- ❖ *variacionCantidadRKU*: la variación que genera el ítem de la actividad en la cantidad del RKU involucrado.

A continuación se muestra como ejemplo un archivo XML reducido, con un agente RKU y un agente SP configurados.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<scem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="agentes.xsd">
<rkus>
<rkus>
<nombre>rkuInit</nombre>
<ae>ae</ae>
```

```
<configuracion>
<valorMinimoVariable>500</valorMinimoVariable>
<valorMaximoVariable>2000</valorMaximoVariable>
<nivelServicioObjetivo>1</nivelServicioObjetivo>
</configuracion>

<ordenes>

<orden>

<numeroOrden>2</numeroOrden>

<inicio>10</inicio>

<duracion>1</duracion>

<cantidad>-1000</cantidad>

<tipoCambioVariable>1</tipoCambioVariable>

<nombreSp>sp</nombreSp>

</orden>

</ordenes>

<estrategia>inicioDesvio</estrategia>

<calculadorUtilidad>default</calculadorUtilidad>

</rku>

</rkus>

<sps>

<sp>

<nombre>sp</nombre>

<idActividad>1</idActividad>

<inicio>10</inicio>

<duracion>2</duracion>

<extension>500</extension>

<unificadorPropuestas>default</unificadorPropuestas>
```



```

<items>
<item>
<nombreRku>rkuInit</nombreRku>
<numeroOrdenRku>2</numeroOrdenRku>
<variacionCantidadRku>-2</variacionCantidadRku>
</item>
<item>
<nombreRku>rkuResp1</nombreRku>
<numeroOrdenRku>1</numeroOrdenRku>
<variacionCantidadRku>1</variacionCantidadRku>
</item>
<item>
<nombreRku>rkuResp2</nombreRku>
<numeroOrdenRku>1</numeroOrdenRku>
<variacionCantidadRku>-2</variacionCantidadRku>
</item>
</items>
</sp>
</sps>
</scem>

```

**Figura 4.2: Ejemplo de un archivo “agentes.xsd”**

Este archivo XML es luego convertido en la estructura de objetos que se muestra en la Figura 4.1. Se crea un único objeto *ParametrosSCem*, que contiene dos listas: una lista de objetos *ParametrosRku* y otra lista de objetos *ParametrosSP*. Cada uno de los agentes Rku configurados en el archivo XML se traduce en un objeto *ParametrosRku*, y cada uno de los agentes SP se traduce en un objeto *ParametrosSP*. Finalmente esta información es utilizada por el agente AP para crear cada uno de los agentes Rku y SP que componen la agencia.

## 4.2.2. AGENTE EVA

Según se explicó en la sección previa, en el prototipo el agente EVA fue implementado como Agente de Eventos (AE). La estructura del agente y clases relacionadas puede observarse en la Figura 4.3.

La clase principal que representa al agente es AE. Extiende la clase *Agent* de JADE y por lo tanto es visto como un agente dentro del contenedor JADE.

El agente AE es el que dispara la ejecución de los eventos en los tiempos correspondientes según la configuración de eventos que es creada por el agente AP descrito en la sección previa.

La *fechaBase* que posee este agente es una fecha que sirve de referencia común para medir el tiempo que debe transcurrir hasta que se dispare cada uno de los eventos.

Cuando el agente es creado, recibe una lista de objetos *ParametrosEvento* con la descripción de cada uno de los eventos a disparar. A partir de esta lista de descripciones, el agente crea los objetos Evento e instancia para cada uno de ellos un *behaviour* que será el responsable de disparar la ejecución de dicho evento en la fecha y hora correspondientes.

Cada evento posee los siguientes datos:

- ❖ *idEvento*: identificador del evento.
- ❖ *numeroOrden*: el número de orden en el agente RKU que se modifica con el evento.
- ❖ *variacionFechaInicio*: la variación en la fecha de inicio de la orden.
- ❖ *variacionCantidad*: la variación en la cantidad de la orden.
- ❖ *fechaDisparo*: la fecha en que debe dispararse el evento. Se determina a partir de la *fechaBase* del agente AE y los *milisegundosDemoraDisparo* del objeto *ParametroEvento*. Este último indica la cantidad de milisegundos que deben transcurrir desde la *fechaBase* para el disparo del evento.
- ❖ *receiver*: el nombre del agente RKU que debe recibir el evento.

Un objeto *DisparadorEvento* es un *WakerBehaviour* de JADE. Este tipo de *behaviour* realiza una única ejecución transcurrido un *timeout* que es indicado cuando es

creado. En este caso, el *timeout* corresponde a la fecha de disparo (*fechaDisparo*) del evento. Es decir que el *behaviour* esperará que se produzca la fecha de disparo y en ese momento comunicará al agente RKU correspondiente el evento ocurrido.

La definición de los eventos se realiza mediante un archivo XML cuya estructura se encuentra definida en el esquema “eventos.xsd”. Por esta razón todo archivo que contenga definiciones de eventos debe incluirlas en un *tag* “ae” como se muestra a continuación:

```
<ae xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="eventos.xsd">
</ae>
```

Dentro del *tag* “ae” se deben incluir, por única vez, dos *tags*: “nombre” y “eventos”. El *tag* “nombre” sirve para definir el nombre del agente AE cuya instancia debe crearse. Dentro del *tag* “eventos”, se definen los eventos que el agente AE debe generar durante la simulación.

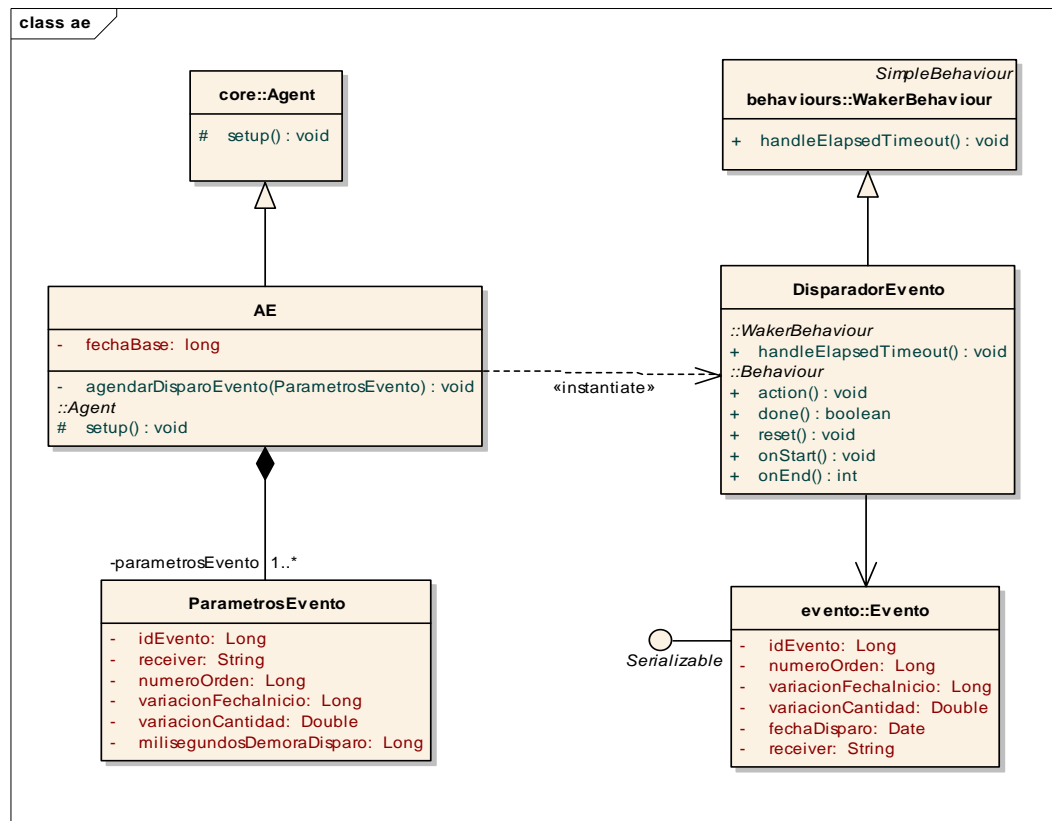


Figura 4.3: Diagrama de clases para el agente AE

En la Figura 4.4 se muestra un ejemplo de un archivo XML utilizado para la configuración de eventos de una simulación.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ae xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="eventos.xsd">
<nombre>ae</nombre>
<eventos>
<evento>
<idEvento>1</idEvento>
<receiver>rkuInit</receiver>
<numeroOrden>2</numeroOrden>
<variacionFechaInicio>0</variacionFechaInicio>
<variacionCantidad>-500</variacionCantidad>
<milisegundosDemoraDisparo>0</milisegundosDemoraDisparo>
</evento>
</eventos>
</ae>
```

**Figura 4.4: Ejemplo de archivo XML de configuración de eventos**

Este archivo XML es luego traducido a la estructura de objetos de la Figura 4.1. Se crea un único objeto *ParametrosAE*, que contiene una lista de objetos *ParametrosEvento*. La información contenida en *ParametrosAE* es utilizada por el agente AP para crear una instancia del agente AE con el nombre especificado en dicho objeto (Figura 4.3). Luego el agente AE recibe la lista de *ParametrosEvento*, y a partir de ella instancia los eventos que deberá disparar durante la simulación.

---

### 4.2.3. AGENTE RKU

Cuando un agente RKU identifica una excepción a partir de un evento enviado por el agente AE, debe llamar a colaborar, para ello iniciar un proceso de coordinación

(Sección 2.3.3.3 del Capítulo 2) el cual implica llevar a cabo una negociación para resolver la excepción.

La estructura del agente RKU y clases relacionadas puede observarse en la Figura 4.5. La clase principal que representa al agente es RKU. Extiende la clase *Agent* de JADE

A continuación se detallan las clases y *behaviours* de JADE, que son utilizadas por el agente RKU para realizar las negociaciones, las diferentes estrategias de negociación que puede utilizar y la forma que dispone para realizar el cálculo de utilidades.

El agente RKU delega la negociación en la clase, *CoordinadorNegociacion* (Figura 4.5) que es la responsable de determinar los cambios a realizar en las órdenes para que el plan de estados del agente RKU sea consistente con su configuración, y de iniciar la negociación con los agentes SP correspondientes.

El agente RKU puede realizar dichas negociaciones mediante diferentes estrategias. Por esa razón, cada agente RKU debe tener configurada una *EstrategiaNegociacion*. En el prototipo se implementaron estrategias simples y compuestas. Estas son:

*EstrategiaNegociacionInicioDesvio*: el agente RKU inicia la negociación solicitando un cambio en la cantidad de la orden correspondiente al tiempo donde se produce el primer desvío que da origen a la excepción. Sería una orden afectada indirectamente por un evento que afectó un recurso. La estrategia corresponde a una solución básica extrema.

*EstrategiaNegociacionOrdenEvento*: el agente RKU inicia la negociación solicitando un cambio en la cantidad de la orden afectada directamente por el evento. La estrategia corresponde a una solución básica extrema.

*EstrategiaNegociacionProponeImposibles*: el agente RKU inicia la negociación solicitando un cambio con valores absurdos. El objetivo de esta estrategia es que los agentes RKU respondan rechazando el pedido. Se utiliza para pruebas de la agencia.

*EstrategiaNegociacionCompuesta*: el agente RKU lleva a cabo la negociación utilizando una lista de sub-estrategias simples. Al momento de iniciar una negociación esta estrategia delega en la primera sub-estrategia de la lista la responsabilidad de

generar los pedidos de propuestas y gestionar las respuestas. En caso que la negociación no tenga éxito intentará con la siguiente estrategia de la lista y así sucesivamente hasta el final de la lista, de ser necesario. Podría ocurrir que ninguna sub-estrategia genere una negociación exitosa.

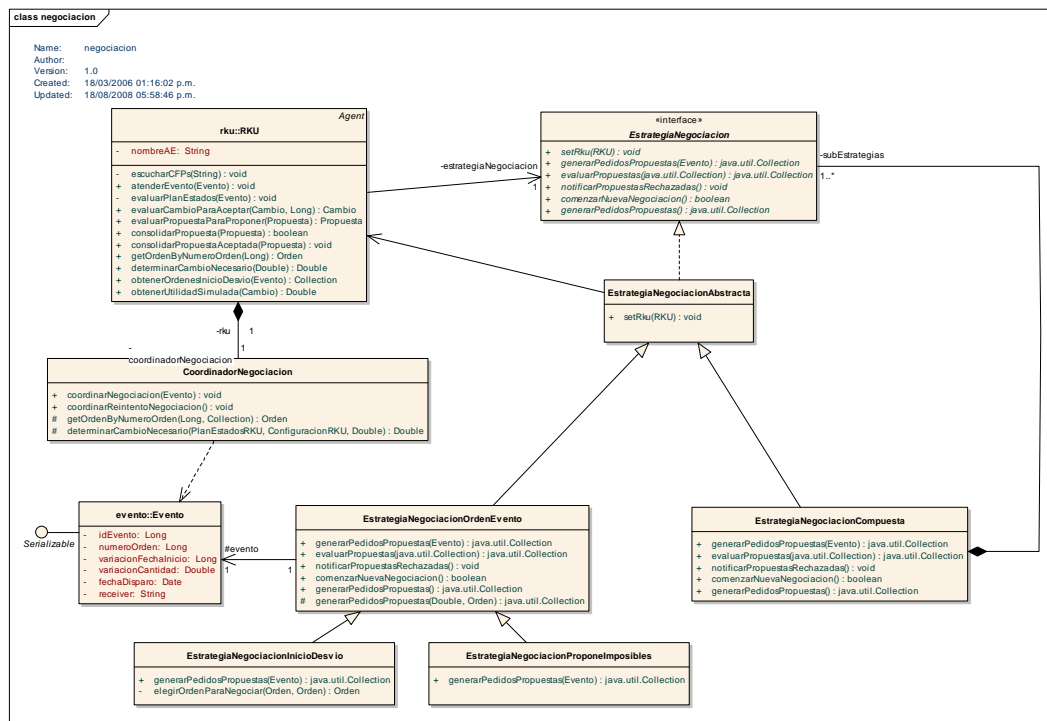


Figura 4.5: Estrategias de negociación

Las estrategias de negociación se definen mediante un archivo XML denominado “estrategias.xml” (Figura 4.6). En este archivo, el tag principal es “estrategias”, dentro del cual se definen las estrategias de negociación que podrá utilizar el agente RKU.

En este tag se define cada estrategia, mediante otros dos tags, según el tipo de estrategia. Para las estrategias simples se utiliza el tag “estrategia-simple” y para las estrategias compuestas el tag “estrategia-compuesta”.

Para las estrategias simples es necesario indicar el nombre de las mismas, mediante el atributo “nombre” y la clase Java que la implementa mediante el atributo “class”.

Para las estrategias compuestas es necesario indicar el nombre de las mismas, mediante el atributo “nombre”. Luego, englobadas bajo el tag “sub-estrategias”, se

enumeran las estrategias que forman parte de la estrategia compuesta mediante el *tag* “sub-estrategia”. Para cada sub-estrategia se especifica su nombre, que debe coincidir con el definido en el *tag* “nombre” de otra estrategia.

En la Figura 4.6 se muestra un extracto del archivo “estrategias.xml”. En el mismo se definen tres estrategias simples: “proponerImposibles”, “ordenEvento” e “inicioDesvio” y dos estrategias compuestas: “compuesta” y “superCompuesta”. La estrategia “compuesta” contiene dos sub-estrategias simples: “proponerImposibles” y “ordenEvento”. En cambio la estrategia “superCompuesta” contiene una estrategia simple, “inicioDesvio” y una estrategia compuesta, “compuesta”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<estrategias xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="estrategias.xsd">
  <estrategia-simple nombre="proponerImposibles"
    class="scems.agencia.agente.rku.negociacion.estrategia.EstrategiaNegociacionPr
  oponerImposibles"/>
  <estrategia-simple nombre="ordenEvento"
    class="scems.agencia.agente.rku.negociacion.estrategia.EstrategiaNegociacionOr
  denEvento"/>
  <estrategia-simple nombre="inicioDesvio"
    class="scems.agencia.agente.rku.negociacion.estrategia.EstrategiaNegociacionIni
  cioDesvio"/>
  <estrategia-compuesta nombre="compuesta">
    <sub-estrategias>
      <sub-estrategia>proponerImposibles</sub-estrategia>
      <sub-estrategia>ordenEvento</sub-estrategia>
    </sub-estrategias>
  </estrategia-compuesta>
  <estrategia-compuesta nombre="superCompuesta">
    <sub-estrategias>
      <sub-estrategia>compuesta</sub-estrategia>
```

<sub-estrategia>inicioDesvio</sub-estrategia>

</sub-estrategias>

</estrategia-compuesta>

</estrategias>

Figura 4.6: Extracto de un archivo “estrategias.xml”

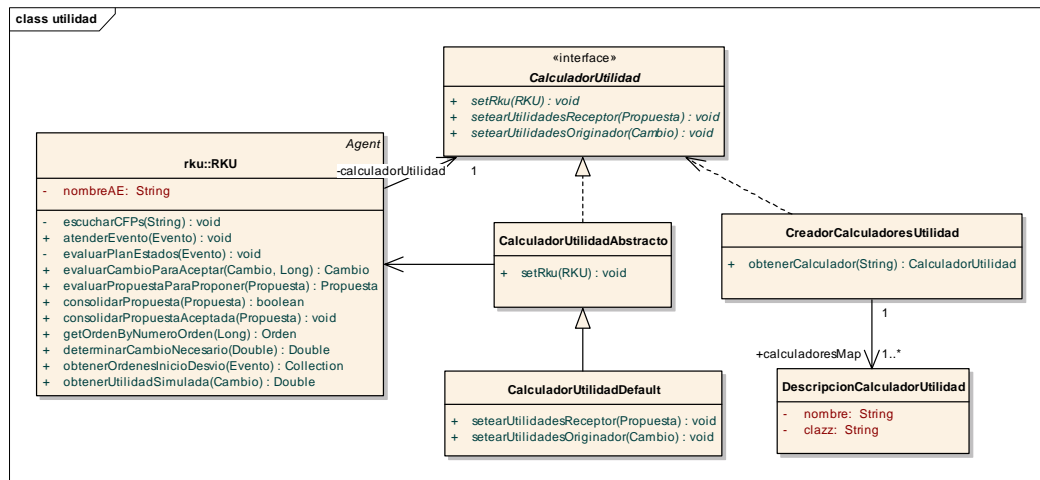


Figura 4.7: Diagrama de clases para calculadores de utilidades

Todo agente RKU debe disponer también de un *CalculadorUtilidad* (Figura 4.7), que es el responsable de realizar los cálculos de utilidades. Por el momento el prototipo dispone de una implementación, que establece todas las utilidades con el mismo valor, obtenido del *belief* simulado del agente RKU.

Los calculadores de utilidad se definen mediante un archivo XML denominado “calculadoresUtilidad.xml” (Figura 4.8). En este archivo, el *tag* principal es “calculadores-utilidad”, dentro del cual se definen todos los calculadores de utilidad disponibles en el sistema. En el mismo se especifica el nombre del calculador mediante el atributo “nombre” y la clase *Java* que lo implementa mediante el atributo “class”.

<?xml version="1.0" encoding="ISO-8859-1"?>

<calculadores-utilidad xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="calculadoresUtilidad.xsd">



