TEORÍA DE ACTIVIDAD PARA EL DESARROLLO DE SISTEMAS MULTI-AGENTE

Memoria que presenta para optar al grado de
Doctor en Informática

Rubén Fuentes Fernández

Dirigida por los profesores

Jorge J. Gómez Sanz
Juan Pavón Mestras

Departamento de Sistemas Informáticos y Programación
Facultad de Informática
Universidad Complutense de Madrid
Septiembre 2004

Y así sucumbió el lobo estepario en su independencia. […] Porque ya resultaba que la soledad y la independencia no eran su afán y su objetivo, eran su destino y su condenación, que su deseo mágico se había cumplido y ya no era posible retirarlo […]

Oh, lo comprendí todo; comprendí a Pablo, comprendí a Mozart, oí en alguna parte detrás de mi su risa terrible; sabía que estaban en mi bolsillo todas las cien mil figuras del juego de la vida: aniquilado, barruntaba su significación; tenía el propósito de empezar otra vez el juego, de gustar sus tormentos otra vez, de estremecerme de nuevo y recorrer una y muchas veces más el infierno de mi interior.

El Lobo Estepario, Hermann Hesse

Agradecimientos

Siempre pensé que escribir una tesis debía ser parecido a realizar un largo viaje, una oportunidad de conocer personas y también a nosotros mismos. Al final, con el trabajo ya realizado, llega el momento de recapitular acerca de esas personas que encontramos en nuestro viaje, las que siguen y las que se fueron. A todos, gracias, porque en cierta medida estoy escribiendo ahora estos agradecimientos por vosotros.

Quiero empezar por mis tutores, Jorge Gómez y Juan Pavón. Llegué a trabajar con ellos un poco por casualidad. A lo largo de este tiempo, Jorge y Juan han sido mis guías y amigos. Juntos hemos compartido divagaciones, planes, inquietudes y algún que otro dolor de cabeza. Si hoy existe esta tesis es gracias a vosotros. Estoy seguro de que ya estabais deseando terminarla.

También quiero agradecer a la doctora Lorna Uden, de la Universidad de Staffordshire en el Reino Unido, nuestras largas discusiones sobre la Teoría de Actividad durante mi estancia allí. En buena medida, a ella se debe que esta tesis hable de una teoría psicológica para Ingeniería del Software.

Quiero además recordar a los responsables de que hoy en día esté trabajando en la Universidad y me haya dedicado a escribir una tesis, Fernando Rubio y Jorge Gómez otra vez. Fuimos compañeros de estudios y en ninguno de nuestros encuentros dejaron de insistirme en que volviera a la Facultad… Al final lo consiguieron. A los dos, gracias porque hoy mi trabajo me hace mucho más feliz.

Por supuesto, no quiero dejar de acordarme de mis compañeros de despacho en la Facultad, Alfredo Fernández-Valmayor y José Luis Sierra. Gracias a los dos por haberme aguantado y haber compartido conmigo vuestro saber.

No se me pueden olvidar tampoco mis compañeros del Departamento, especialmente aquellos a los que no he nombrado todavía, que han hecho más agradable mi enclaustramiento en la Facultad y a veces conseguían que abandonase el ordenador: Antonio Vaquero, Carmen, Luis, Baltasar, Mercedes, Antonio Navarro, Belén, Eva y Antonio Pareja. Gracias especiales a Pilar por tantos cafés en los que hemos estado arreglando el mundo.

Dejando ya el mundo académico, quiero acordarme de mis amigos que todo este tiempo tuvieron la fe que a veces me falto a mi. Alaitz, Alberto, Alejandro, Alicia, Antonio, Bego, Berni, Eduardo, Dani, Guillermo, Luisme, Paco, Raúl, Santi y Xavier.

A mis amigas Beatriz y Laura que han sido mis confidentes y psiquiatras durante estos tiempos de locura. Gracias a las dos por ser tan especiales. Nunca os agradeceré lo bastante vuestra sinceridad, buen juicio y ánimo. Gracias también por tu inestimable paciencia con mi Inglés Laura.

Finalmente, y aunque suene tópico, nada de esto hubiera sido posible sin el amor y apoyo de mi familia. Finalmente hemos llegado al final de la odisea. Gracias a todos de corazón por haber estado siempre conmigo. Os quiero.