```
<calculador-utilidadnombre="default"
  class="scems.agencia.agente.rku.negociacion.calculoUtilidad.CalculadorUtilidad
  Default"/>
</calculadores-utilidad>
```

**Figura 4.8: Extracto de un archivo “calculadoresUtilidad.xml”**

En el contexto de la tecnología de agentes los *behaviours* determinan qué es lo que el agente puede hacer, en la Figura 4.9 se representan los comportamientos que un agente RKU puede desempeñar.

La clase *RKUAEMessageReceiver* tiene la responsabilidad de ser el *behaviour* que implementa el comportamiento necesario para manejar los mensajes provenientes del agente AE. Extiende la clase *CyclicBehaviour* de JADE.

La clase *RKUContractNetManager* tiene la responsabilidad de ser el *behaviour* que maneja varias interacciones del protocolo FIPA *Contract Net* correspondientes a una negociación. Extiende la clase *SimpleBehaviour* de JADE.

La clase *RKUContractNetInitiator* tiene la responsabilidad de ser el *behaviour* que implementa el comportamiento necesario para que el agente RKU pueda participar en el rol de iniciador de una negociación utilizando el protocolo de interacción FIPA *Contract Net*. Extiende de la clase *ContractNetInitiator* de JADE.

La clase *RKUContractNetResponder* tiene la responsabilidad de ser el *behaviour* que implementa el comportamiento necesario para que el agente RKU pueda participar en el rol de responder de una negociación utilizando el protocolo de interacción FIPA *Contract Net*. Extiende la clase *ContractNetResponder* de JADE. La clase *RKUHandleAllResponses* tiene la responsabilidad de ser el *behaviour* que maneja todas las respuestas obtenidas por la interacción.

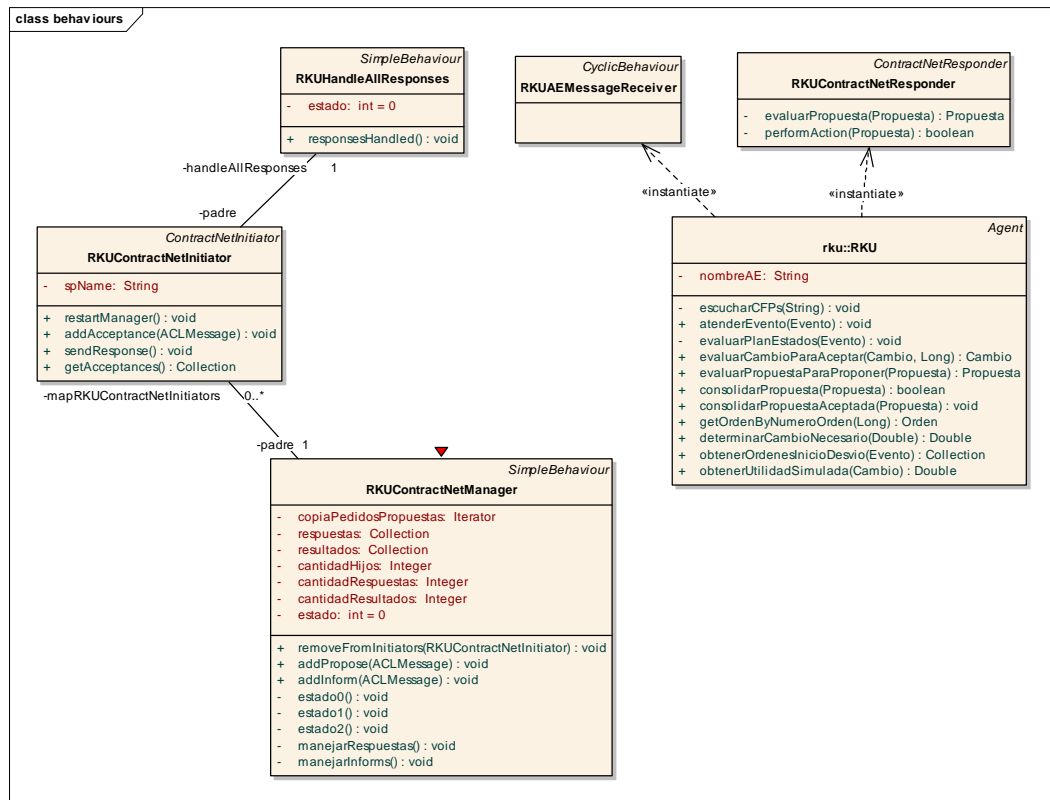


Figura 4.9: Behaviours del agente RKU

#### 4.2.4. AGENTE SP

En esta sección se detallan las clases y *behaviours* de JADE, que son utilizadas por un agente SP para mediar en el proceso de negociación, como así también las clases de utilidad para transformar y unificar propuestas.

Un agente SP tiene asociado un transformador de propuestas, que es una instancia de la clase *TransformadorPropuestas*. Su función es básicamente la transformación de propuestas, convirtiéndose de esta forma en la clase fundamental para la implementación del protocolo *Double Contract Net* (Sección 3.3.1 del Capítulo 3).

Cuando el agente SP recibe una propuesta de un agente *RKU iniciador*, invoca a un método del transformador de propuestas, que le permite convertir esa propuesta recibida en varias propuestas, una para cada agente *RKU responder* involucrado en la negociación. De esta forma, el agente SP puede comenzar las interacciones con cada uno de ellos.

Cuando recibe las respuestas de cada uno de los agentes *RKU responder*, el agente SP invoca a otro método del transformador de propuestas para convertir ese conjunto de respuestas en una respuesta única para el agente *RKU iniciador*.

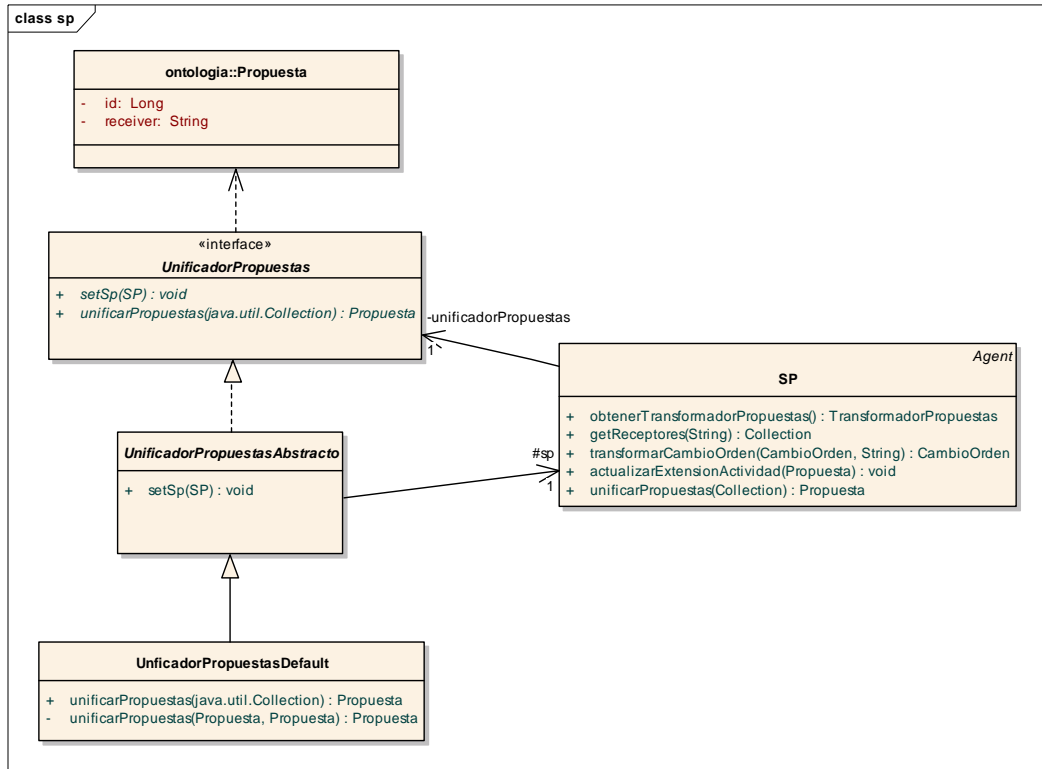
El agente SP contiene un unificador de propuestas, que es el responsable de convertir en única propuesta a todas aquellas respuestas recibidas de los agentes *RKU* participantes. La transformación del conjunto de propuestas a una única propuesta para el agente *RKU* iniciador es compleja y puede ser implementada de muchas formas. Por el momento el prototipo posee una única implementación básica del unificador de propuestas. En este caso se implementa asumiendo que todas las propuestas a unificar tendrán la misma cantidad de objetos *Cambio*, cada *Cambio* tendrá la misma cantidad de objetos *CambioOrden* y cada *CambioOrden* tendrá los mismos valores en la listas de tiempos y cantidades.

Los unificadores de propuesta (Figura 4.7) se definen mediante un archivo XML denominado *unificadoresPropuesta.xml* (Figura 4.6). En este archivo, el *tag* principal es “unificadores-propuestas”, dentro del cual se definen todos los unificadores de propuestas disponibles en el sistema.

Dentro de este *tag*, se define cada unificador dentro de un *tag* “unificador-propuestas”. En el mismo se especifica el nombre del unificador mediante el atributo “nombre” y la clase *Java* que lo implementa mediante el atributo “class”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<unificadores-propuestas xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="unificadoresPropuestas.xsd">
<unificador-propuestasnombre="default"
class="scems.agencia.agente.sp.negociacion.unificacionPropuestas.UnificadorPro
puestasDefault"/>
</unificadores-propuestas>
```

**Figura 4.10: Extracto de un archivo “UnificadoresPropuestas.xml”**



**Figura 4.11: Diagrama de clases para los *Unificadores de propuestas***

La clase *SPContractNetInitiator* (Figura 4.12) tiene la responsabilidad de ser el *behaviour* que implementa el comportamiento necesario para que el agente SP pueda participar en el rol de iniciador de una negociación utilizando el protocolo de interacción FIPA *Contract Net*. Extiende la clase *ContractNetInitiator* de JADE.

La clase *SPContractNetResponder* tiene la responsabilidad de ser el *behaviour* que implementa el comportamiento necesario para que el agente SP pueda participar en el rol de responder de una negociación utilizando el protocolo de interacción FIPA *Contract Net*. Extiende la clase *ContractNetResponder* de JADE.

La clase *SPHandleAllResponses* tiene la responsabilidad de ser el *behaviour* que gestiona todas las respuestas obtenidas por la interacción. Extiende la clase *SimpleBehaviour* de JADE.

La clase *SPPrepareResponse* tiene la responsabilidad de ser el *behaviour* que implementa el comportamiento necesario para que el agente SP pueda iniciar interacciones *Contract Net* con todos los agentes RCU participantes y luego combinar



Es necesario que cada agente pueda tener una representación interna de la información que sea fácil de gestionar, la misma se podría representar mediante objetos *Java*. Por lo cual la representación de la información contenida en un mensaje ACL debe ser convertida.

JADE está diseñado para realizar automáticamente la conversión y los chequeos semánticos entre el contenido del mensaje ACL y objetos *Java* utilizados para representar internamente la información por parte de los agentes.

Los principales elementos son:

- ❖ *Content manager*: provee los métodos para transformar objetos *Java* en *strings* o secuencias de *bytes* e insertarlos en el campo contenido de mensajes ACL y viceversa. Es accesible en un agente mediante el método *Agent.getContentManager()*. En realidad delega la conversión y las operaciones de chequeo a una *Ontología* y a un *Codec* de lenguaje de contenido.
- ❖ *Ontología*: valida la información a ser convertida desde el punto de vista semántico.
- ❖ *Codec*: realiza la transformación a *strings* o secuencias de *bytes* de acuerdo a las reglas sintácticas del lenguaje de contenido relacionado.

Las principales características en JADE son:

- ❖ *Definición de una ontología*: Una ontología en JADE es una instancia de la clase *Ontology*, a la cual le fueron agregados los esquemas, definiendo la estructura de tipos de predicados, acciones de agente y conceptos relevantes para el dominio.
- ❖ *Desarrollo de las clases Java ontológicas*: Cada esquema de la ontología tiene asociado una clase *Java*. La estructura de estas clases debe ser coherente con los esquemas asociados.
- ❖ *Selección del lenguaje de contenido*: JADE incluye *Codecs* para dos lenguajes de contenido: *Semantic Language* y *Lightweight Extensible Agent Platform*. Ambos soportan el modelo de referencia de contenido. Un

*Codec* para un lenguaje de contenido es un objeto *Java* capaz de manejar expresiones de contenido escritas en ese lenguaje.

Semantic Language es un lenguaje entendible para los humanos. Incluye un conjunto de operadores útiles, como los operadores lógicos *AND*, *OR* y *NOT* y los operadores modales *BELIEF*, *INTENTION* y *UNCERTAINLY*.

*Lightweight Extensible Agent Platform* es codificado como una secuencia de *bytes*. No es estándar. Sin embargo tiene algunas ventajas, como ser que el *Codec* es más liviano requiriendo menos memoria y soporta secuencias de *bytes*.

- ❖ *Registro del lenguaje de contenido y de la ontología al agente:* Antes de que un agente pueda usar una ontología y un lenguaje de contenido, éstos deben ser registrados al mismo. Esto se realiza con el método *setup* del agente.
- ❖ *Creación y manipulación de expresiones de contenido como objetos Java:* Una vez que se definió una ontología, se seleccionó el lenguaje de contenido adecuado y se registraron ambos en el agente, es muy fácil crear y manipular expresiones de contenido como objetos *Java*.

---

#### 4.3.2. CREACIÓN DE LAS ONTOLOGÍAS EN JADE

El desarrollo “manual” de la clase *Java* para la definición de una ontología (donde se definen los esquemas y términos a utilizar) y de las demás clases que representan los conceptos de dicha ontología puede demandar mucho tiempo.

Para el desarrollo de la ontología del sistema SCEM prototipo, se utilizó el *plugin beangenerator* (<http://www.swi.psy.uva.nl/usr/aart/beangenerator>) de Protégé, el cual es un editor gráfico de ontologías de uso libre y de código abierto. Soporta la creación, visualización y manipulación de ontologías en varios formatos.

Permitió desarrollar en modo gráfico la ontología y luego automáticamente generar la clase de definición de la ontología y las clases que representan los conceptos como objetos *Java*.

### 4.3.3. DOCUMENTO DE NEGOCIO DE SCEMS

El proceso de coordinación descrito en la Sección 2.3.3.3 del Capítulo 2 implica un proceso de negociación entre agentes. En la agencia propuesta, la negociación entre agentes se realiza mediante el protocolo *Double Contract Net* definido en la Sección 3.3.1 del Capítulo 3, cuyos mensajes tienen anexo un documento de negocio con la información intercambiada entre los agentes. A medida que la negociación avanza cada agente RKU va completando las distintas partes del documento de negocio y cada agente SP realiza conversiones y unifica los documentos de negocio provenientes de distintos agentes RKU utilizando el plan de actividades.

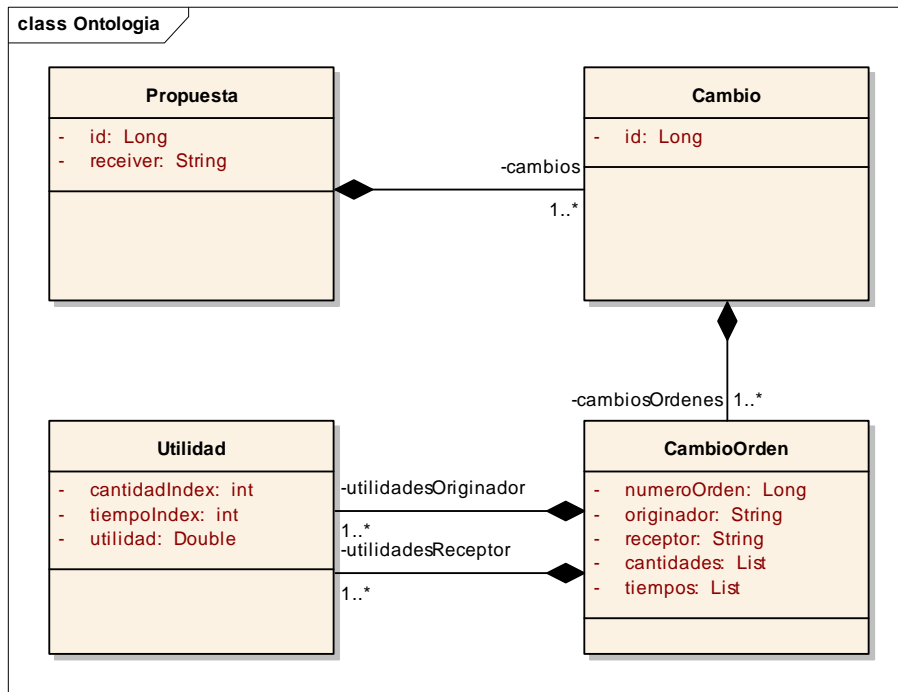
En la Figura 4.13 se detalla la estructura del documento de negocio, las clases utilizadas para cada porción de información intercambiada y la semántica de los atributos. En dicha figura se observa que una propuesta representada con la clase *Propuesta* está compuesta por una lista de cambios, cada uno representado por la clase *Cambio* y que cada cambio está compuesto de una lista de cambios para una orden específica, cada uno representado por la clase *CambioOrden*. Esto es: una misma propuesta puede contener varios cambios, donde cada uno de estos es independiente de los demás. La negociación puede concluir en la concreción de alguno o algunos de estos cambios. La lista de objetos *CambioOrden* representa los distintos cambios que deben consolidarse sobre órdenes específicas. Para que un cambio sea concretado todos los cambios de orden que lo componen deben ser consolidados.

Un objeto *CambioOrden* hace referencia a una orden en particular identificada mediante al atributo *numeroOrden*. Además del número de orden, estos objetos tienen los siguientes atributos:

- ❖ *originador*: Nombre del agente que origina el *CambioOrden*.
- ❖ *receptor*: Nombre del agente receptor del *CambioOrden*.
- ❖ *cantidades*: Lista de variación de cantidades del *CambioOrden*.
- ❖ *tiempos*: Lista de variación de tiempos del *CambioOrden*.
- ❖ *utilidadesOriginador*: Lista de objetos *Utilidad*, que representa la matriz de utilidades correspondientes a las variaciones de cantidades y tiempos del agente que origina el *CambioOrden*.



- ❖ *utilidadesReceptor*: Lista de objetos *Utilidad*, que representa la matriz de utilidades correspondientes a las variaciones de cantidades y tiempos del agente receptor del *CambioOrden*.



**Figura 4.13: Clases que modelan los conceptos utilizados en la ontología de SCEMS**

Cada objeto *CambioOrden* contiene rangos dentro de los que puede variar la cantidad de la orden o el tiempo de la misma. Las listas de variaciones de cantidades y tiempos modelan esos rangos. El producto cartesiano de estas dos listas determina una matriz. Cada elemento de esa matriz representado por la 2-tupla [variación de cantidad, variación de tiempo] debe tener una utilidad asociada, tanto para el agente que origina el *CambioOrden* como para el agente receptor. La Tabla 4.1 ilustra esta estructura.

Cada objeto *Utilidad* contiene un valor numérico, que expresa el valor de la función de utilidad del *originador* y del *receptor* para la correspondiente variación de cantidad y de tiempo (este es el valor absoluto de la utilidad, no una variación de utilidad). Ese valor es completado por cada agente y está determinado por la función de utilidad del mismo.

Si bien las propuestas de variaciones tanto en cantidad como en tiempo están definidas como conjuntos de valores discretos  $\{\Delta Q_1, \Delta Q_2, \dots, \Delta Q_n\}$  y  $\{\Delta t_1, \Delta t_2, \dots, \Delta t_n\}$  no implica que sólo esos valores pueden ser considerados durante la negociación,

sino que cualquier valor de variación de cantidad dentro del rango  $[\Delta Q_1; \Delta Q_n]$  y cualquier valor de variación de tiempo dentro del rango  $[\Delta t_1; \Delta t_n]$  puede ser acordado durante la negociación. Para ello se parte del supuesto que la función de utilidad varía linealmente para cualquier valor de  $\Delta Q$  dentro del sub-rango  $[\Delta Q_i; \Delta Q_{i+1}]$  y para cualquier valor de  $\Delta t$  dentro del sub-intervalo  $[\Delta t_j; \Delta t_{j+1}]$ , así el valor de la utilidad para cada punto contenido en cada región puede ser calculado por interpolación lineal.

	$\Delta Q_1$	$\Delta Q_2$	...	$\Delta Q_n$
$\Delta t_1$	Ut.Orig[1,1]	Ut.Orig[1,2]		Ut.Orig[1,n]
	Ut.Rec[1,1]	Ut.Rec[1,2]		Ut.Rec[1,n]
$\Delta t_2$	Ut.Orig[2,1]	Ut.Orig[2,2]		Ut.Orig[2,n]
	Ut.Rec[2,1]	Ut.Rec[2,2]		Ut.Rec[2,n]
...				
$\Delta t_n$	Ut.Orig[m,1]	Ut.Orig[m,2]		Ut.Orig[m,n]
	Ut.Rec[m,1]	Ut.Rec[m,2]		Ut.Rec[m,n]

**Tabla 4-1: Variaciones en cantidad y tiempo para una orden y sus utilidades**

## 4.6. CONCLUSIONES

La herramienta de software descrita en este capítulo es un prototipo de sistema SCEM que permite realizar simulaciones sobre escenarios configurables utilizando diferentes estrategias de búsqueda de solución ante cada evento disruptivo. Esto lo convierte en una herramienta de análisis en el ámbito de investigación, para evaluar el desempeño de diferentes algoritmos de búsqueda de solución.

Durante los procesos de negociación los agentes intercambian un documento de negocio que contiene la información que cada punto de control debe analizar al tomar decisiones. La definición de la estructura de este documento de negocio también formó parte de este trabajo, creando una ontología que provee una semántica común entre los agentes para que todos puedan interpretar la información de la misma manera.

Al ejecutar un escenario, la evolución del estado de cada agente a lo largo de las etapas de la negociación es almacenada en formato XML. Luego este XML permite generar reportes que ayudan a analizar la eficacia y eficiencia de las soluciones implementadas.

A partir de lo desarrollado, surge un abanico de mejoras y extensiones a implementar para ampliar las funcionalidades existentes. Por ejemplo, desarrollar e implementar estrategias de negociación más complejas, generar mecanismos de persistencia del estado del agente, incorporar información histórica para mejorar los procesos de toma de decisión de los agentes, mejorar las pantallas de interacción con los usuarios, e incorporar nuevos algoritmos para la toma de decisión de los agentes.

Finalmente es de destacar que si bien a nivel de modelo teórico conceptual se diferenciaron las entidades RKU y su especialización MKU, en el prototipo se implementó como un solo tipo de agente RKU donde la diferencia se establece a nivel de instanciación.

También es de destacar que la interfaz IOA no fue descrita debido a que no presenta detalles innovativos relevantes.

## CAPÍTULO 5 –CASO DE ESTUDIO

En este capítulo se presenta una validación experimental preliminar del sistema SCEM que implementa el modelo de gestión de eventos disruptivos en una cadena de suministros propuesto en esta tesis, utilizando el prototipo desarrollado descrito en el capítulo previo. Para la validación experimental, se ha estudiado el caso de la logística de distribución de productos a través de una cadena de suministro definida por un grupo de empresas.

### 5.1. METODOLOGÍA

Como validación preliminar se ha realizado un análisis del sistema propuesto a través de un caso de estudio en diferentes escenarios. Un caso de estudio ofrece la oportunidad de estudiar un fenómeno en su contexto, en el cual relaciones complejas y significativas subyacentes pueden ser exploradas; además permite estudiar toda la cadena [Miles, 1994] [Yin, 1994]. El caso de estudio también es apropiado donde el conocimiento es limitado [Oke, 2009].

### 5.2. CASO DE ESTUDIO

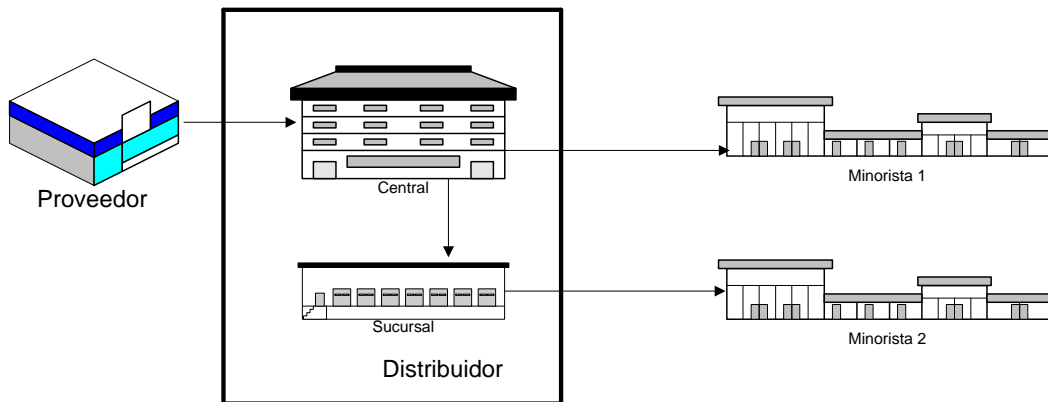
El caso desarrollado a continuación aborda el problema de la logística de distribución de productos a través de una cadena de suministro definida por cuatro empresas principales identificadas como: Proveedor, Distribuidor (en el que se identifica una casa central y una sucursal) y dos Minoristas. Para satisfacer los requerimientos de un mercado competitivo y cambiante estas cuatro empresas han decidido establecer una relación de colaboración. La colaboración ha sido acordada para gestionar dos productos principales identificados como P1 y P2 respectivamente. El producto P1 se comercializa en dos envases identificados como envase 1 y envase 2, que llevan a diferenciar al producto P1 como P11 (producto P1 en el envase 1) y P12 (producto P1 en el envase 2); en tanto que el producto P2 se comercializa en un único envase.

Además de estas empresas, la cadena de suministro está integrada por otros clientes minoristas que si bien adquieren los productos bajo análisis, no participan del

acuerdo de colaboración, por lo cual no son identificados. Las órdenes suministradas a éstos no tendrán identificado su destino. Del mismo modo los proveedores de segunda línea, que proveen los productos referidos a la empresa identificada como Proveedor en la cadena de suministro, dado que tampoco participan del acuerdo de colaboración no serán identificados. Las órdenes suministradas por éstos no tendrán identificado su origen.

Las órdenes de suministro sin origen o destino identificado no pueden estar sujetas a negociación, por esta razón cualquier cambio que en ellas se produzca generará un evento de frontera (Sección 2.3.3.2, Capítulo 2).

En la Figura 5.1 se muestra la estructura de la cadena de suministro formada por las cuatro empresas que participan del acuerdo de colaboración.



**Figura 5.1: Cadena de suministro del caso de estudio**

Las empresas que participan de esta cadena de suministro presentan restricciones en sus capacidades de almacenamiento y altos costos de almacenamiento. Esto las ha impulsado a desarrollar un proceso de planificación en colaboración a efectos de sincronizar sus programas de abastecimiento.

Debido a la incertidumbre de la demanda de los consumidores, la cual tiene lugar en los almacenes locales de los minoristas, las empresas han acordado una política de stock de seguridad en los distintos almacenes. Dicha política define niveles mínimos de inventario para cada producto en cada almacén.

Los programas de abastecimiento están definidos por órdenes que representan procesos de abastecimiento cuya función de transformación corresponde a un cambio de lugar,  $\Delta$ , del producto en cuestión (Sección 2.3.2, Capítulo 2). Es decir, se trata de un

proceso de transporte de productos entre dos almacenes, para ello el recurso no material limitante involucrado en cada caso es un camión. En la Tabla 5-1 se detallan los recursos para cada transformación. En todos los casos el proceso de transporte requiere entre una y tres horas, por lo cual el envío y la recepción ocurren en el mismo día.

<b>Origen</b>	<b>Destino</b>	<b>Tiempo Requerido [hs]</b>	<b>Recurso</b>
Proveedor	Central	2	Camion 1
Central	Sucursal	3	Camión 2
Central	Minorista 1	1	Camión 3
Sucursal	Minorista 2	1	Camión 4

**Tabla 5-1: Recurso y tiempo requerido por los procesos de transporte**

### 5.3. MODELADO DEL CASO DE ESTUDIO

En esta sección se detalla el programa de abastecimiento para un horizonte de tiempo de una semana, de cada una de las cuatro empresas que integran la cadena de suministro bajo estudio. Dichos programas fueron generados en base a un proceso de planificación en colaboración, de modo que están sincronizados.

En base al programa de abastecimiento cada empresa crea una instancia de la agencia. Los detalles de la misma se describen a continuación.

#### 5.3.1 MODELADO DEL PROGRAMA DE ABASTECIMIENTO DEL PROVEEDOR

El Proveedor es responsable de abastecer, dentro de la cadena de suministro, al Distribuidor, transportando los productos P11, P12 y P2 a la casa central del mismo. Para realizar esta tarea hace uso del recurso identificado como Camión1 (Tabla 5-1).

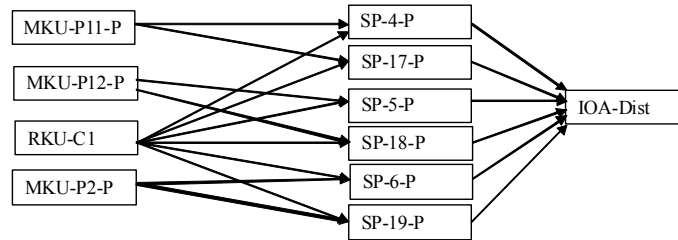
El programa de abastecimiento del Proveedor se presenta en la Tabla 5-2. El mismo es constituido por 23 órdenes, de las cuales 6 corresponden a abastecimientos al Distribuidor, 6 son órdenes con origen no identificado y 11 con destino no identificado.

<b>Id-Orden</b>	<b>Inicio</b>	<b>Origen</b>	<b>Destino</b>	<b>Producto</b>	<b>Cantidad</b>	<b>Recurso</b>
1	Día1	-	Proveedor	P11	600	-
2	Día1	-	Proveedor	P12	400	-
3	Día1	-	Proveedor	P2	600	-
4	Día1	Proveedor	Central	P11	120	Camión1
5	Día1	Proveedor	Central	P12	100	Camión1
6	Día1	Proveedor	Central	P2	210	Camión1
7	Día2	Proveedor	-	P11	150	-

8	Día2	Proveedor	-	P12	280	-
9	Día2	Proveedor	-	P2	160	-
10	Día2	Proveedor	-	P2	130	-
11	Día2	Proveedor	-	P11	160	-
12	Día3	Proveedor	-	P11	100	-
13	Día3	Proveedor	-	P12	100	-
14	Día4	-	Proveedor	P11	600	-
15	Día4	-	Proveedor	P12	400	-
16	Día4	-	Proveedor	P2	600	-
17	Día5	Proveedor	Central	P11	120	Camión1
18	Día5	Proveedor	Central	P12	100	Camión1
19	Día5	Proveedor	Central	P2	210	Camión1
20	Día6	Proveedor	-	P11	150	-
21	Día6	Proveedor	-	P12	280	-
22	Día6	Proveedor	-	P2	160	-
23	Día7	Proveedor	-	P2	130	-

**Tabla 5-2: Programa de abastecimiento del Proveedor**

En base al programa de abastecimiento de la Tabla 5-2, el agente PAGE de la agencia del Proveedor generará los agentes necesarios para representarlo junto con sus datos y planes asociados. En la Figura 5.3 se detalla la red de agentes resultante.



**Figura 5.2: Agencia que modela el programa de abastecimiento del Proveedor**

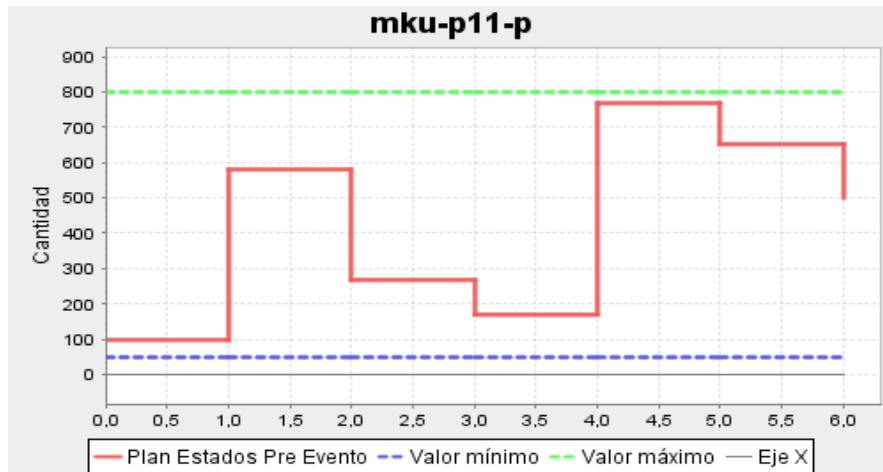
Los agentes MKU-y-P para  $y \in Y = \{P11, P12, P2\}$  representan los productos en el almacén del Proveedor involucrados en las órdenes de transporte; el agente RKU-C1 representa al recurso Camión 1 utilizado por el proceso de transporte; los agentes SP-x-P para  $x \in X = \{4, 5, 6, 17, 18, 19\}$  representan los procesos de transporte asociados a cada orden; y el agente IOA-Dist es el agente de interfaz con el Distribuidor.

En las Tablas 5-3 a 5-5 se muestran la lista de entrada/salida de cada uno de los agentes MKU-y-P para  $y \in Y = \{P11, P12, P2\}$  que el agente PAGE genera a partir del programa de abastecimiento del Proveedor (Tabla 5.2). En dichas tablas también se representa el stock de seguridad (valor mínimo del inventario), la capacidad máxima para dicho material, el nivel de servicio y el inventario inicial. En las Figuras 5.3 a 5.5 se representan gráficamente los planes de estado (perfiles de inventario) iniciales,

previos a la ocurrencia de eventos, que cada uno de los agentes MKU-y-P para  $y \in Y = \{P11, P12, P2\}$  genera a partir de su lista de entrada/salida. Dichas figuras corresponden a capturas de pantallas de reportes gráficos generados por el prototipo. Como se puede observar en dichas figuras, los perfiles de inventario iniciales de los tres productos en el almacén del Proveedor, correspondientes al programa de abastecimiento son factibles.

Valor Mínimo de la Variable: 50					
Valor Máximo de la Variable: 800					
Nivel de Servicio:1					
Inventario Inicial: 100					
Id-Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día1	0	+600	Principio	-
2	Día1	0	-120	Principio	SP-4-P
3	Día2	0	-150	Principio	-
4	Día2	0	-160	Principio	-
5	Día3	0	-100	Principio	-
6	Día4	0	+600	Principio	-
7	Día5	0	-120	Principio	Sp-17-P
8	Día6	0	-150	Principio	-

**Tabla 5-3: Lista de Entrada/Salida del Agente MKU-P11-P**



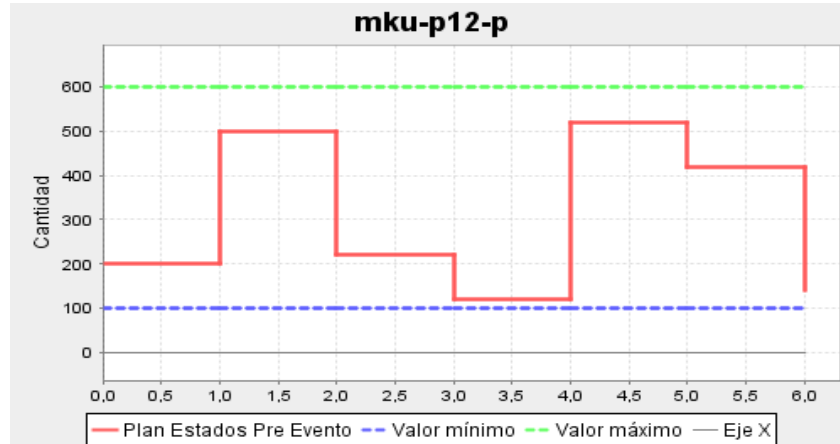
**Figura 5.3: Perfil de inventario del Agente MKU-P11-P**

Valor Mínimo de la Variable: 100					
Valor Máximo de la Variable: 600					
Nivel de Servicio: 1					
Inventario Inicial: 200					
Id-Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día1	0	+400	Principio	-
2	Día1	0	-100	Principio	SP-5-P
3	Día1	0	-280	Principio	-



4	Día4	0	-100	Principio	-
5	Día4	0	+400	Principio	-
6	Día5	0	-100	Principio	SP-18-P
7	Día6	0	-280	Principio	

**Tabla 5-4: Lista de Entrada/Salida del agente MKU-P12-P**



**Figura 5.4: Perfil de inventario del agente MKU-P12-P**

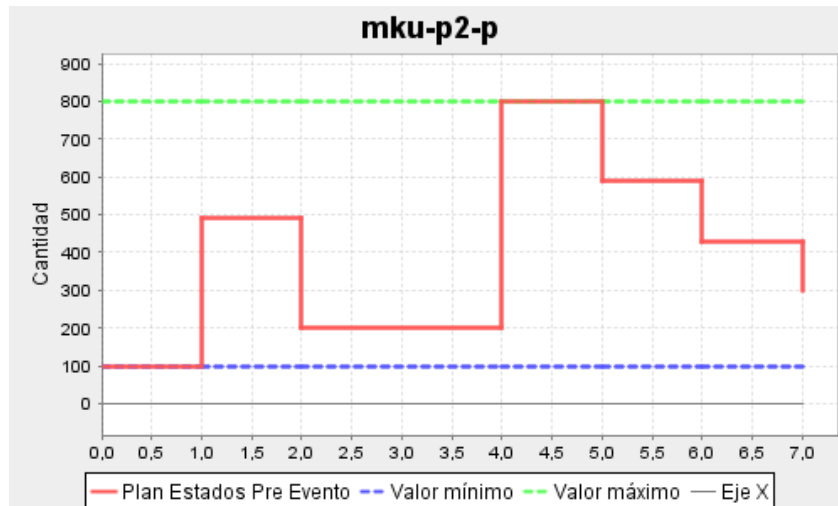
Valor Mínimo de la Variable: 100					
Valor Máximo de la Variable: 800					
Nivel de Servicio: 1					
Inventario Inicial: 100					
Id-Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día1	0	+400	Principio	-
2	Día1	0	-100	Principio	SP-6-P
3	Día2	0	-280	Principio	-
4	Día2	0	-100	Principio	-
5	Día4	0	+400	Principio	-
6	Día5	0	-100	Principio	SP-19-P
7	Día6	0	-280	Principio	-
8	Día7	0	-130	Principio	-

**Tabla 5-5: Lista de Entrada/Salida del agente MKU-P2-P**

Debido a que en las listas de entrada/salida hay órdenes con origen o destino no identificado, los agentes MKU-y-P para  $y \in Y = \{P11, P12, P2\}$  asumen el rol de agentes de frontera para gestionar los eventos de frontera que pueden ser generados por dichas órdenes.

En la Tabla 5-6 se presenta la agenda de uso del agente RKU-C1 que el agente PAGE genera a partir del programa de abastecimiento del Proveedor (Tabla 5.2). En dicha tabla también se representa el valor mínimo y máximo de la disponibilidad del recurso, y el valor inicial. En la Figura 5.6 se representa gráficamente el plan de estado

(perfil de requerimiento) inicial que el agente RKU-C1 genera a partir de su agenda de uso. Dicha figura corresponde a una captura de pantalla del reporte gráfico generado por el prototipo.



**Figura 5.5: Perfil de inventario del agente MKU-P2-P**

Valor Mínimo de la Variable: 0					
Valor Máximo de la Variable: 500					
Nivel de Servicio: 1					
Valor Inicial: 0					
Id-Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día1	0	120	Principio	SP-4-P
2	Día1	0	100	Principio	SP-5-P
3	Día1	0	210	Principio	SP-6-P
4	Día5	0	120	Principio	SP-17-P
5	Día5	0	100	Principio	SP-18-P
6	Día5	0	210	Principio	SP-19-P

**Tabla 5-6: Agenda de uso del agente RKU-C1**

En las Tablas 5-7 a 5-12 se presenta el plan de actividades de cada uno de los agentes SP-x-P para  $x \in X = \{4, 5, 6, 17, 18, 19\}$  que el agente PAGE genera a partir del programa de abastecimiento del Proveedor (Tabla 5.2).

Id-Actividad	Inicio	Duración [hs]	Recurso	Id-Orden	Inicio	Duración [hs]	Cantidad	Modo
4	Día1	2	MKU-P11-P	2	Día1	0	-120	c
			RKU-C1	1	Día1	2	120	u
			IOA-Dist	1	Día2	0	+120	p

**Tabla 5-7: Plan de actividad del SP-4-P**

<b>Id-Actividad</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Recurso</b>	<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Cantidad</b>	<b>Modo</b>
17	Día5	2	MKU-P11-P	7	Día5	0	-120	c
			RKU-C1	4	Día5	2	120	u
			IOA-Dist	4	Día6	0	+120	p

**Tabla 5-8: Plan de actividad del SP-17-P**

<b>Id-Actividad</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Recurso</b>	<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Cantidad</b>	<b>Modo</b>
5	Día1	2	MKU-P12-P	2	Día1	0	-100	c
			RKU-C1	2	Día1	2	100	u
			IOA-Dist	2	Día2	0	+100	p

**Tabla 5-9: Plan de actividad del SP-5-P**

<b>Id-Actividad</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Recurso</b>	<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Cantidad</b>	<b>Modo</b>
18	Día5	2	MKU-P12-P	6	Día5	0	-100	c
			RKU-C1	5	Día5	2	100	u
			IOA-Dist	5	Día6	0	+100	p

**Tabla 5-10: Plan de actividad del SP-18-P**

<b>Id-Actividad</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Recurso</b>	<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Cantidad</b>	<b>Modo</b>
6	Día1	2	MKU-P2-P	2	Día1	0	-210	c
			RKU-C1	3	Día1	2	210	u
			IOA-Dist	3	Día2	0	+210	p

**Tabla 5-11: Plan de actividad del SP-6-P**

<b>Id-Actividad</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Recurso</b>	<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Cantidad</b>	<b>Modo</b>
19	Día5	2	MKU-P2-P	6	Día5	0	-210	c
			RKU-C1	6	Día5	2	210	u
			IOA-Dist	6	Día6	0	+210	p

**Tabla 5-12: Plan de actividad del SP-19-P**

En la Tabla 5-13 se presenta el programa de órdenes del agentes IOA-Dist que el agente PAGE genera a partir del programa de abastecimiento del Proveedor (Tabla 5.2).