Este trabajo de investigación ha sido financiado parcialmente por el Ministerio Español de Ciencia y Tecnología con los proyectos *Agentes software para tratamiento de información hipermedia (TIC 2000-0737-C03-02)* e *INGENiería de Agentes Software (INGENIAS) en múltiples plataformas (TIC 2002-04516-C03-03)* y por la Unión Europea con los proyectos *DEMOS: Delphi Mediation Online System (IST 1999-20530)* y *AgentLink III, the European Co-ordination Action for Agent-Based Computing (IST 002006)*.

Abstract of the Thesis

Methodologies of the Agent Oriented Software Engineering (AOSE) deal with the construction of Multi-Agent Systems (MAS). Two relevant features of MAS are their social and intentional features. AOSE currently faces these features by means of conventional development techniques that have their origin in other paradigms, or specific techniques focused on isolated components of those features. However, these solutions have some drawbacks. The first techniques are not very suitable for social and intentional features, as they are different from traditional artefacts in Software Engineering. The second ones have limitations when considering features concerning several MAS aspects.

To overcome the problems related to those features, this thesis proposes the use of an integrating framework that manages social and intentional properties in a systemic way, from both individual and social levels. In this work, that framework is inspired by the Activity Theory (AT), which is a well-stablished conceptual and methodological proposal to analyze both individuals and societies in social disciplines. The AT studies properties in human societies that are alike to those that capture our attention in AOSE.

Given these precedents, the aim of this thesis can be stated as follows: to provide a generic method based on AT to describe and apply the social and intentional features of MAS in development. This method is intended for developers. The purpose is not to devise a new methodology, but to provide existing methodologies with helpful techniques that have automated support tools. The use of information about these aspects in this research is thought to be focused in the requirement elicitation and the verification-validation, considering as well the modification of specifications to satisfy properties.

Based upon the knowledge of TA and its techniques, the previous aim has crystallised in the definition of modelling languages and structures to represent AT information, as well as methods to elicit that information, to solve contradictions in MAS specifications, and to translate models between AT and MAs methodologies. Translation is performed using correspondences that specify how to express a structure described with the vocabulary of AT to another one that uses a language from ISOA. Requirement elicitation and contradiction solving are based on libraries of properties which have their roots in the knowledge of AT and are described with the structures for AT.

Throughout the development cycle of any methodology of ISOA, developers get involved on an ongoing process of translation of specifications to AT concepts, elicitation of new knowledge and/or verification of the social and intentional aspects of MAS and back again to translation into the agent methodology.

By these means, developers can work with the system choosing the desired techniques: these from the AOSE or those from AT. At the same time they are supported by the expert knowledge acquired from social sciences in order to work in a holistic manner with problems that are specific to the agent paradigm, such as motivation, organization, leadership, evolution, cognitive skills, learning, law, or trust.

Resumen de la Tesis

Las metodologías de la Ingeniería del Software Orientada a Agentes (ISOA) han abordado los aspectos sociales e intencionales en Sistemas Multi-Agente (SMA) mediante técnicas tradicionales de desarrollo, procedentes de otros paradigmas, o con técnicas específicas centradas en parcelas aisladas de estos aspectos. Sin embargo, este tipo de soluciones no son satisfactorias. La primera opción no está completamente adaptada a esta clase de aspectos, distintos de los tradicionales artefactos de la Ingeniería del Software. La segunda tiene problemas al trabajar con propiedades sociales e intencionales que afectaban a múltiples facetas del SMA.

Para abordar los problemas en el tratamiento de estas propiedades, la presente tesis propone usar un marco integrador que aborde los aspectos sociales e intencionales de los SMAs de forma global, tanto a nivel individual como de comunidad. En el presente trabajo, dicho marco está inspirado por la Teoría de Actividad (TA). Se trata de una propuesta conceptual y metodológica que se usa para el análisis de individuos y sociedades en las disciplinas sociales. La TA estudia propiedades similares a las que centran nuestro interés en la ISOA pero en las sociedades humanas.

Con estos precedentes se puede enunciar con mayor precisión el objetivo global de esta tesis: proporcionar un método genérico basado en la TA para la descripción y uso de los aspectos intencionales y sociales de los SMAs, que puedan llevar a cabo los desarrolladores, que complemente los procesos de desarrollo de SMAs existentes y que cuente con herramientas automatizadas de soporte. El uso de la información sobre estos aspectos en el proceso de desarrollo de SMAs se realizará esencialmente mediante técnicas de captura de requisitos y de verificación y validación, incluyendo también la transformación de las especificaciones para satisfacer propiedades.