<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [hs]</b>	<b>Cantidad</b>	<b>Tipo</b>	<b>SP</b>	<b>Producto</b>
1	Día1	2	120	Principio	SP-4-P	P11
2	Día1	2	100	Principio	SP-17-P	P12
3	Día1	2	210	Principio	SP-5-P	P2
4	Día5	2	120	Principio	SP-18-P	P11
5	Día5	2	100	Principio	SP-6-P	P12
6	Día5	2	210	Principio	SP-19-P	P2

**Tabla 5-13: Programa del agente IOA-Dist**

### 5.3.2 MODELADO DEL PROGRAMA DE ABASTECIMIENTO DEL DISTRIBUIDOR

El Distribuidor es responsable de abastecer, dentro de la cadena de suministro, a los Minoristas. Su red de distribución involucra el envío de los productos desde el almacén central a la sucursal y al Minorista 1, y desde la sucursal al Minorista 2. Los recursos utilizados y los tiempos requeridos para cada recorrido de la red se presentan en la Tabla 5-1. Su programa de abastecimiento se presenta en la Tabla 5-14. El mismo es constituido por 52 órdenes, de las cuales 11 son órdenes con destino no identificado.

Es de observar en la Tabla 5-14 que: los productos en consideración son en general provistos al Distribuidor por el Proveedor, la excepción la constituye la orden 28 cuyo origen es desconocido; el conjunto de órdenes {10, 11, 12, 41, 42, 43} no tiene especificado recurso de transporte, esto se debe a que dichas órdenes se corresponden con el conjunto de órdenes de abastecimiento {4, 5, 6, 17, 18, 19} del Proveedor, cuyo proceso de transporte está a cargo del Proveedor, quien utiliza el Camión 1 que es un recurso de su propiedad, y por lo tanto quien controla su ejecución a través del agente RKU-C1 en la Figura 5.2.

<b>Id-Orden</b>	<b>Inicio</b>	<b>Origen</b>	<b>Destino</b>	<b>Producto</b>	<b>Cantidad</b>	<b>Recurso</b>
1	Día1	Central	Minorista1	P12	30	Camión3
2	Día1	Central	Minorista1	P2	40	Camión3
3	Día1	Central	Sucursal	P11	60	Camión2
4	Día1	Central	Sucursal	P12	20	Camión2
5	Día1	Central	Sucursal	P2	70	Camión2
6	Día1	Sucursal	Minorista2	P11	25	Camión4
7	Día1	Sucursal	Minorista2	P2	30	Camión4
8	Día1	Sucursal	-	P12	15	-
9	Día1	Sucursal	-	P11	10	-
10	Día2	Proveedor	Central	P11	120	-
11	Día2	Proveedor	Central	P12	100	-
12	Día2	Proveedor	Central	P2	210	-
13	Día2	Central	Minorista1	P12	30	Camión3
14	Día3	Central	Minorista1	P2	70	Camión3
15	Día2	Sucursal	Minorista2	P11	25	Camión4
16	Día2	Sucursal	Minorista2	P2	20	Camión4
17	Día2	Sucursal	-	P2	10	-
18	Día3	Central	Minorista1	P12	30	Camión3
19	Día3	Central	Minorista1	P2	40	Camión3
20	Día3	Central	Sucursal	P11	60	Camión2
21	Día3	Central	Sucursal	P12	20	Camión2
22	Día2	Central	Sucursal	P2	70	Camión2
23	Día3	Sucursal	Minorista2	P11	25	Camión4

24	Día3	Sucursal	Minorista2	P2	30	Camión4
25	Día3	Sucursal	-	P11	10	-
26	Día3	Sucursal	-	P12	5	-
27	Día4	Central	Minorista1	P12	30	Camión3
28	Día4	-	Central	P2	30	-
29	Día4	Sucursal	Minorista2	P11	25	Camión4
30	Día3	Sucursal	Minorista2	P2	40	Camión4
31	Día4	Sucursal	-	P2	10	-
32	Día5	Central	Minorista1	P12	30	Camión3
33	Día5	Central	Minorista1	P2	110	Camión3
34	Día5	Central	Sucursal	P11	60	Camión2
35	Día5	Central	Sucursal	P12	20	Camión2
36	Día4	Central	Sucursal	P2	80	Camión2
37	Día5	Sucursal	Minorista2	P11	25	Camión4
38	Día5	Sucursal	Minorista2	P2	40	Camión4
39	Día5	Sucursal	-	P11	10	-
40	Día5	Sucursal	-	P12	15	-
41	Día6	Proveedor	Central	P11	120	-
42	Día6	Proveedor	Central	P12	100	-
43	Día5	Proveedor	Central	P2	210	-
44	Día6	Central	Minorista1	P12	30	Camión3
45	Día6	Central	Minorista1	P2	40	Camión3
46	Día6	Sucursal	Minorista2	P11	25	Camión4
47	Día5	Sucursal	Minorista2	P2	30	Camión4
48	Día6	Sucursal	-	P11	10	-
49	Día6	Sucursal	-	P12	15	-
50	Día7	Sucursal	Minorista2	P11	25	Camión4
51	Día6	Sucursal	Minorista2	P2	30	Camión4
52	Día7	Sucursal	-	P2	10	-

**Tabla 5-14: Programa de abastecimiento del Distribuidor**

En base al programa de abastecimiento de la Tabla 5-14, el agente PAGE de la agencia del Distribuidor generará los agentes necesarios para representarlo junto con sus datos y planes asociados. En la Figura 5.6 se representa la red de agentes resultante.

Los agentes MKU-y-C para  $y \in Y = \{P11, P12, P2\}$  representan los productos en el almacén central del Distribuidor, mientras que los agentes MKU-y-S para  $y \in Y = \{P11, P12, P2\}$  representan los productos en la sucursal del Distribuidor. Los agente RKU-Cz para  $z \in Z = \{2, 3, 4\}$  representa al recurso Camión  $z$  utilizado por los procesos de transporte que tienen lugar en la red del Distribuidor. Los agentes SP-x-P para  $x \in X = \{1-7, 10-16, 18-24, 27, 29-30, 32-38, 41-47, 50, 51\}$  representan los procesos de transporte asociados a cada orden. El agente IOA-Proveedor es el agente de interfaz con el Proveedor, y los agentes IOA-Minorista1 e IOA-Monorista2 son los agentes de interfaz con los Minoristas 1 y 2 respectivamente.

Los agentes MKU-y-S, para y en  $Y = \{P11, P12, P2\}$  y el agente MKU-P2-C poseen órdenes con destino no identificado, por lo cual asumen el rol de agentes de frontera para gestionar los eventos de frontera que pueden ser generados por dichas órdenes.

El conjunto de órdenes  $\{10, 11, 12, 41, 42, 43\}$  definen el programa de órdenes del agente IOA-Proveedor, el que se detalla en la Tabla 5-15. En esta tabla la duración toma valor 0 porque el camión se descarga en menos de una hora, por lo cual para un período de un día el ingreso se modela como instantáneo.

<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [días]</b>	<b>Cantidad</b>	<b>Tipo</b>	<b>SP</b>	<b>Producto</b>
10	Día 2	0	120	Principio	SP-10-C	P11
11	Día 2	0	100	Principio	SP-11-C	P12
12	Día 2	0	210	Principio	SP-12-C	P2
41	Día 6	0	120	Principio	SP-41-C	P11
42	Día 6	0	100	Principio	SP-42-C	P12
43	Día 6	0	210	Principio	SP-43-C	P2

**Tabla 5-15: Programa del agente IOA-Proveedor**

En las Tablas 5-16 y 5-17 se muestran los programas de órdenes de los agentes IOA-Minorista1 e IOA-Minorista2.

<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [días]</b>	<b>Cantidad</b>	<b>Tipo</b>	<b>SP</b>	<b>Producto</b>
1	Día1	-	30	Medio	SP-1-C	P12
2	Día1	-	40	Medio	SP-2-C	P2
13	Día2	-	30	Medio	SP-13-C	P12
14	Día3	-	70	Medio	SP-14-C	P2
18	Día3	-	30	Medio	SP-18-C	P12
19	Día3	-	40	Medio	SP-19-C	P2
27	Día4	-	30	Medio	SP-27-C	P12
32	Día5	-	30	Medio	SP-32-C	P12
33	Día5	-	110	Medio	SP-33-C	P2
44	Día6	-	30	Medio	SP-44-C	P12
45	Día6	-	40	Medio	SP-45-C	P2

**Tabla 5-16: Programa del agente IOA-Minorista1**

<b>Id-Orden</b>	<b>Inicio</b>	<b>Duración [días]</b>	<b>Cantidad</b>	<b>Tipo</b>	<b>SP</b>	<b>Producto</b>
6	Día1	-	25	Medio	SP-6-S	P11
7	Día1	-	30	Medio	SP-7-S	P2
15	Día2	-	25	Medio	SP-15-S	P11
16	Día2	-	20	Medio	SP-16-S	P2
23	Día3	-	25	Medio	SP-23-S	P11
24	Día3	-	30	Medio	SP-24-S	P2
29	Día4	-	25	Medio	SP-29-S	P11
30	Día3	-	40	Medio	SP-30-S	P2

37	Día5	-	25	Medio	SP-37-S	P11
38	Día5	-	40	Medio	SP-38-S	P2
46	Día6	-	25	Medio	SP-46-S	P11
47	Día5	-	30	Medio	SP-47-S	P2
50	Día7	-	25	Medio	SP-50-S	P11
51	Día6	-	30	Medio	SP-51-S	P2

**Tabla 5-17: Programa del agente IOA-Minorista2**

La lista de entrada/salida de los agentes MKU-y-C y MKU-y-S para y en  $Y = \{P12, P2\}$  se presentan en las Tablas 5-18 a 5-21 dado que dichos agentes participarán en las negociaciones que tendrá lugar en los escenarios mostrados como ejemplo de eventos disruptivos gestionados utilizando el prototipo, presentados posteriormente.

Valor Mínimo de la Variable: 20					
Valor Máximo de la Variable: 120					
Nivel de Servicio: 1					
Inventario Inicial: 70					
Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día 1	-	-30	Principio	SP-1-C
2	Día 1	-	-20	Principio	SP-4-C
3	Día 2	-	100	Principio	SP-11-C
4	Día 2	-	- 30	Principio	SP-13-C
5	Día 3	-	-20	Principio	SP-21-C
6	Día 3	-	-30	Principio	SP-18-C
7	Día 4	-	-30	Principio	SP-27-C
8	Día 5	-	-30	Principio	SP-32-C
9	Día 5	-	-20	Principio	SP-35-C
10	Día 5	-	100	Principio	SP-42-C
11	Día 6	-	-30	Principio	SP-44-C

**Tabla 5-18: Lista de Entrada/Salida del agente MKU-P12-C**

Valor Mínimo de la Variable: 30					
Valor Máximo de la Variable: 210					
Nivel de Servicio: 1					
Inventario Inicial: 150					
Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día 1	-	-40	Principio	SP-2-C
2	Día 1	-	-70	Principio	SP-5-C
3	Día 2	-	210	Principio	SP-12-C
4	Día 3	-	-70	Principio	SP-14-C
5	Día 3	-	-40	Principio	SP-19-C
6	Día 2	-	-70	Principio	SP-22-C
7	Día 4	-	40	Principio	-
8	Día 5	-	-110	Principio	SP-33-C
9	Día 4	-	-80	Principio	SP-36-C
10	Día 5	-	210	Principio	SP-43-C
11	Día 6	-	-40	Principio	SP-45-C

**Tabla 5-19: Lista de Entrada/Salida del agente MKU-P2-C**

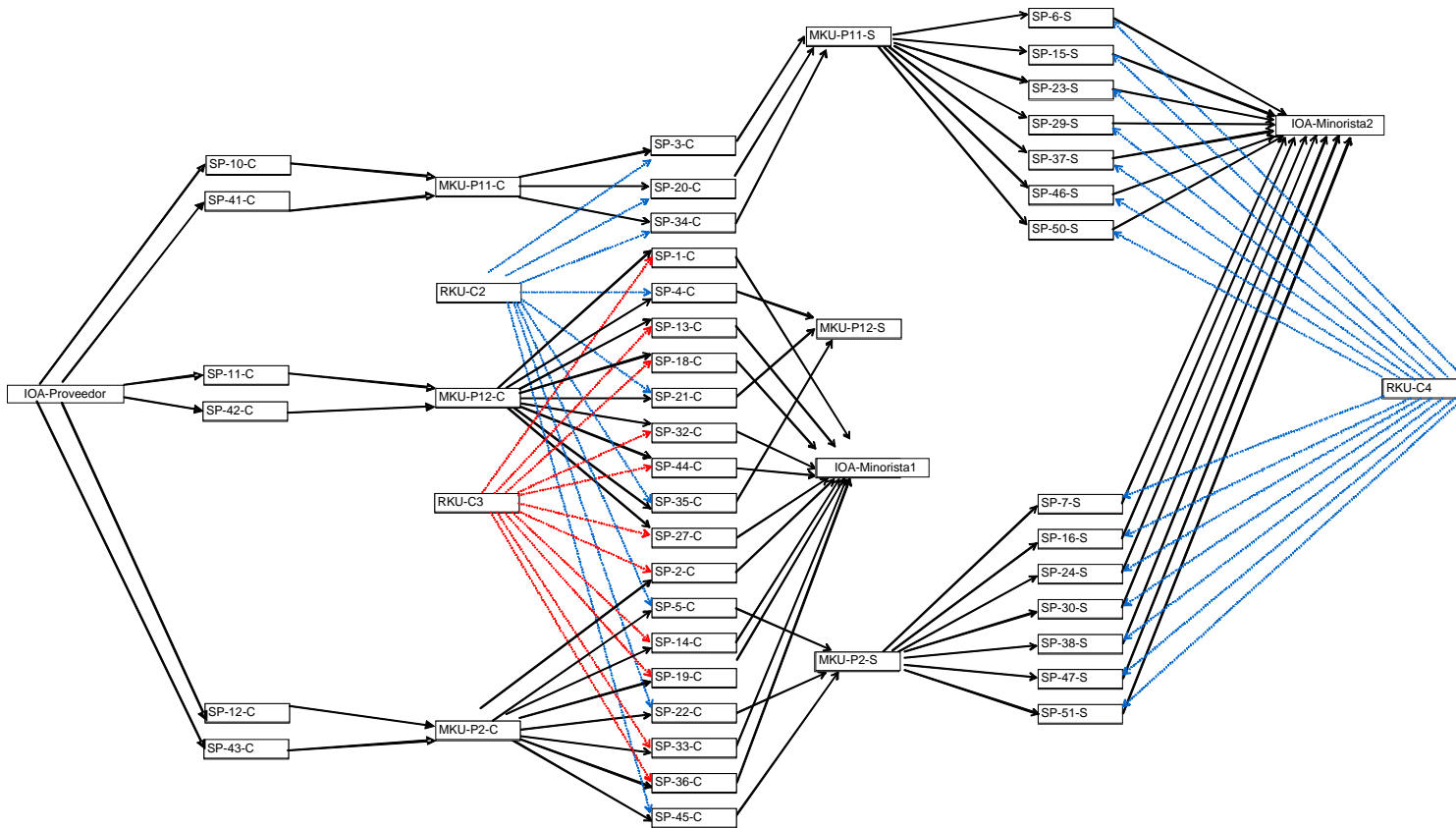


Figura 5.6: Agencia que modela el programa de abastecimiento del Distribuidor



Valor Mínimo de la Variable: 5					
Valor Máximo de la Variable: 25					
Nivel de Servicio: 1					
Inventario Inicial: 10					
Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día 1	-	20	Principio	SP-4-C
2	Día 1	-	-15	Medio	-
3	Día 3	-	20	Principio	SP-21-C
4	Día 3	-	-5	Medio	-
5	Día 5	-	20	Principio	SP-35-C
6	Día 5	-	-15	Medio	-

**Tabla 5-20: Lista de Entrada/Salida del agente MKU-P12-S**

Valor Mínimo de la Variable: 10					
Valor Máximo de la Variable: 120					
Nivel de Servicio: 1					
Inventario Inicial: 40					
Orden	Inicio	Duración	Cantidad	Tipo	SP
1	Día 1	-	70	Principio	SP-5-C
2	Día 1	-	-30	Principio	SP-7-S
3	Día 2	-	-20	Principio	SP-16-S
4	Día 2	-	-10	Medio	-
5	Día 2	-	70	Principio	SP-22-C
6	Día 3	-	-30	Principio	SP-24-S
7	Día 3	-	-40	Principio	SP-30-S
8	Día 4	-	-10	Medio	-
9	Día 4	-	80	Principio	SP-36-C
10	Día 5	-	-40	Principio	SP-38-S
11	Día 5	-	-30	Principio	SP-47-S
12	Día 6	-	-30	Principio	SP-51-S
13	Día 7	-	-10	Principio	SP-52_S

**Tabla 5-21: Lista de Entrada/Salida del agente MKU-P2-S**

La lista de entrada/salida de los agentes MKU-P11-C y MKU-P11-S, la agenda de uso de los agentes RKU-C<sub>z</sub> para  $z \text{ en } Z = \{2, 3, 4\}$  y el programa de actividades de los agentes SP- $w$ -C para  $w \text{ en } W = \{1-5, 10-14, 18-22, 27, 32-36, 41-45\}$  y SP- $u$ -S para  $u \text{ en } U = \{6, 7, 15, 16, 23, 24, 29, 30, 37, 38, 46, 47, 50, 51\}$  no se presentan debido a su extensión.

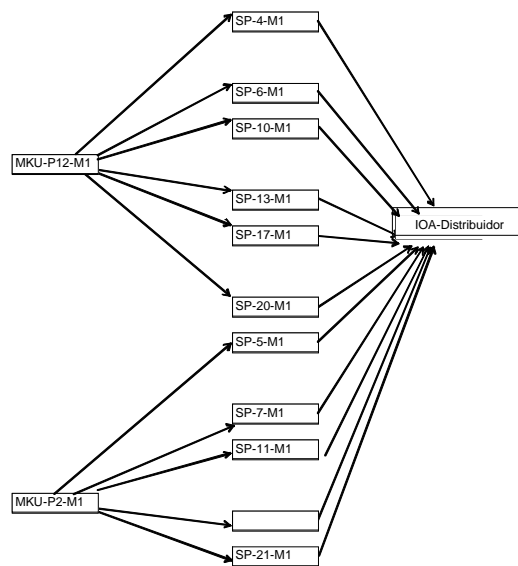
### 5.3.3 MODELADO DEL PROGRAMA DE ABASTECIMIENTO DEL MINORISTA 1

El Minorista 1 es responsable de abastecer, dentro de la cadena de suministro, a los clientes de su región. Su programa de abastecimiento se presenta en la Tabla 5-22. El mismo es constituido por 23 órdenes, de las cuales 12 son órdenes con destino a

clientes que no participan del acuerdo de colaboración, por lo cual corresponden a órdenes con origen no identificado. Las otras 11 corresponden a órdenes de abastecimiento por parte del Distribuidor.

<b>Id-Orden</b>	<b>Inicio</b>	<b>Origen</b>	<b>Destino</b>	<b>Producto</b>	<b>Cantidad</b>	<b>Recurso</b>
1	Día1	Minorista1	Cliente	P12	10	-
2	Día1	Minorista1	Cliente	P2	25	-
3	Día1	Minorista1	Cliente	P2	5	-
4	Día1	Central	Minorista1	P12	30	-
5	Día1	Central	Minorista1	P2	40	-
6	Día2	Central	Minorista1	P12	30	-
7	Día3	Central	Minorista1	P2	70	-
8	Día3	Minorista1	Cliente	P12	35	-
9	Día3	Minorista1	Cliente	P2	10	-
10	Día3	Central	Minorista1	P12	30	-
11	Día3	Central	Minorista1	P2	40	-
12	Día3	Minorista1	Cliente	P2	10	-
13	Día4	Central	Minorista1	P12	30	-
14	Día4	Minorista1	Cliente	P12	35	-
15	Día4	Minorista1	Cliente	P12	35	-
16	Día4	Minorista1	Cliente	P2	30	-
17	Día5	Central	Minorista1	P12	30	-
18	Día5	Central	Minorista1	P2	110	-
19	Día5	Minorista1	Cliente	P2	10	-
20	Día6	Central	Minorista1	P12	30	-
21	Día6	Central	Minorista1	P2	40	-
22	Día7	Minorista1	Cliente	P2	20	-
23	Día7	Minorista1	Cliente	P12	40	-

**Tabla 5-22: Programa de abastecimiento del Minorista1**



**Figura 5.7: Agencia que modela el programa de abastecimiento del Minorista1**

El producto P11 no forma parte del programa de abastecimiento. Ninguna orden tiene especificados recursos de distribución, esto se debe a que los clientes retiran el producto directamente del local de venta, mientras que las órdenes del conjunto {4-6, 10, 11, 13, 17-20} se corresponden con las órdenes de abastecimiento del Distribuidor (Tabla 5-16), cuyo proceso de transporte está a cargo del mismo, quien utiliza el Camión 3 que es un recurso de su propiedad, y por lo tanto quien controla su ejecución a través del agente RKU-C3 en la Figura 5.6.

En base al programa de abastecimiento de la Tabla 5-22, el agente PAGE de la agencia del Minorista 1 generará los agentes necesarios para representarlo junto con sus datos y planes asociados. En la Figura 5.7 se representa la red de agentes resultante.

### 5.3.4 MODELADO DEL PROGRAMA DE ABASTECIMIENTO DEL MINORISTA 2

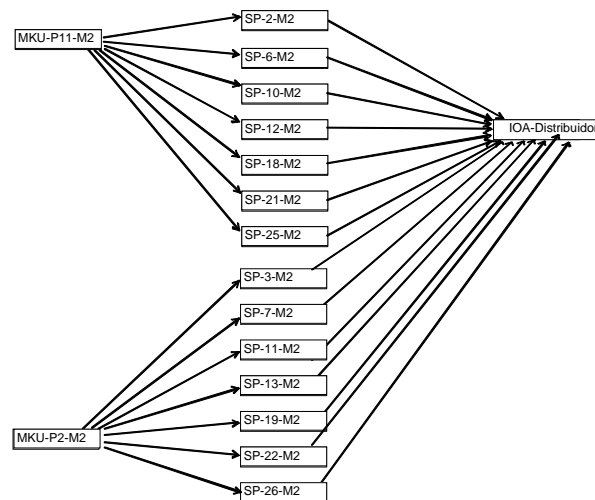
El Minorista 2 es responsable de abastecer, dentro de la cadena de suministro, a los clientes de su región. Su programa de abastecimiento se presenta en la Tabla 5-23. El mismo es constituido por 26 órdenes, de las cuales 12 son órdenes con destino a clientes que no participan del acuerdo de colaboración, por lo cual corresponden a órdenes con origen no identificado. El producto P12 no forma parte del programa de abastecimiento. Al Igual que para el Minorista 1, ninguna orden tiene especificados recursos de distribución, esto se debe a que los clientes retiran el producto directamente del local de venta, mientras que las órdenes del conjunto {2, 3, 6, 7, 10-13, 18-21, 25, 26} se corresponden con las órdenes de abastecimiento del Distribuidor (Tabla 5-17), cuyo proceso de transporte está a cargo del mismo, quien utiliza el Camión 4 que es un recurso de su propiedad, y por lo tanto quien controla su ejecución a través del agente RKU-C4 en la Figura 5.6.

<b>Id-Orden</b>	<b>Inicio</b>	<b>Origen</b>	<b>Destino</b>	<b>Producto</b>	<b>Cantidad</b>	<b>Recurso</b>
1	Día1	Minorista2	Cliente	P2	40	-
2	Día1	Sucursal	Minorista2	P11	25	-
3	Día1	Sucursal	Minorista2	P2	30	-
4	Día1	Minorista2	Cliente	P11	20	-
5	Día2	Minorista2	Cliente	P2	15	-
6	Día2	Sucursal	Minorista2	P11	25	-
7	Día2	Sucursal	Minorista2	P2	20	-
8	Día2	Minorista2	Cliente	P2	40	-
9	Día2	Minorista2	Cliente	P11	40	-

10	Día3	Sucursal	Minorista2	P11	25	-
11	Día3	Sucursal	Minorista2	P2	30	-
12	Día4	Sucursal	Minorista2	P11	25	-
13	Día3	Sucursal	Minorista2	P2	40	-
14	Día4	Minorista2	Cliente	P11	20	-
15	Día4	Minorista2	Cliente	P2	40	-
16	Día5	Minorista2	Cliente	P11	40	-
17	Día5	Minorista2	Cliente	P2	15	-
18	Día5	Sucursal	Minorista2	P11	25	-
19	Día5	Sucursal	Minorista2	P2	30	-
20	Día6	Minorista2	Cliente	P2	40	-
21	Día6	Sucursal	Minorista2	P11	25	-
22	Día5	Sucursal	Minorista2	P2	40	-
23	Día6	Minorista2	Cliente	P11	20	-
24	Día6	Minorista2	Cliente	P2	40	-
25	Día7	Sucursal	Minorista2	P11	25	-
26	Día6	Sucursal	Minorista2	P2	30	-

**Tabla 5-23: Programa de abastecimiento del Minorista2**

En base al programa de abastecimiento de la Tabla 5-23, el agente PAGE de la agencia del Minorista 2 generará los agentes necesarios para representarlo junto con sus datos y planes asociados. En la Figura 5.8 se representa la red de agentes resultante.



**Figura 5.8: Agencia que modela el programa de abastecimiento del Minorista2**

Las Figuras 5.2, 5.6, 5.7 y 5.8 definen las cuatro agencias de la cadena de suministro del caso de estudio que representan los programas de suministros acordados y sincronizados por las cuatro entidades de negocio de la cadena que participan del acuerdo de colaboración, para sus tres productos principales. Dichas agencias operan de manera independiente para gestionar los eventos disruptivos que pudieran afectar el

programa de abastecimiento que cada una representa, y podrán operar en coordinación a través de los correspondientes agentes IOA cuando lo consideren necesario para buscar una solución a una excepción.

## 5.4. ANALISIS DE RESULTADOS

El caso de estudio ha sido utilizado para validar los conceptos del sistema SCEM que implementa el modelo de gestión de eventos disruptivos propuesto en esta tesis. El objetivo de la validación se limitó a verificar que el sistema puede responder a los eventos disruptivos que se van sucediendo a medida que los programas de abastecimiento se ejecutan, tomando decisiones válidas desde la perspectiva de un decisor humano.

### 5.4.1. PROCESO DE INSTANCIACIÓN

A partir de la información descrita previamente se generaron los archivos XML para su uso en el prototipo. Para esta primera versión del prototipo el proceso de negociación se limitó a mecanismos de colaboración de primer nivel (Sección 2.3.3.3, Capítulo 2).

El archivo XML completo para el caso de estudio contiene 2239 líneas de código equivalente a 43 páginas en el formato de esta tesis, por lo cual solo a modo ilustrativo en la Figura 5.9 se presenta una pequeña porción del mismo.

```
<scems:scem xmlns:scems="scems:agentes">
  <rkus>
    <rku>
      <nombre>mku-p1 l-c</nombre>
      <ae>ae</ae>
      <configuracion>
        <valorMinimoVariable>20</valorMinimoVariable>
        <valorMaximoVariable>150</valorMaximoVariable>
        <nivelServicioObjetivo>1</nivelServicioObjetivo>
      </configuracion>
      <ordenes>
        <orden>
          <numeroOrden>0</numeroOrden>
          <inicio>0</inicio>
          <duracion>1</duracion>
          <cantidad>80</cantidad>
          <tipoCambioVariable>1</tipoCambioVariable>
          <nombreSp></nombreSp>
        </orden>
        ...
      </ordenes>
    </rku>
  </rkus>
</scems:scem>
```

```

        </ordenes>
        <estrategia>inicioDesvio</estrategia>
        <calculadorUtilidad>default</calculadorUtilidad>
    </rku>
    ...
    </rku>
</rkus>
<sps>
    <sp>
        <nombre>sp-3-c</nombre>
        <idActividad>3</idActividad>
        <inicio>1</inicio>
        <duracion>2</duracion>
        <extension>0</extension>
        <unificadorPropuestas>default</unificadorPropuestas>
        <items>
            <item>
                <nombreRku>mku-p11-c</nombreRku>
                <numeroOrdenRku>1</numeroOrdenRku>
                <variacionCantidadRku>-60</variacionCantidadRku>
            </item>
            <item>
                <nombreRku>rku-C2-c</nombreRku>
                <numeroOrdenRku>1</numeroOrdenRku>
                <variacionCantidadRku>60</variacionCantidadRku>
            </item>
            <item>
                <nombreRku>mku-p11-s</nombreRku>
                <numeroOrdenRku>1</numeroOrdenRku>
                <variacionCantidadRku>60</variacionCantidadRku>
            </item>
        </items>
    </sp>
    ...
</sps>
</scems:scem>

```

**Figura 5.9: Extracto del archivo de configuración XML para la definición de la agencia**

---

### 5.4.2. ESCENARIOS DE PRUEBA

Se han realizado pruebas con una muestra variada de eventos disruptivos. En todos los casos los resultados fueron los esperados. Esto fue fácil de verificar manualmente debido a la restricción de mecanismos de colaboración de primer nivel, el cual reduce el número de alternativas a considerar a unas pocas. Si bien los resultados son de algún modo obvios para los responsables de tomar decisiones, la importancia del aporte radica en la automatización del proceso de identificación de excepción, búsqueda de solución e implementación de la misma. A modo de ejemplo se presentan tres escenarios de prueba, cada uno de ellos basado en un evento disruptivo que usualmente se suele presentar en este tipo de cadena de suministro. Para cada escenario se presentan

detalles de la respuesta de la agencia a través de los reportes gráficos proporcionados por el prototipo.



**Figura 5.10: Planes de estados de los agentes involucrados en el escenario 1**

***Escenario de Prueba 1***

En el primer escenario se presenta como variación un incremento en 10 unidades de la cantidad inicial del inventario del MKU-P12-S. Esta variación pudo ser producida por ejemplo por la reducción en 10 unidades de una orden de abastecimiento con origen en la sucursal y destino no identificado o por devolución a sucursal del producto P12.

La información relacionada con este evento disruptivo, que ingresa al prototipo a través del agente de eventos AE para la simulación, es la siguiente:

idEvento: 1  
 receiver: MKU-P12-S  
 numeroOrden: 0  
 VariacionFechaInicio: 0  
 VariaciónCantidad: +10  
 MilisegundosDemoraDisparo: 2000

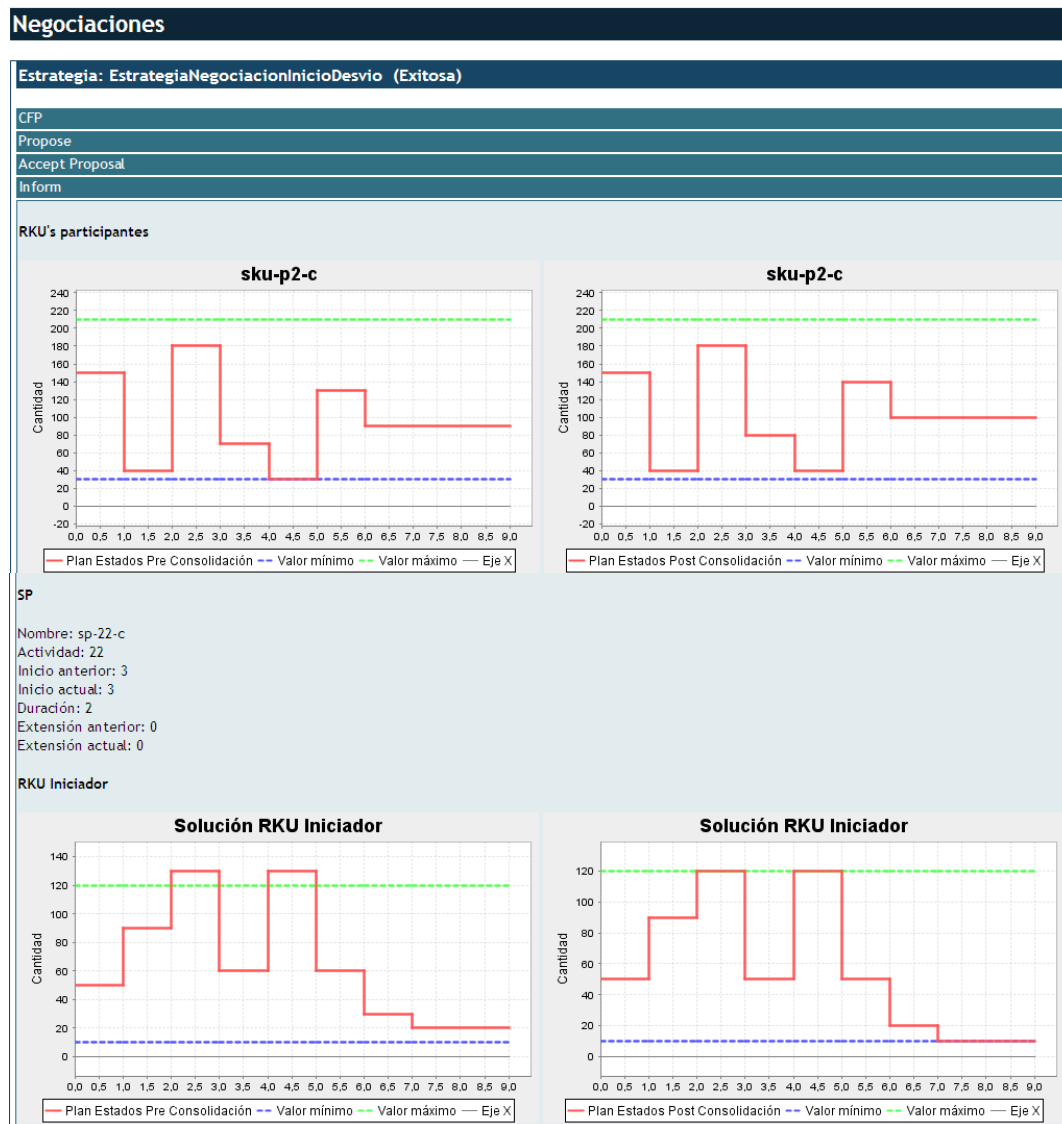


Figura 5.11: Planes de estados de los agentes involucrados en el escenario 2



El evento es identificado con el número 1, afecta al MKU-P12-S, el número de la orden afectada se identifica con 0, representando al inventario inicial. El evento ocurre al inicio del horizonte de tiempo, implicando un incremento en la cantidad del inventario inicial de 10 unidades del producto P12 en la sucursal.

El reporte generado por el prototipo se muestra en la Figura 5.10. En el primer cuadrante inferior se puede observar que como consecuencia del evento disruptivo el agente MKU-P12-S detectó que de mantenerse el programa de abastecimiento, a partir del día 3 el inventario superaría el nivel máximo permitido. Es decir que de manera anticipada identificó una excepción en el plan de estado (perfil de inventario). Por esta razón el agente inició un proceso de colaboración haciendo uso del protocolo DCN (rol RCU Iniciador). Para ello identificó por medio de la lista entrada/salida (Tabla 5-20) al agente SP-21-C el cual asumió el rol de mediador propagando el llamado a participar al agente MKU-P12-C cuyo perfil de inventario, previo a la modificación, se muestra en el primer cuadrante superior de la figura. La propuesta acordada por los agentes ha sido reducir en 10 unidades la orden en cuestión, que corresponde a la orden 21 del programa de abastecimiento del Distribuidor (Tabla 5-14). Los perfiles de inventario de ambos agentes, resultantes del acuerdo, se muestran en el segundo cuadrante inferior y superior respectivamente de la figura antes referida. Se puede observar en los mismos que con la modificación del programa de abastecimiento acordada la excepción fue resuelta.

### ***Escenario de Prueba 2***

En el segundo escenario se presenta como variación un incremento en 10 unidades de la cantidad inicial del inventario del MKU-P2-S, similar al caso del escenario de prueba 1. La información relacionada con este evento disruptivo que ingresa al prototipo a través del agente de eventos AE para la simulación es la siguiente:

idEvento: 2  
receiver: MKU-P2-S  
nroOrden: 0  
variacionFechaInicio: 0  
variacionCantidad: +10  
milisegundosdemoraDisparo: 0

El reporte generado por el prototipo se muestra en la Figura 5.11. En el primer cuadrante inferior se puede observar que como consecuencia del evento disruptivo el agente MKU-P2-S detectó que de mantenerse el programa de abastecimiento, durante

los periodos comprendidos entre los días 2-3 y 4-5 el inventario superaría el nivel máximo permitido. Es decir que de manera anticipada identificó una excepción en el plan de estado (perfil de inventario). Por esta razón el agente inició un proceso de colaboración haciendo uso del protocolo DCN (rol RKU Iniciador). Para ello identificó por medio de la lista entrada/salida (Tabla 5-21) al agente SP-22-C el cual asumió el rol de mediador propagando el llamado a participar al agente MKU-P2-C (sku-p2-c en el reporte) cuyo perfil de inventario previo a la modificación se muestra en el primer cuadrante superior de la figura. La propuesta acordada por los agentes ha sido reducir en 10 unidades la orden en cuestión, que corresponde a la orden 22 del programa de abastecimiento del Distribuidor (Tabla 5-14). Los perfiles de inventario de ambos agentes, resultantes del acuerdo, se muestran en el segundo cuadrante inferior y superior respectivamente de la figura antes referida. Se puede observar en los mismos que con la modificación del programa de abastecimiento acordada la excepción fue resuelta.

### ***Escenario de Prueba 3***

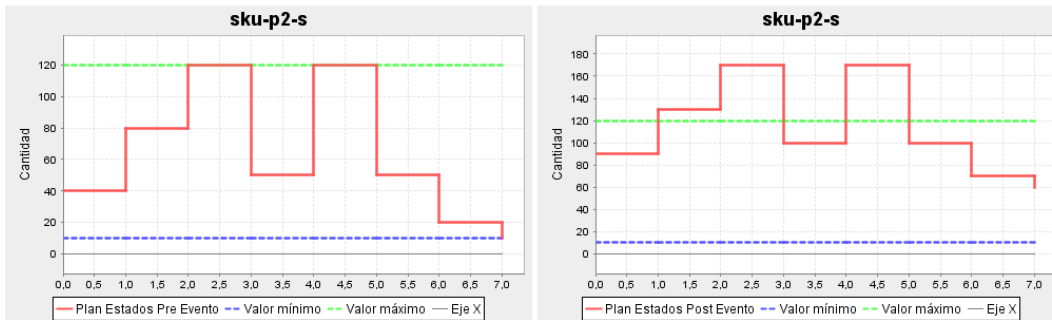
En el tercer escenario se presenta como variación un incremento en 50 unidades de la cantidad inicial del inventario del MKU-P2-S. La información relacionada con este evento que ingresa al prototipo a través del agente de eventos AE para la simulación es la siguiente:

```
idEvento: 3
receiver: MKU-P2-S
nroOrden: 0
variacionFechaInicio: 0
variacionCantidad: +50
milisegundosdemoraDisparo: 0
```

El reporte generado por el prototipo se muestra en la Figura 5.12. En este caso el reporte muestra en el primer cuadrante superior el perfil de inventario del agente MKU-P2-S (sku-p2-c en el reporte) previo a la modificación, y en el segundo cuadrante el perfil de inventario luego del evento disruptivo. En este último se puede observar que como consecuencia del evento disruptivo de mantenerse el programa de abastecimiento, a partir del día 1 se produciría una excepción. Por esta razón el agente inició un proceso de colaboración cuyo resultado fue la imposibilidad de resolver dicha excepción, como se observa en la indicación dada en la parte inferior del reporte, donde se especifica que no fue posible unificar una propuesta.