A partir del conocimiento y técnicas de la TA, el objetivo anterior se ha realizado con la definición de lenguajes de modelado software y estructuras para representar la información de la TA, así como métodos para la captura de esa información, la resolución de contradicciones en las especificaciones de SMAs y la traducción de modelos entre el vocabulario de la TA y los usados en la ISOA. Los procesos de traducción se basan en la definición de correspondencias entre estructuras expresadas con el vocabulario de la TA y otras expresadas con lenguajes de agentes. Los procesos de captura de información y resolución de contradicciones se fundamentan en librerías de propiedades extraídas del conocimiento de la TA y expresadas con las estructuras desarrolladas para tal efecto.

Sobre el ciclo de desarrollo de las metodologías de la ISOA tiene lugar un trabajo permanente de traducción de las especificaciones a los conceptos propios de la TA, captura de nuevo conocimiento y/o verificación centrada en los aspectos sociales e intencionales del SMA y traducción de nuevo a la metodología de agentes.

Con estos resultados los desarrolladores pueden trabajar sobre sus SMAs con las técnicas que deseen: las de la metodología de la ISOA o las de la TA. Al mismo tiempo cuentan con el apoyo de un cuerpo de conocimiento basado en las ciencias sociales para trabajar sistémicamente con problemas propios del paradigma de agentes, tales como motivación, organización, liderazgo, evolución, capacidades cognitivas, aprendizaje, leyes o confianza.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Este capítulo describe la motivación que ha conducido a la realización de esta tesis, así como los objetivos y el método de trabajo seguido en su realización. El capítulo finaliza describiendo la estructura y el contenido de este documento.

1.1. Contexto

El paradigma orientado a agentes [Newell 1982, Shoham 1993, Genesereth & Ketchpel 1994, Sykara 1998] propone modelar los sistemas software mediante abstracciones de carácter intencional y social. Sus conceptos básicos son los de *agente* y *sistema multi-agente*. Un *agente* es una entidad resolutora de problemas autónoma, que se encuentra situada en un entorno concreto y que exhibe cierta capacidad de comunicación [Maes 1994]. Esta capacidad de comunicación hace que los agentes puedan trabajar juntos en la resolución de problemas, surgiendo entonces la dimensión social del paradigma. Un *Sistema Multi-Agente* (SMA) es un conjunto de agentes que interactúan para resolver problemas que están más allá de sus capacidades individuales [Sykara 1998].

Desarrollar sistemas basados en agentes no es una tarea trivial pese al elevado nivel de abstracción de sus conceptos. Los desarrolladores han de analizar multitud de aspectos tales como la motivación de los agentes, la planificación de las tareas en el sistema, las comunicaciones, el aprendizaje o la movilidad del código. Toda esa información se transforma a lo largo del proceso de desarrollo, desde la fase de captura de requisitos de los clientes hasta la construcción del sistema final. La dirección de este proceso, los procedimientos de obtención y manipulación del conocimiento así como la detección y solución de los posibles errores, siguen necesitando la asistencia de técnicas de desarrollo de software como las ya conocidas bajo otros paradigmas en la Ingeniería del Software.

1.2. Motivación

En la actualidad, la Ingeniería del Software Orientada a Agentes (ISOA) emplea un amplio abanico de técnicas de ayuda para la construcción de SMAs. Estas técnicas incluyen metodologías de desarrollo, resultados de investigación, guías de trabajo y herramientas que son adaptadas para propósitos específicos, como pueden ser la especificación del sistema o la verificación de propiedades. Todas las propuestas revisadas en esta investigación contemplan las componentes sociales e intencionales

que constituyen uno de los pilares esenciales del paradigma de agentes, aunque desde diferentes perspectivas. El trabajo con estas componentes se hace mediante notaciones y procesos que surgen de la experiencia de los investigadores en el desarrollo de SMAs (e.g. las metodologías INGENIAS [Pavón & Gómez-Sanz 2003] y DESIRE [Brazier *et al.* 1997]), de la observación de organizaciones humanas (e.g. la captura de requisitos del sistema a construir en relación con su entorno en i* [Yu 1999] y KAOS [Dardenne *et al.* 1993]), de adaptaciones de investigaciones en otras ciencias, especialmente en las ciencias sociales (e.g. el trabajo con el modelo BDI [Bratman 1987] en la metodología AAII/BDI [Kinny *et al.* 1996]) o incluso de otros paradigmas de desarrollo de software (e.g. los casos de uso de UML adaptados a la metodología de agentes ADELFE [Bernon *et al.* 2002] o los modelos de clases en ALBERT [Dubois *et al.* 1994a]).