## Evento: 3

### RKU Iniciador



### Negociaciones

Estrategia: EstrategiaNegociacionInicioDesvio

CFP

Documento original:

Cambio: 2

Orden	Tiempo	Cantidad	Utilidad Originador	Utilidad Receptor
1	0	-50,0		

Documentos transformados:

sku-p2-c:

Cambio: 2

Orden	Tiempo	Cantidad	Utilidad Originador	Utilidad Receptor
2	0	50,0		

Propose

Documentos originales:

sku-p2-c :

-

Documento unificado :

No se pudo unificar una propuesta

Figura 5.12: Planes de estados de los agentes involucrados en el escenario 3

## 5.5. CONCLUSIONES

El caso de estudio permitió verificar utilizando el prototipo que el modelo propuesto en esta tesis permite automatizar el proceso de gestión de eventos disruptivos en la cadena de suministro. El sistema SCEM que implementa dicho modelo pudo detectar de manera anticipada una excepción que ocurriría a futuro a raíz de un evento disruptivo, llevar a cabo procesos de colaboración para resolver dicha excepción y

generar las modificaciones pertinentes en los programas de abastecimientos. Cuando no hubo solución pudo detectar la imposibilidad de resolverla.

En todos los casos, los resultados fueron los esperados por los responsables de tomar decisiones, ofreciendo de este modo una importante propuesta para la automatización del proceso de identificación de excepción, búsqueda de solución e implementación de la misma.

Algunos aspectos a destacar en relación a la prueba realizada son:

Si bien el trabajo de validación se limitó a mecanismos de colaboración de primer nivel, es esta una limitación de la primera versión del prototipo de prueba y no del modelo propuesto.

En esta primera prueba las instancias de las agencias fueron creadas en una sola computadora, por lo que no hay interacción entre agentes ubicados en diferentes locaciones, lo cual es un problema de implementación que no afecta el modelo.

Según se destacó en el Capítulo 4, en la arquitectura propuesta el Sistema SCEM es un componente que debe interactuar con el Sistema de Programación y el Sistema de Ejecución; si bien esta prueba se limitó a validar el mecanismo de coordinación, la interacción entre componentes se prevé realizar mediante archivos XML, lo cual no tiene consecuencias sobre el modelo. De hecho la información de entrada/salida del prototipo se realiza mediante archivos XML sin importar el proceso de su generación.

## CAPÍTULO 6 – CONCLUSIONES Y FUTUROS TRABAJOS

En las siguientes secciones es presentada una síntesis de la propuesta de solución al problema de la gestión de eventos en la cadena de suministro abordándose también aquellas oportunidades de futuras investigaciones.

### 6.1. CONCLUSIONES

Las características de la propuesta de solución para el problema de la gestión de eventos disruptivos en la cadena de suministro presentada en esta tesis quedaron plasmadas en los modelos desarrollados.

El modelo de componentes principales del sistema SCEM distribuido respeta la autonomía de las entidades de negocio que integran la cadena de suministro y la heterogeneidad de los sistemas que componen el soporte informático de las mismas.

El modelo funcional del proceso de gestión de eventos disruptivos describe el comportamiento autónomo de cada entidad de negocio que compone la cadena de suministro en la gestión de un evento disruptivo y el comportamiento frente a la necesidad de una acción en colaboración con otra entidad de negocio.

El modelo conceptual permite representar el problema en cuestión como una red de puntos de control definidos sobre los recursos o materiales conectados entre si mediante procesos de suministro.

El modelo del proceso de integración entre componentes, el proceso de identificación de una excepción y el proceso de coordinación para llevar a cabo una búsqueda de solución a una excepción en colaboración, permiten sistematizar la identificación de una excepción, búsqueda de solución e implementación de la misma.

A diferencia de las propuestas existentes, las cuales centran el monitoreo de eventos disruptivos en las órdenes de abastecimiento, esta propuesta centra el monitoreo y las acciones de control en los recursos. Esto representa un aspecto innovativo en la forma de modelar el problema cuya ventaja radica en que los recursos son los directamente afectados por los eventos disruptivos, mientras que los eventos disruptivos

que afectan a una orden de abastecimiento son consecuencia de los que afectan los recursos asociados a la misma.

El enfoque propuesto permite anticipar la ocurrencia de una excepción, que afecta una o más órdenes, producida por un evento disruptivo sobre un recurso. Esto permite tener más y mejores alternativas de decisión a la hora de buscar una solución a la disrupción.

Los modelos descriptos fueron implementados mediante agencias cuya arquitectura fue definida como una red de puntos de control sobre inventarios a cargo de agentes MKU cuyas relaciones son modeladas con puntos de enlaces gestionados por agentes SP, los cuales para su funcionamiento emplean recursos cuyo control está a cargo de agentes RKU, y agentes de servicio con funciones específicas de interfaz.

En el contexto del problema a resolver el proceso de coordinación principal se ha pensado como un mecanismo de reasignación de recursos. Los agentes RKU o MKU cuentan con una asignación inicial de holguras (*buffers* de recursos). Cuando un evento ocurre deben usar las holguras para poder hacer frente a la variación producida. Es decir los agentes intercambiarán holguras con el fin de reestablecer el equilibrio. Para el problema de coordinar la asignación de recursos se planteó una solución tomado como base la propuesta de Mercado Computacional la cual ofrece un paradigma de control descentralizado. En base a este paradigma se propuso el protocolo *Double Contract Net*, el cual da soporte al mecanismo de coordinación de la agencia, sustentado en el uso de agentes que actúan de mediadores entre el agente coordinador y los agentes participantes de la colaboración, lo cual facilita la escalabilidad de la agencia.

El comportamiento autónomo en la gestión de eventos disruptivos en la cadena de suministro de la agencia propuesta surge de la interacción entre agentes que llevan a cabo comportamientos relativamente sencillos. Cada agente se especializa en realizar una tarea, pero ninguno de ellos tiene la visión global del problema que se está resolviendo ni conoce el objetivo global del sistema. A partir de la interacción coordinada entre pares de agentes RKU o MKU y SP surge el comportamiento tendiente a resolver una excepción.

El rol de coordinador es asumido por el agente RKU o MKU que modela el punto de control donde se generó el evento. Una vez que un agente detecta una

excepción, necesita encontrar una solución llamando a colaborar a otros agentes para distribuir el desvío y de esta manera resolver la excepción. Dado que no conoce quienes son los agentes responsables de recursos relacionados debe interactuar con el agente SP correspondiente y éste es quien conoce los agentes responsables de estos recursos.

Para minimizar la cantidad de agentes participantes, el proceso es planteado por niveles pudiéndose definir la cantidad de niveles a involucrar o bien limitar esto por tiempo en el cual una solución debe ser hallada.

La herramienta de software desarrollada a nivel prototipo permite realizar simulaciones sobre escenarios configurables utilizando diferentes estrategias de búsqueda de solución ante cada evento disruptivo. Esto lo convierte en una herramienta de análisis en el ámbito de investigación, para evaluar el desempeño de diferentes algoritmos de búsqueda de solución.

Durante los procesos de negociación los agentes intercambian un documento de negocio que contiene la información que cada punto de control debe analizar al tomar decisiones. La definición de la estructura de este documento de negocio también formó parte de este trabajo, creando una ontología que provee una semántica común entre los agentes para que todos puedan interpretar la información de la misma manera.

El caso de estudio permitió verificar utilizando el prototipo del sistema SCEM que el modelo propuesto en esta tesis permite automatizar el proceso de gestión de eventos disruptivos en la cadena de suministro. El sistema SCEM que implementa dicho modelo mostró capacidad para detectar de manera anticipada excepciones que ocurrirían a futuro a raíz de la ocurrencia de eventos disruptivos simulados, llevar a cabo procesos de colaboración para resolver dichas excepción y generar las modificaciones pertinentes en los programas de abastecimientos. Cuando no hubo solución pudo detectar la imposibilidad de resolverla.

En todos los casos, los resultados fueron los esperados por los responsables de tomar decisiones, ofreciendo de este modo una importante propuesta para la automatización del proceso de identificación de excepción, búsqueda de solución e implementación de la misma.

Si bien el trabajo de validación se limitó a mecanismos de colaboración de primer nivel, es esta una limitación de la primera versión del prototipo de prueba y no del modelo propuesto.

La tecnología de agentes ha sido de gran importancia para el desarrollo de este prototipo inicial facilitando las tareas de escritura, “debuging” y mantenimiento dado el carácter pequeño de los componentes generados. Por otra parte, estas tareas se vieron facilitadas por el uso de la plataforma de desarrollo JADE.

## 6.2. FUTUROS TRABAJOS

A partir de lo desarrollado, surge un abanico de mejoras y extensiones a implementar para ampliar las funcionalidades existentes.

A nivel de modelo, entre las más significativas se puntualizan: desarrollar estrategias de negociación más complejas; en base a estas desarrollar nuevos algoritmos para la toma de decisión de los agentes; e incorporar información histórica para mejorar los procesos de toma de decisión de los agentes.

A nivel de implementación, generar mecanismos de persistencia del estado del agente y mejorar las pantallas de interacción con los usuarios.

En esta primera prueba las instancias de las agencias fueron creadas en una sola computadora, por lo que no hay interacción entre agentes ubicados en diferentes locaciones. La próxima versión del prototipo se desarrollará para que cada agencia opere en un lugar físico diferente atendiendo a la arquitectura de sistema propuesta. Si bien esto no producirá mejoras en el proceso de colaboración, permitirá operar atendiendo las necesidades de distribución física de las entidades de negocio que integran la cadena de suministro.

También se prevé utilizar la nueva versión del prototipo del Sistema SCEM interactuando con el Sistema de Programación y el Sistema de Ejecución de las entidades de negocio. Esto implica desarrollar en estos sistemas una interfaz que pueda generar y recibir los archivos XML con la información que requiere y genera el prototipo.



## GLOSARIO Y LISTADO DE SÍMBOLOS

**ACL** (*Agent Communication Language*): lenguaje empleado por los agentes para el intercambio de mensajes.

**Agenda de Uso:** representa el plan de cada RKU, detalla las órdenes del recurso indicando cuándo el recurso es requerido para su uso y cuánto de él se usará.

**Agente:** entidad de software autónoma o semi-autónoma capaz de percibir el ambiente por medio de sus sensors y de actuar en el mismo por medio de sus efectores, siendo reactivo, proactivo, flexible y cooperante.

**Agente RKU:** representa un punto de control para la gestión de eventos relacionados al recurso que representa. Desde la perspectiva de un problema de control, la variable monitoreada es la disponibilidad del recurso que representa y sus variables controladas son los tiempos y para algunos recursos la capacidad requerida. Existe un agente RKU por cada recurso relevante de la cadena de suministro. Son creados por el PAGE.

**Agente MKU:** es una especialización del agente RKU, por lo tanto es un punto de control especializado en la gestión de eventos sobre inventarios. Son creados por el PAGE recibiendo como entrada su Lista de Entrada/Salida y sus tres atributos básicos. La Lista de Entrada/Salida determina con cuales SPs se encuentran relacionados y si esas relaciones son de entrada o de salida. Desde la perspectiva de un problema de control la variable monitoreada es el inventario y sus variables controladas son los parámetros tiempo y cantidad de cada orden de su Lista de Entrada/Salida. Detecta los eventos comparando el valor actual con el valor planeado del inventario, el valor actual es obtenido a través del EVA.

**Agente SP:** representa un punto de enlace entre los distintos puntos de control (MKUs y RKUs). Representa una actividad de transición.

**CNP** (*Contract Net Protocol*): protocolo desarrollado para el control y la comunicación de nodos en un DPS. La distribución de la tarea es realizada por medio de un proceso de negociación, una discusión es llevada a cabo entre los nodos con tareas a ser ejecutadas y los nodos que pueden ejecutar esas tareas.

**FIPA** (*Foundation for Intelligent Physical Agents*): es una fundación internacional que trabaja en el desarrollo de estándares para los agentes y sistemas multi-agentes.

**Función de utilidad:** en este contexto es definida como la función que indica lo importante que es para el agente tener el recurso que se intercambia, indicando a su vez el estado en que se encuentra el agente con ese nivel del recurso.

**JADE** (*Java Agent Development Environment*): es un proyecto desarrollado por el CSELT (*Centro Studio e Laborati Telecomunicazione*), siendo la implementación más extendida del estándar FIPA. Es una herramienta de software totalmente implementada en Java, cuyo objetivo es simplificar el desarrollo de sistemas multi-agente a través de un conjunto de sistemas, servicios y agentes. Proporciona varios agentes ya implementados que pueden ser utilizados para distintos propósitos y están ligados a la plataforma.

**Lista de Entrada/Salida:** elemento de información fundamental de un MKU en la que se detallan todas las órdenes de entrada del material y de salida del mismo.

**MKU** (*Material Keeping Unit*): de los recursos existentes los inventarios constituyen un subgrupo de especiales características. Ejemplos de MKU: materiales, materia prima, productos en proceso y productos terminados; estos recursos son englobados bajo la común denominación de inventarios, el punto de control que representa un inventario recibe el nombre de MKU. Los eventos sobre los inventarios repercuten directamente en el comportamiento y éxito de la cadena de suministro. La función del MKU es la de gestionar los eventos que se producen en el inventario que representa. Un MKU se encuentra definido por tres atributos básicos, los cuales permiten su identificación unívocamente, estos atributos básicos son: material, envase y lugar.

**Perfil de Disponibilidad:** indica cuándo y cuánto el recurso estará disponible en el horizonte de planificación.

**Perfil de Factibilidad:** calculado a partir de la diferencia entre ambos perfiles, si este perfil determina regiones por debajo del eje x, el recurso se encuentra en un estado de infactibilidad ya que presenta al menos un requerimiento que no puede ser totalmente satisfecho, por otro lado las superficies definidas por el eje del tiempo y la curva del perfil de factibilidad definen la holgura del recurso.

**Perfil de Requerimientos:** es la curva que indica cuánta capacidad del recurso es requerida, cuándo y por cuánto tiempo.

**Perfil de Uso:** construido a partir de la agenda de uso, puede ser interpretado como la función de su variable de estado, indicando para el horizonte de planificación la capacidad usada y por cuánto tiempo es utilizado en esa capacidad.

**Plan de Actividad:** relacionado a un SP, detalla cuando la actividad se inicia, su duración y los recursos afectados (detallando el tiempo y la cantidad afectada o requerida), así como la Lista de Materiales. Sus elementos son:

**Plan de Estado:** Para un MKU es una función elaborada a partir de la Lista de Entrada/Salida la cual consiste de una lista de tuplas de la forma  $[i, t]$  que indica el inventario,  $i$ , esperado en el tiempo  $t$ .

**Programa de abastecimiento:** en el ámbito de las cadenas de suministro es definido como la asignación de materiales a distintas ubicaciones, donde la asignación es representada por órdenes de transferencia de materiales, indicando los recursos afectados en cada una de esas órdenes.

**Protocolos de comunicación:** permiten a los agentes intercambiar y comprender mensajes.

**RKU (*Resource Keeping Unit*):** es el nombre de los puntos de control, existe uno por cada recurso requerido en la cadena de suministro. Su función es gestionar los eventos que puedan alterar la ejecución del programa asignado al recurso que representan. Ejemplos de RKUs: unidades de procesamiento, equipos de transporte, etc.

**SCEM (*Supply Chain Event Management*):** herramienta que permite a las organizaciones responder a eventos no planeados, incrementando la visibilidad global de la cadena de suministro y reduciendo los costos asociados con la ocurrencia de estos eventos.

**Sistema:** conjunto de elementos o partes que interactúan entre sí con el fin de alcanzar un objetivo concreto. Existe una influencia mutua entre sus elementos.

**Sistemas Complejos:** se define como tal a un sistema compuesto por una gran cantidad de componentes interactuantes simples, capaces de intercambiar información con su entorno y capaces de adaptar su estructura interna como consecuencia de tales interacciones. Las interacciones pueden brindar comportamientos emergentes, coherentes y complejos.

**Agencia:** sistema compuesto de muchos agentes, los cuales son capaces de interactuar. La interacción puede tener lugar mediante mensajes o produciendo cambios en su ambiente común.

**SP (*Supply Process*):** puntos de enlace que establecen los vínculos entre los distintos puntos de control. Un SP representa la transición de uno o más MKUs a otro/s MKUs. que puede requerir el empleo de diferentes RKUs. Los SPs no son puntos de control sino procesos de balance, ellos establecen cómo los MKUs están vinculados y que RKUs son requeridos para que esa vinculación sea efectiva.

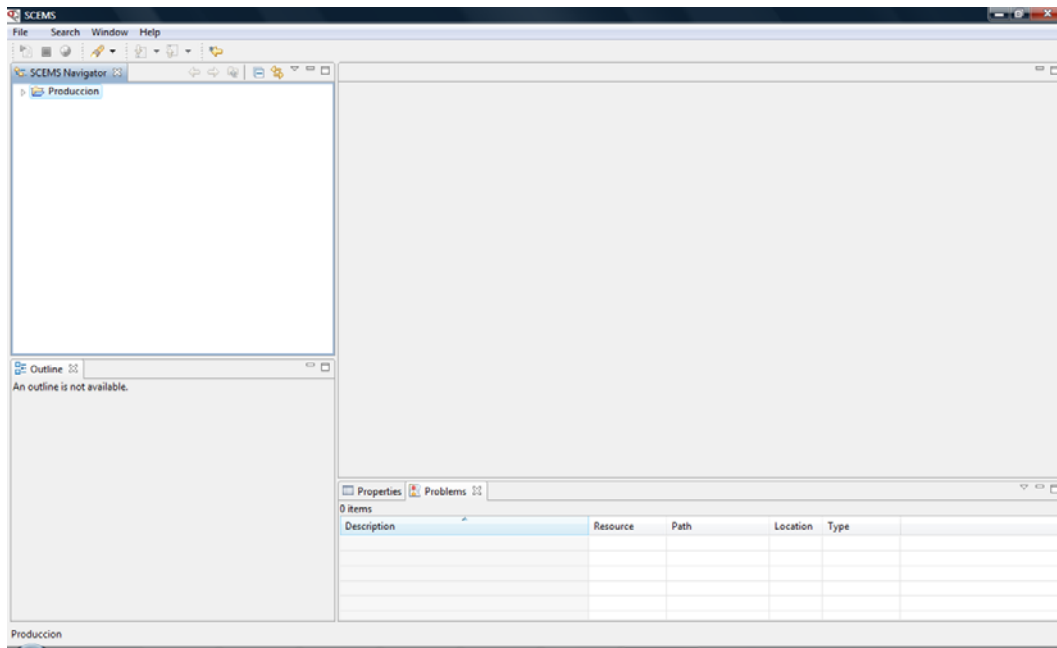
## ANEXO– EJECUCIÓN DEL SCEMS

En este anexo se describe cómo ejecutar el prototipo del sistema SCHEM denominado *SCEMS* y cómo interactuar con el mismo para construir los archivos XML de configuración de agencia y de eventos a disparar, ejecutar una simulación y evaluar sus resultados.

### A.1. SCEMS

SCEMS se ejecuta mediante el archivo *SCEMS.exe* que se encuentra en el directorio de instalación. Si el sistema fue instalado en un sistema operativo Windows, dependiendo de las opciones seleccionadas durante la instalación, también podría ejecutarse mediante el acceso en la carpeta correspondiente del menú “Inicio”, un icono en el escritorio y/o un icono de Inicio Rápido.

Al ejecutar este archivo se abre la ventana inicial de SCEMS, la cual se muestra en la Figura A.1.



**Figura A.1: Ventana inicial de SCEMS**

En esta ventana se pueden identificar las siguientes secciones:

- ❖ *Menú*: contiene todas las opciones disponibles que corresponden a: archivos (*File*), búsquedas (*Search*), ventanas y vistas (*Window*), ayuda (*Help*).
- ❖ *Barra de herramientas*: contiene accesos a las funciones más comunes. Estas funciones dependen del elemento con el cual se está trabajando.
- ❖ *SCEMS Navigator*: contiene y permite navegar los proyectos que utiliza SCEMS.
- ❖ *Outline*: cuando se está editando un archivo XML, permite ver su estructura en forma de árbol.
- ❖ *Área de trabajo y visualización de resultados*: área donde se abren los editores de archivos. Por ejemplo, aquí se visualizan los archivos XML cuando son abiertos.
- ❖ *Área con vistas Properties y Problems*: vistas que proporcionan información adicional, como ser las propiedades de un elemento y los errores que ocurren durante la ejecución de la aplicación.