Pese a la diferencia de perspectivas, se puede considerar que el tratamiento que las propuestas mencionadas hacen de las componentes sociales e intencionales adolece de ciertos inconvenientes. Estos se relacionan esencialmente con los siguientes aspectos:

- *Necesidad de una visión integradora de los aspectos sociales e intencionales*. Muchas de estas propuestas se centran sólo en parcelas concretas y aisladas del SMA y no explican qué influencia existe entre lo social e intencional y otras características del agente, como la inteligencia o la autonomía.
- *Notación empleada*. Los lenguajes empleados en las metodologías basadas en agentes han de tener una adecuada capacidad expresiva para describir estos conceptos y facilitar su procesamiento [Wooldridge & Ciancarini 2000].
- *Conocimiento disponible para los procesos*. Es necesario conocimiento experto sobre los aspectos sociales e intencionales lo cual conduce a disciplinas como la Sociología, la Psicología, la Etnografía o la Antropología.

Las propuestas existentes en la ISOA no presentan necesariamente los tres grupos de inconvenientes a la vez ni en el mismo grado. Cómo se verá en el Capítulo 2, dedicado al estado del arte, algunos problemas son evidentes en ciertas propuestas mientras que prácticamente no existen en otras.

En todo caso, estos inconvenientes conducen a considerar la necesidad de proporcionar mecanismos genéricos e integradores, así como conocimiento experto, para trabajar con las componentes sociales e intencionales del paradigma de agentes en el seno de las metodologías de desarrollo. Al igual que ocurre con otros artefactos del proceso de desarrollo de software, el marco de análisis que trabaje con estos aspectos proporcionará guía acerca de su tratamiento, métodos para capturar la información sobre ellos y procesos para su detección y manipulación en las especificaciones de los SMAs.

Dada la naturaleza de estas componentes intencionales y sociales, el marco buscado para su estudio puede estar inspirado en las ciencias sociales. Las sociedades humanas y los SMAs [Maes 1994, Sykara 1998] se caracterizan, entre otras, por su componente social e intencional. De hecho es frecuente considerar los sistemas de agentes software como sociedades de componentes intencionales. Este nexo facilita la aplicabilidad de las teorías sociológicas y psicológicas a los agentes. El conocimiento de las disciplinas sociales acerca de las organizaciones humanas establece una base sólida desde la que abordar las facetas intencionales y sociales de los SMAs de una forma integrada. Estas ciencias han trabajado durante largo tiempo con esta clase de

propiedades en las comunidades humanas: han desarrollado el vocabulario preciso para su análisis; han creado marcos integrados de estudio que contemplan los niveles social e individual y sus mutuas influencias; además, su experiencia investigadora proporciona un cuerpo amplio de casos de estudio que puede usarse como fuente de conocimiento experto sobre propiedades sociales concretas. Los principales inconvenientes de estos métodos son la necesidad de contar con expertos en ellos, que sus resultados no están directamente dirigidos al diseño software y que todos sus métodos y artefactos se describen en lenguaje natural, lo que dificulta su integración en un proceso de desarrollo software basado en lenguajes de diseño.

Siguiendo terminología de las ciencias sociales [Constantine 1995], nos referiremos en este trabajo a los aspectos relacionados con las componentes intencionales y sociales de los SMAs como *propiedades sociales* de los SMAs. Las *propiedades sociales* consideran conceptos organizativos, cognitivos, de desarrollo y evolución, o de motivación en el sistema. Ejemplos de estas propiedades serían que la consecución de los objetivos individuales no dificulte la satisfacción de los objetivos de la comunidad o la influencia de la jerarquía de la organización en el acceso a los recursos.

Entre los marcos de análisis existentes en las ciencias sociales, esta tesis propone el uso de la Teoría de Actividad (TA) [Vygotsky 1978], un marco antropológico, psicológico y sociológico para el estudio de organizaciones humanas. Como se verá en el Capítulo 3, la elección de la TA como ayuda al desarrollo de SMAs se debe a que ya ha sido aplicada al desarrollo de software, cumple los requisitos establecidos en esta misma sección para el marco de estudio de las *propiedades sociales*, reduce algunos de los problemas citados anteriormente en el uso de técnicas sociales en Ingeniería del Software (e.g. su integración con el desarrollo de software o la necesidad de expertos en ciencias sociales) y presenta características específicas que la hacen adecuada para guiar el proceso de desarrollo de SMAs.