La interacción normal con el sistema para ejecutar simulaciones y analizar sus resultados es la siguiente:

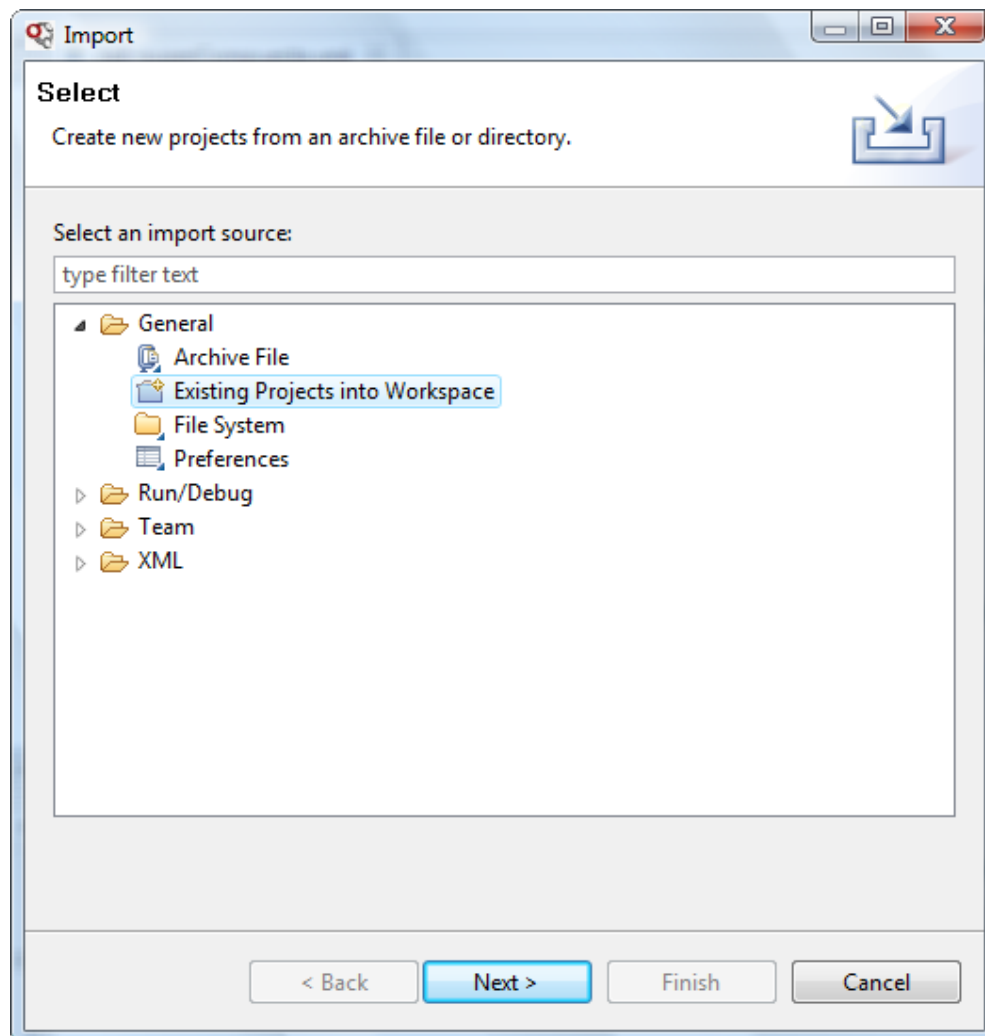
- ❖ Importación de un proyecto para crear archivos de configuración (sólo la primera vez).
- ❖ Configuración de escenarios de prueba en el proyecto.
- ❖ Ejecución de un escenario de prueba.
- ❖ Visualización del resultado de una ejecución.
- ❖ Generación del reporte de la ejecución.

## A.2. EJECUCIÓN DEL SCEMS

SCEMS organiza un conjunto de escenarios de prueba en unidades llamadas proyecto, visibles a través de un navegador. La forma recomendada de crear un proyecto, es importando el que provee SCEMS con algunos escenarios de prueba de ejemplo. Para ello se debe hacer *clic* derecho dentro del *SCEMS Navigator* y seleccionar la opción “Import...”.

Luego, de las opciones disponibles, se debe seleccionar “Existing Projects into Workspace”, como se muestra en la Figura A.2.

Luego se debe presionar “Next” y en la siguiente pantalla, hacer *clic* en “Browse...” y seleccionar dentro del directorio de instalación de SCEMS la carpeta “/projects/Produccion”. Una vez seleccionado este proyecto, se debe hacer *clic* en la opción “Finish”. En la Figura A.3 se puede observar lo mencionado.



**Figura A.2: Importación de proyectos**

De este modo, se podrá visualizar el proyecto “Produccion” en el SCEMS *Navigator*. Cada proyecto contiene tres tipos de archivos, organizados en tres carpetas:

- ❖ *Agencias*: Contiene los archivos que describen cada configuración de agencia. Puede a su vez tener sub-carpetas, para permitir una mejor organización de los casos de prueba.
- ❖ *Eventos*: Contiene los archivos que describen los eventos a generar para cada configuración de agencia. Debe existir, dentro de esta carpeta, un archivo por cada uno de los contenidos en la carpeta *Agencias*, respetando la misma estructura de sub-carpetas si las hubiera.
- ❖ *Resultados*: Contiene los archivos con la información resultante de cada ejecución de algún escenario de prueba realizada por *SCEMS*.

Para ejecutar un caso de prueba, primeramente se debe seleccionar en el navegador el archivo de configuración de agencia, como se muestra en la Figura A.4.

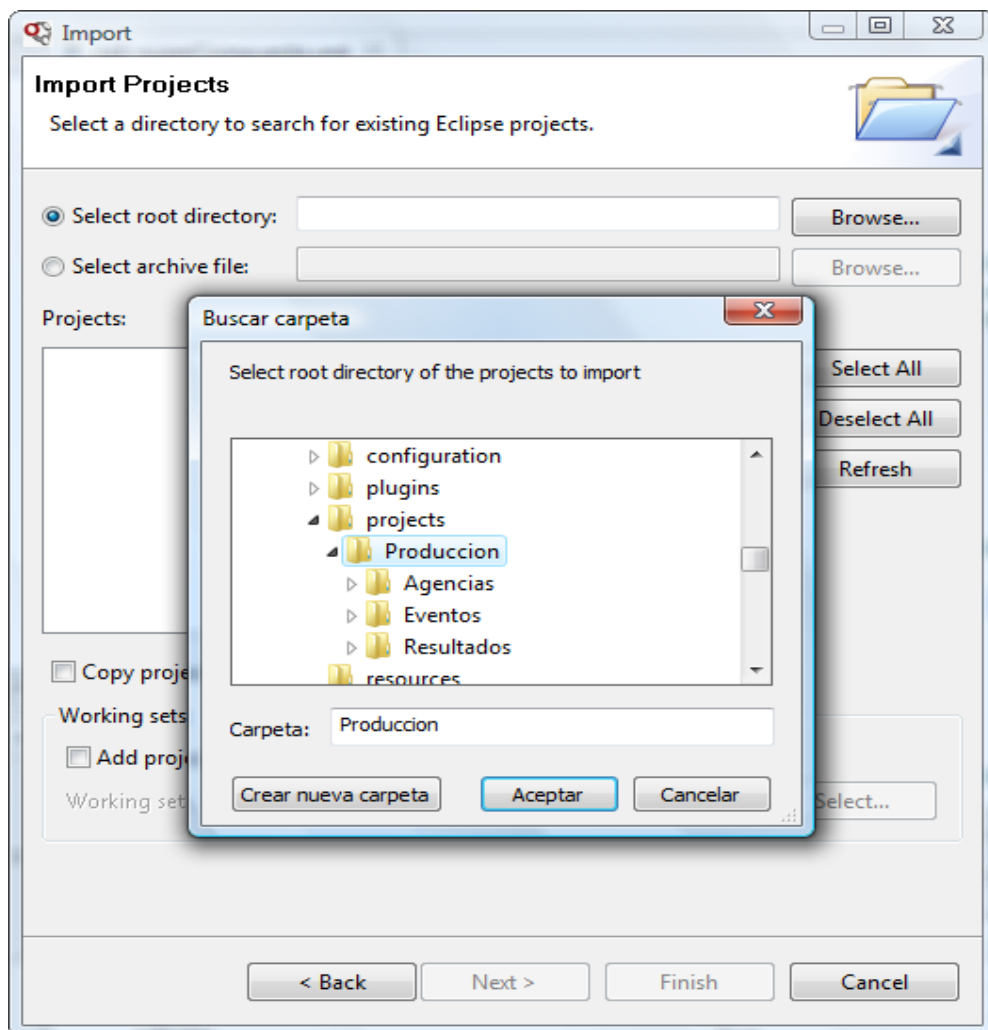


Figura A.3: Selección del proyecto “Produccion”

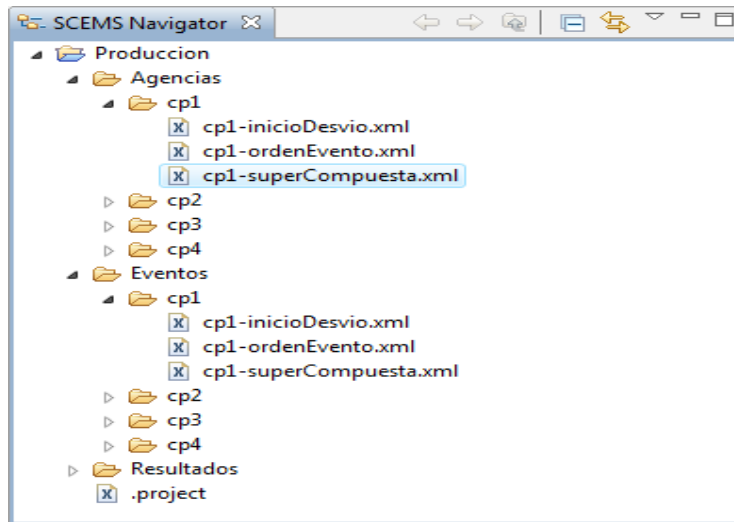


Figura A.4: SCEMS Navigator

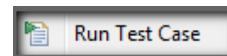
### A.3. CONFIGURACIÓN DE ESCENARIOS DE PRUEBA

La configuración de escenarios se realiza en los archivos XML de agencias y eventos. Para abrir uno de estos archivos, se debe hacer *doble clic* sobre ellos. Esta acción abre un editor para el archivo en el área de trabajo.

La aplicación verifica la sintaxis de los archivos, marcando los errores, si existen. Además, presionando **Ctrl + Espacio**, la aplicación brindará las opciones disponibles para auto-completar el archivo, simplificando de esta forma la configuración de escenarios.

### A.4. EJECUCIÓN DE UN ESCENARIO DE PRUEBA

Para ejecutar un caso de prueba, se debe hacer *clic* derecho sobre el archivo de configuración de agencia y elegir la opción “Run Test Case”.




También puede realizarse mediante la barra de herramientas.



Esta acción hará que se dispare el motor de ejecución de SCEMS, que procesa el archivo de configuración de agencia, levantando la plataforma JADE, creando los agentes necesarios para el escenario a ejecutar y configurándolos con los valores



correspondientes. Luego se procesa el archivo de eventos, creando el agente de eventos correspondiente y configurándolo para que disparen los eventos programados.

El motor de SCEMS permanecerá en ejecución (la agencia continuará corriendo) hasta que se presione el botón “Finish Test Case”.  Esta acción hará que se detenga la plataforma JADE y que los agentes dejen de ejecutarse.

Una vez finalizado el caso de prueba, SCEMS almacena en un archivo con formato XML el resultado de la ejecución.

## A.5. RESULTADO DE UNA EJECUCIÓN

Al finalizar la ejecución, el motor de SCEMS genera una salida en formato XML, conteniendo la información generada e intercambiada por los agentes durante las negociaciones realizadas y el estado de cada agente antes y después de cada solución implementada.

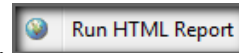
Esta salida se almacena en el mismo proyecto, que contiene los archivos de configuración de agencia y de eventos, bajo la carpeta “Resultados” y utilizando la misma estructura de subdirectorios. El nombre del archivo creado coincide con el nombre de los archivos de configuración de agencia y de eventos, pero incluye también la fecha y la hora en que el caso de prueba fue ejecutado. Por ejemplo, para un archivo de configuración de agencia “Agencias/cp1/cp1-superCompuesta.xml” y archivo de definición de eventos “Eventos/cp1/cp1-superCompuesta.xml”, se crea un archivo de resultados “Resultados/cp1/2009-12-31-23-59-30-cp1-superCompuesta.xml”, para el caso en que el escenario de prueba se ejecutó el 31 de diciembre de 2009 a las 23:59:30.

La estructura de este archivo XML es el punto de partida para el desarrollo y uso de otras aplicaciones, que realicen un análisis del contenido y provean estadísticas e indicadores útiles a los especialistas que deben evaluar el comportamiento de diferentes estrategias de negociación ante diferentes escenarios. Con esto se alcanza el objetivo primario de SCEMS, como herramienta de simulación y de soporte de decisión en el campo de la investigación para la mejora de la gestión de eventos.

## A.6. GENERACIÓN DEL REPORTE DE EJECUCIÓN

Si bien el archivo XML de resultados de ejecución contiene toda la información útil para el procesamiento y análisis automatizado de cada escenario de prueba, resulta conveniente poder hacer un análisis sencillo en el mismo momento en que se ha ejecutado un caso de prueba.

Para generar y visualizar un reporte, se debe hacer *clic* derecho sobre el archivo de resultados de ejecución y elegir la opción “Run HTML Report”.



Esta acción disparará la herramienta de generación de reportes y se abrirá en una solapa interna a SCEMS un navegador Web mostrando el HTML generado. La Figura A.5 muestra la distribución de la ventana principal de SCEMS mientras se visualiza un reporte.

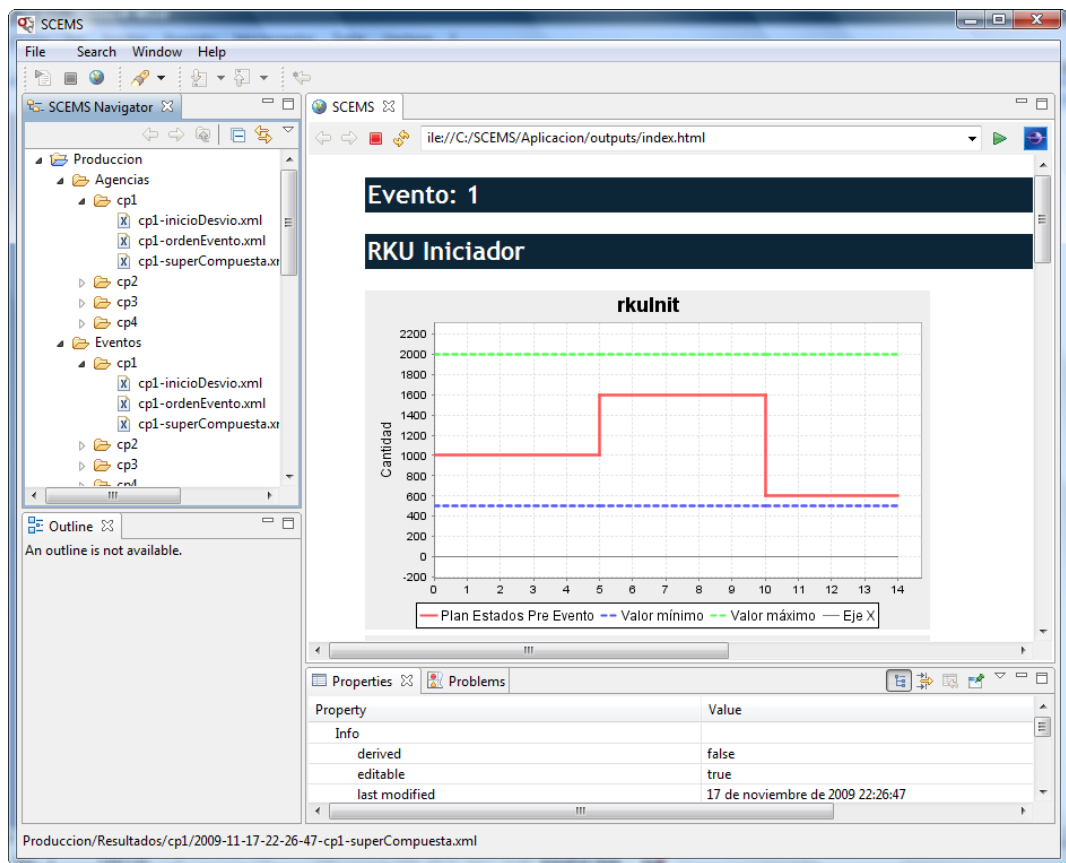


Figura A.5: Visualización de reporte HTML

## A.7. CONFIGURACIÓN DE LOS AGENTES

Para ejecutar una simulación con SCEMS se debe indicar el escenario correspondiente. Un Sistema de Programación genera un programa de abastecimiento en determinado formato. La entrada para una simulación con SCEMS es un programa de abastecimiento sobre el que se simularán diversos eventos.

La existencia de distintos tipos de Sistemas de Programación, y la heterogeneidad de los formatos de los programas de abastecimiento producidos por éstos, hacen necesaria la definición de una interfaz común que sirva como entrada para los escenarios de SCEMS. Se optó por utilizar XML para proveer estos datos por ser un lenguaje claro, estándar y ampliamente difundido para el intercambio de información entre aplicaciones.

A continuación se describirá cómo definir la configuración de la agencia en un archivo XML, lo cual puede realizarse desde la interfaz gráfica. Luego este archivo será utilizado desde la misma interfaz gráfica para realizar simulaciones.

---

### A.7.1. CONFIGURACIÓN DE LA AGENCIA

El elemento raíz de un archivo de configuración de agencia es un *tag* `<scems:scem>`. Este tiene un atributo:

- `xmlns:scems="scems:agentes"` Es una declaración de *namespace*, utilizada por el *parser* XML. El *tag* `<scems:scem>` debe contener dos *tags* hijos. Uno de ellos es un *tag* `<rkus>`, que contendrá una lista de *tags* `<rku>`, donde cada uno de ellos describe a un agente RKU de la agencia. El segundo es un *tag* `<sps>`, que contendrá una lista de *tags* `<sp>`, donde cada uno de ellos describe a un agente SP de la agencia.

La estructura general del archivo es la siguiente (Figura A.6):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<scems:scem xmlns:scems="scems:agentes">
  <rkus>
    <rku>
      ...
    </rku>
```

```

<rku>
    . . .
  </rku>
</rkus>
<sps>
  <sp>
    . . .
  </sp>
  <sp>
    . . .
  </sp>
</sps>
</scem>

```

**Figura A.6: Extracto del archivo general de configuración**

---

## A.7.2. CONFIGURACIÓN DE LOS AGENTES RKU

La configuración inicial de cada uno de los agentes RKU se especifica mediante un *tag* `<rku>`, que contiene otros seis *tags*, siguiendo la estructura que se muestra en la Figura A.7:

```

<rku>
  <nombre>rku1</nombre>
  <ae>ae</ae>
  <configuracion>
    <valorMinimoVariable>500</valorMinimoVariable>
    <valorMaximoVariable>2000</valorMaximoVariable>
    <nivelServicioObjetivo>1</nivelServicioObjetivo>
  </configuracion>
  <ordenes>
    <orden>...</orden>

```

```
<orden>...</orden>
</ordenes>
<estrategia>inicioDesvio</estrategia>
<calculadorUtilidad>default</calculadorUtilidad>
</rku>
```

**Figura A.7: Extracto del archivo de configuración de un agente RKU**

Semántica de la información contenida en cada uno de estos *tags*:

1. nombre: El nombre del agente RKU.
2. ae: El nombre del agente AE del cual el agente RKU deberá esperar mensajes. Este nombre debe coincidir con el valor del *tag* <nombre> del archivo XML de eventos a disparar, y que corresponde al nombre del agente AE que disparará los eventos.
3. configuracion: La configuración del agente RKU. Se especifica mediante tres valores, contenidos en tres *tags*:
  - a. valorMinimoVariable: El valor mínimo permitido para la variable monitoreada por el agente RKU. Debe ser un valor decimal.
  - b. valorMaximoVariable: El valor máximo permitido para la variable monitoreada por el agente RKU. Debe ser un valor decimal.
  - c. nivelServicioObjetivo: El nivel de servicio objetivo para el agente RKU. Debe ser un valor decimal.
4. ordenes: La lista de órdenes de entrada/salida del agente RKU. Contiene varios *tags* <orden>, donde cada uno de ellos describe una orden.
5. estrategia: El nombre de la estrategia de negociación que utilizará el agente RKU. Este nombre debe coincidir con el nombre lógico de alguna estrategia de negociación (simple o compuesta). Es decir, el nombre debe coincidir con el valor del atributo nombre de algún *tag*

<estrategia-simple> o <estrategia-compuesta> del archivo XML donde se especifican las estrategias de negociación.

6. `calculadorUtilidad`: El nombre del calculador de utilidad que utilizará el agente RKU. Este nombre debe coincidir con el nombre lógico de algún calculador de utilidad. Es decir, el nombre debe coincidir con el valor del atributo nombre de algún *tag* <calculador-utilidad> del archivo XML donde se especifican los calculadores de utilidad.

La lista de órdenes del agente RKU está contenida en un *tag* <ordenes>, el cual contiene una lista de *tags* <orden>, donde cada uno de ellos describe una orden. Un *tag* <orden> debe contener otros seis *tags*, siguiendo la estructura detallada en la Figura A.8:

```
<orden>
<numeroOrden>11</numeroOrden>
<inicio>100</inicio>
<duracion>15</duracion>
<cantidad>-1000</cantidad>
<tipoCambioVariable>1</tipoCambioVariable>
<nombreSp>sp14</nombreSp>
</orden>
```

**Figura A.8: Detalle en la especificación de una orden para el agente RKU**

Semántica de la información contenida en cada uno de estos *tags*:

1. `numeroOrden`: El número de la orden. Debe ser un valor entero positivo. Su valor no puede coincidir con el de otro *tag* <numeroOrden> dentro del *tag* <ordenes> de un mismo *tag* <rku>.

2. `inicio`: El momento de inicio del procesamiento de la orden. Debe ser un valor entero positivo.

3. `duracion`: La duración del procesamiento de la orden. Debe ser un valor entero positivo.

4. `cantidad`: La cantidad que entra o sale del recurso representado por el agente RKU debido a la orden. Debe ser un valor decimal. Un valor positivo implica una orden de entrada (el valor de la variable monitoreada se ve incrementado por la

orden). Un valor negativo implica una orden de salida (el valor de la variable monitoreada se ve disminuido por la orden).

5. `tipoCambioVariable`: El tipo de cambio de la variable de estado del agente RKU. Describe la forma en que la variable monitoreada va cambiando durante el procesamiento de la orden. Admite tres valores enteros:

- 1: Representa que el tipo de cambio de la entrada/salida se da completa e instantáneamente al principio del procesamiento.
- 2: Representa que el tipo de cambio de la entrada/salida se da completa e instantáneamente al final del procesamiento.
- 3: Representa que el tipo de cambio de la entrada/salida se da en forma gradual y constante durante el procesamiento.

6. `nombreSp`: El nombre del agente SP asociado a la orden.

---

### A.7.3. CONFIGURACIÓN DE LOS AGENTES SP

La configuración inicial de cada agente SP se especifica mediante un *tag* `<sp>`, el cual debe contener otros siete *tags*, siguiendo la estructura detallada en la Figura A.9:

```
<sp>
<nombre>sp14</nombre>
<idActividad>13</idActividad>
<inicio>100</inicio>
<duracion>25</duracion>
<extension>250</extension>
<unificadorPropuestas>default</unificadorPropuestas>
<items>
<item>...</item>
<item>...</item>
</items>
</sp>
```

**Figura A.9: Detalle de configuración de los agentes SP**

Semántica de la información contenida en cada uno de estos tags:

1. `nombre`: El nombre del agente SP.
2. `idActividad`: El identificador de la actividad representada por el agente SP. Debe ser un valor entero. Su valor no puede coincidir con el de otro *tag* `<idActividad>` dentro del archivo.

3. *inicio*: El momento de inicio del procesamiento de la actividad. Debe ser un valor entero positivo.

4. *duracion*: La duración del procesamiento de la actividad. Debe ser un valor entero positivo.

5. *extension*: La cantidad de veces que se ejecuta el proceso, descrito por el plan de actividades, durante la actividad.

6. *unificadorPropuestas*: El nombre del unificador de propuestas que utilizará el agente SP. Este nombre debe coincidir con el nombre lógico de algún unificador de propuestas. Es decir, el nombre debe coincidir con el valor del atributo *nombre* de algún *tag* `<unificador-propuestas>` del archivo XML donde se especifican los unificadores de propuestas.

7. *items*: El conjunto de ítems del plan de actividades. Contiene varios *tags* `<item>`, donde cada uno de ellos describe un ítem.

El plan de actividades describe la relación entrada/salida en cada ejecución del proceso para cada agente RKU asociado a la actividad representada por el agente SP. Este conjunto está contenido en un *tag* `<items>`, el cual contiene una lista de *tags* `<item>`, donde cada uno de ellos describe un ítem. Un *tag* `<item>` debe contener otros tres *tags*, siguiendo la estructura representada en la Figura A.10:

```
<item>
<nombreRku>rku1</nombreRku>
<numeroOrdenRKU>11</numeroOrdenRKU>
<variacionCantidadRKU>-4</variacionCantidadRKU>
</item>
```

**Figura A.10: Detalle para la configuración del plan de actividades**

Semántica de la información contenida en cada uno de estos *tags*:

1. *nombreRku*: El nombre del agente RKU.
2. *numeroOrdenRKU*: El número de la orden en el agente RKU. Debe ser un valor entero positivo.

*variacionCantidadRKU*: La relación entrada/salida en cada ejecución del proceso para cada agente RKU. Debe ser un valor entero. Un valor positivo implica una entrada (el valor de la variable monitoreada por el agente RKU se ve incrementado por



la ejecución del proceso). Un valor negativo implica una salida (el valor de la variable monitoreada por el agente RKU se ve disminuido por la ejecución del proceso).

#### A.7.4. DEFINICIÓN DE LA SECUENCIA DE EVENTOS

Para ejecutar una simulación con SCEMS se debe indicar el escenario correspondiente. Para ello se utiliza un archivo XML para definir la secuencia de eventos que se desean simular sobre un programa de abastecimiento.

El elemento raíz de un archivo de eventos a disparar es un *tag* `<eventos:ae>` y el mismo contiene la configuración del agente AE. El *tag* `<eventos:ae>` tiene un atributo:

- `xmlns:eventos="eventos:eventos"`

Es una declaración de *namespace*, utilizada por el *parser* XML.

El *tag* `<eventos:ae>` debe contener dos *tags* hijos. Uno de ellos es un *tag* `<nombre>`, que define el nombre del agente de eventos. El segundo es un *tag* `<eventos>`, que contendrá una lista de *tags* `<evento>`, donde cada uno de ellos describe un evento a disparar, cuya estructura se presenta en al Figura A.11.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ae xmlns:eventos="eventos:eventos">
<nombre>ae</nombre>
<eventos>
<evento>
...
</evento>
<evento>
...
</evento>
</eventos>
</ae>
```

**Figura A.11: Estructura general del archivo de configuración de los eventos**

Cada evento a disparar se especifica mediante un *tag* `<evento>`, el cual debe contener otros seis *tags*, siguiendo la estructura de la Figura A.12.

Semántica de la información contenida en cada uno de estos *tags*:

1. `idEvento`: El identificador del evento. Debe ser un valor entero. Su valor no puede coincidir con el de otro *tag* `<idEvento>` dentro del archivo.

2. `receiver`: El nombre del agente RKU que debe recibir el evento. Debe referenciar a un agente RKU existente. Es decir, su valor debe coincidir con el del `tag <nombre>` de algún `tag <rku>` del archivo de configuración de agencia.

3. `numeroOrden`: El número de orden en el agente RKU que se modifica con el evento. Debe ser un valor entero positivo. Este número debe referenciar a una orden existente en el agente RKU. Es decir, su valor debe coincidir con el valor del `tag <numeroOrden>` contenido en alguno de los `tags <orden>` del agente RKU correspondiente en el archivo de configuración de agencia.

4. `variacionFechaInicio`: La variación en la fecha de inicio de la orden. Debe ser un valor entero. Un valor positivo indica un adelanto en el tiempo de la orden. Un valor negativo indica un retraso. El valor absoluto indica la cantidad de unidades de tiempo en que variará el inicio de la orden.

5. `variacionCantidad`: La variación en la cantidad de la orden. Debe ser un valor decimal. Si se trata de una orden de entrada un valor negativo indica una disminución en la cantidad de la orden y un valor positivo un incremento. Si se trata de una orden de salida un valor negativo indica un incremento en la cantidad de la orden y un valor positivo una disminución.

6. `milisegundosDemoraDisparo`: La cantidad de milisegundos que debe esperarse para disparar el evento, desde el momento en que comienza la simulación. Debe ser un valor entero positivo. Se utiliza para indicar el momento en que cada evento debe ser notificado al agente RKU correspondiente.

```
<evento>
<idEvento>3</idEvento>
<receiver>rkul</receiver>
<numeroOrden>11</numeroOrden>
<variacionFechaInicio>0</variacionFechaInicio>
<variacionCantidad>500</variacionCantidad>
<milisegundosDemoraDisparo>2000</milisegundosDemoraDisparo>
</evento>
```

**Figura A.12: Estructura para la especificación de eventos**

## REFERENCIAS

- [**Adhitya et al, 2007**] Adhitya, A., Srinivasan, R. and Karimi I.A., 2007. A model-based rescheduling framework for managing abnormal supply chain events. *Computers and Chemical Engineering*, 31(5), pp 496-518.
- [**Alvarenga and Schoenthaler, 2003**] Alvarenga C. and Schoenthaler R. *A New Take on Supply Chain Event Management*. In *Supply Chain Management Review*. March/April 2003.
- [**Anosike and Zhang, 2002**] Anosike, A I and Zhang, Z, "An Agent-Oriented Modelling Approach for Agile Manufacturing", The 3rd International Symposium on Multi-Agent Systems Large Complex Systems, and E-Business (MALCEB 2002), Erfurt, Thuringia, Germany, 8-10 October 2002.
- [**Bartschi, 1996**] Bartschi, M. A Genetic Algorithm for Resource-Constrained Scheduling. Ph.D. Thesis. MIT. 1996.
- [**Bauer et al, 2001**] Bauer, B., Müller J. And Odell J. "A Formalism for Specifying Multiagent Interaction", In *Agent-Oriented Software Engineering*, Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, pp. 91-103, 2001.
- [**Bearzotti et al, 2011**] Bearzotti, Lorena A., Enrique Salomone, Omar J. Chiotti. An autonomous multi-agent approach to supply chain event management. *International Journal of Production Economics*, Available online 12 September 2011, ISSN 0925-5273, 10.1016/j.ijpe.2011.08.023. (<http://www.sciencedirect.com/science/article/pii/S092552731100377X>).
- [**Bellifemine et al, 1999**] Bellifemine F., Poggi, A. and Rimassa, G., 1999. JADE – A FIPA-compliant agent framework. CSELT Internal Technical Report. Part of this report has been also published in *Proceedings of PAAM'99*, London, April, 97-108.
- [**Bodendorf and Zimmermann, 2005**] Bodendorf, F., and Zimmermann, R., 2005. Proactive supply-chain event management with agent technology. *International Journal Electronic Commerce*, 9(4), 58-89.
- [**Busetta et al, 1999**] Busetta, P., Ralph Ronnquist, Andrew Hodgson and Andrew Lucas, 1999. JACK Intelligent Agent – Components for Intelligent Agents in Java. In *AgentLink News Issue 2*.
- [**Bussmann et al, 2004**] Bussmann, S.; Jennings, N. R. and Wooldridge, M., 2004. Multiagent systems for manufacturing control: A design methodology. T. Ishida, N.R. Jennings, K. Sycara (Series Eds.). *Springer Series on Agent Technology*. Springer-Verlag, Berlin Heidelberg, Germany.
- [**Cauvin et al, 2009**] Cauvin, A.C.A., Ferrarini, A.F.A. and Tranvouez, E.T.E., 2009. Disruption management in distributed enterprises: A multi-agent modelling and simulation of cooperative recovery behaviours. In *International Journal of Production Economics*, 122(1), 429-439.
- [**Christopher, 2005**] Christopher, M., 2005. *Logistics and Supply Chain Management. Creating Value-Adding Networks*. Prentice Hall, New York, USA.
- [**FIPA-OS, 2006**] <http://fipa-os.sourceforge.net/index.htm> visitado en Marzo 2006.
- [**Guo and Zhang, 2009**] Guo, Q. and Zhang, M. 2009. A novel approach for multi-agent-based intelligent manufacturing system. In *Information Sciences*, 179(18), 3079-3090.

- [Handfield and Nichols, 1999]** Handfield R.B. and Nichols E.L. Introduction to Supply Chain Management. Prentice Hall. 1999
- [Hoffmann et al, 1999]** Hoffmann, O., Deschner, D., Reinheimer, S. and Bodendorf, F., 1999. Agent-supported information retrieval in the logistic chain. In proceeding of the 32<sup>nd</sup> Hawaii International Conference on System Sciences, Maui, USA.
- [Howden et al, 2001]** Howden, N., Ralph Ronnquist and Andrew Lucas, 2001. JACK Intelligent agents – Summary of an agent infrastructure. In Proceedings of the 5th International Conference on Autonomous Agents.
- [Kärkkäinen et al, 2003]** Kärkkäinen, M., Främling, K. And Ala-Ri, M., 2003. Integrating material and information flows using a distributed peer-to-peer information system. In Collaborative System for Production Management, 305-319, Kuwer Academic Publishers, Boston, USA.
- [Kleindorfer and Saad, 2005]** Kleindorfer, P.R. and Saad G.H., Managing Disruption Risks in Supply Chains, Production and Operations Management, vol. 14, no.1, pp.53–68, 2005.
- [Knickle and Kemmeler, 2002]** Knickle, K. and Kemmeler, J., 2002. Supply chain event management in the field success with visibility. AMR Research, Boston, USA.
- [Landeghem and Vanmaele, 2002]** Landeghem V., and H. Vanmaele, Robust planning: a new paradigm for demand chain planning, Journal of Operations Management, vol. 20, pp.769–783, 2002.
- [Lee et al, 1997]** Lee, H. L. , V. Padmanabhan, and S. Whang, The Bullwhip Effect in Supply Chains, Sloan Management Review, vol.38, no.3, pp.93-102, 1997.
- [Leitao, 2004]** Leitao, P., 2004. An agile and adaptive holonic architecture for manufacturing control. Ph.D. Dissertation. Department of Electrotechnical Engineering Polytechnic Institute of Braganca, University of Porto, Portugal. Online: <http://www.ipb.pt/pleitao>
- [Mabert et al, 2002]** Mabert, Vincent A., Soni, Ashok, Venkataramanan, M.A. Sistemas ERP: mitos comunes "versus" realidad evolutiva. Revista Harvard Deusto Business Review, 2002 ENE-FEB (106) pag 74-82.
- [Mas, 2005]** Mas, A. Agente Software y Sistemas Multiagente. Conceptos, Arquitectura y Aplicaciones. Pearson Educación, S.A. Madrid, 2005.
- [Masing, 2003]** Masing, N., 2003. Supply chain event management as strategic perspective – market study: SCEM software performance in the european market. Master Thesis. Université du Québec en Outaouasis.
- [Miles and Huberman, 1994]** Miles, M.B., and Huberman, A.M., 1994. Qualitative Data Analysis. A source Books of New Methods. Sage, Newbury Park, California, USA.
- [Monostori et al, 1998]** Monostori L, Szelke, E. and Kadar, B., 1998. Management of changes and disturbances in manufacturing systems. Annual Reviews in Control, 22, 85-97.
- [Oke and Gopalakrishnan, 2009]** Oke, A., and Gopalakrishnan, M., 2009. Managing disruptions in supply chains: A case study of a retail supply chain. International Journal of Production Economics, 118(1), 168-174.
- [Oprea, 2003]** Oprea, M., 2003. Coordination in an agent-based virtual enterprise. In Studies in Informatics and Control, 12(3), 215-225.
- [Ottino, 2003]** Ottino, J.M. Complex Systems in *AICHE Journal*. Journal February 2003 Vol 49, Nro 2.

- [Pereira, 2009]** Pereira, J., 2009. The new supply chain's frontier: Information management. In *International Journal of Information Management*, 29(5), 372-379.
- [Radjou et al, 2002]** Radjou, N.; L.M. Orlov, and T. Nakashima, *Adapting To Supply Network Change*, in *The TechStrategy Report*, F. Research, Editor, 2002.
- [Rao and Georgeff, 1995]** Rao, A. and Georgeff, M., 1995. BDI Agents: From theory to practice. *Proceedings of the First International Conference on Multi-Agent System*. 12-14 June, San Francisco, USA.
- [Schoenthaler and Alvarenga, 2003]** Schoenthaler, R and Alvarenga, C., 2003. A new take on supply chain event management. In *Supply Chain Management Review*. March/April, 29-35.
- [Simchi-Levi et al, 1999]** Simchi-Levi, D., Kamanisky P., and Simchi-Levi E., 1999. *Designing and Managing the Supply Chain*. Mc Graw Hill, USA.
- [Smith, 1980]** Smith, R. G., 1980. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29 (12), 1104-1113.
- [Speyerer and Zeller, 2004]** Speyerer, J. and Zeller, A., 2004. Managing supply networks: Symptom recognition and diagnostic analysis with web services. *Proceedings of the 37<sup>th</sup> Hawaii International Conference on System Sciences*. Hawaii, USA.
- [Swaminathan et al, 1998]** Swaminathan, J., Smith, S and Sadeh, N., 1998. Modeling supply chain dynamics: A multiagent approach. *Decision Sciences*, 29(3), 607-632.
- [Szirbik et al, 2000]** Szirbik, N.B., Wortmann, J.C., Hammer, D.K., Goosenaerts, J.B. M. and Aerts, A.T.M., 2000. Mediating negotiations in a virtual enterprise via mobile agents. In *Proceedings of the Academia/Industry Working Conference on Research Challenges (IEEE Computer Society Eds.)*, Buffalo, New York, USA, pp 237-242.
- [Teuteberg and Schreber, 2005]** Teuteberg F and Schreber, D., 2005. Mobile computing and auto-ID technologies in supply chain event management – An agent-based approach. In *Proceedings of the Thirteenth European Conference on Information Systems*, 21-25 September, Regensburg, Germany.
- [Villarreal et al, 2003a]** Villarreal, P. D.; Caliusco, M. L.; Zucchini, D.; Arredondo, F.; Zanel, C.; Galli, M. R.; y Chiotti, O. *Integrated Production Planning and Control in a Collaborative Partner-to-Partner Relationship*. *Managing e-Business in the 21st Century*. Ed. S. Sharma & J. Gupta. Heidelberg Press, pp. 91-110. Australia, 2003.
- [Villarreal et al, 2003b]** Villarreal, P.; Caliusco, M. L.; Galli, M. R.; Salomone, E.; y Chiotti, O. *Decentralized Process Management for Inter-Enterprise Collaboration*. *Vision, Special Issue on Supply Chain Management*, Vol 7, pp. 69-79. Editor: B. S. Sahay, Management Development Institute, Gurgaon, India, 2003.
- [Wellman, 1994]** Wellman, M. A Computational Market for distributed Configuration Design. In *Proc. of the 12<sup>th</sup> National Conference on Artificial Intelligence (AAAI 94)*, pager 401- 407. Seattle, WA. July 1994. AAAI, AAAI Press.
- [Wellman, 1996]** Wellman, M.P. *Market-Oriented Programming: Some Early Lessons*. S. ed. Clearwater. *Market-based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, Singapore. 1996.
- [Wurman, 1999]** Wurman, P. *Market Structure and Multidimensional Auction Design for Computational Economies*. PhD Thesis, University of Michigan. 1999.
- [Wurman and Wellman, 1999]** Wurman, P. and Wellman, M. *Equilibrium Prices in Bundle Auctions*. Santa Fe Institute Working Paper, 1999.

**[Yin, 1994]** Yin, R. K., 1994. Case Study Research: Design and Method. second ed. Sage, Thousand Oaks, California, USA.

**[Zhou and Venkatesh, 1999]** MengChu Zhou and Kurapati Venkatesh. Modeling, simulation and control of flexible manufacturing systems: a Petri Net Approach. Series in Intelligent Control and Intelligent Automatisation; vol 6. ISBN 981023029X. Editor in Charge: Fei-Yue Wang.

**[Zimmermann, 2006]** Zimmermann, R., 2006. Agent-based Supply Network Event Management. Whitestein Series in Software Agent Technologies, Eds: Marius Walliser, Stefan Brantschen, Monique Calisti y Thomas Hempfling. Birkhäuser Verlag, Switzerland.