El modo de trabajo de la TA consiste en identificar la situación de una sociedad, sus posibles conflictos y las soluciones que ésta adopta para resolverlos. Con dicha información, se pueden estudiar otras sociedades identificando sus contextos sociales, algunos de los cuales corresponden a *contradicciones* [Engeström 1987], i.e. situaciones conflictivas. En base a estos contextos, la TA puede explicar el comportamiento y evolución de los nuevos grupos humanos estudiados.

Este trabajo propone concebir el proceso de desarrollo de un SMA como un ciclo guiado por las *contradicciones* bajo el prisma de la TA. La definición en la TA de *contradicción* psicológica y social guarda ciertas semejanzas con la citada en [Pressman 2000] para sistemas software. Por *contradicción* en las especificaciones se entiende cualquier inconsistencia, ambigüedad, vaguedad o incompletitud [Pressman 2000]. En ambos casos el sistema representado no cumple plenamente sus expectativas porque se ignoran ciertas facetas o no se consideran todos sus elementos de una forma holística. Las definiciones de ambos tipos de contradicciones inciden en aspectos diferentes: la TA en la motivación y actividades de las sociedades y sus individuos; la Ingeniería del Software en los artefactos de un sistema y su modo de funcionamiento. El paradigma de agentes con sus características intencionales y sociales ofrece un nexo entre ambas definiciones.

Siguiendo la TA, nuestra investigación plantea realizar sobre el proceso de desarrollo del SMA un ciclo continuo de recolección de nueva información (i.e.

identificación de la situación en la sociedad), detección de *contradicciones* en las especificaciones (i.e. detección de *contradicciones* sociales) y generación de nuevas especificaciones para solventar los problemas detectados (i.e. solución de los conflictos en las sociedades). La recolección de nueva información corresponde en los procesos de desarrollo de software a la fase de captura de requisitos, donde se obtiene nueva información del sistema. En cuanto a la detección de *contradicciones* puede verse como una forma de verificación de ciertas propiedades en SMAs. La solución de las *contradicciones* es una modificación guiada de las especificaciones.

Por supuesto, podríamos plantear el uso de la TA en otras tareas de un proceso de desarrollo de SMAs, distintas de la obtención de información y la verificación de propiedades. No obstante, hay que señalar que el objetivo de este trabajo no es crear otra metodología de desarrollo sino métodos para trabajar con las *propiedades sociales* en las metodologías existentes. Por ello, la presente investigación se centra en la captura de requisitos, en la comprobación de *propiedades sociales* y en la sugerencia de adaptaciones en las especificaciones a dichas propiedades. Se considera que estas son las actividades esenciales para trabajar con las *propiedades sociales*, los puntos donde se puede aplicar la TA con mayor idoneidad y que además requieren una menor modificación de las rutinas de trabajo de los desarrolladores. De todas formas, la aplicación de la TA como ayuda en otras fases del desarrollo de SMAs se plantea como trabajo futuro de esta tesis.

Por último señalar que una ventaja adicional del uso de conceptos relacionados con las sociedades humanas, particularmente los de la TA, en el proceso software es que ayudan en la comunicación entre desarrolladores y clientes, como se sostiene en [Fjeld *et al.* 2002, McGrath & Uden 2000]. Las abstracciones usadas por la TA son cercanas a todos los implicados en el desarrollo puesto que representan conocimiento universal acerca de las organizaciones humanas. Esto permite su uso como herramienta de comunicación común, no específica de los dominios de trabajo de clientes o desarrolladores.

Dados estos precedentes, se puede perfilar con mayor precisión el objetivo global de esta tesis: proporcionar un método genérico basado en la TA para la descripción y uso de las *propiedades sociales* en los SMAs, que puedan llevar a cabo los desarrolladores, que complemente los procesos de desarrollo de SMAs existentes y que cuente con herramientas automatizadas de soporte. El uso de las *propiedades sociales* en el proceso de desarrollo de SMAs se realizará esencialmente mediante técnicas de captura de requisitos y de verificación y validación, incluyendo también la transformación de las especificaciones para satisfacer dichas propiedades.

1.3. Objetivos

Partiendo de las premisas introducidas en el apartado anterior, los objetivos de esta tesis se han centrado en los siguientes aspectos:

- *Integración de la TA en el desarrollo de SMAs*. Se trata de contar con técnicas aplicables a la ISOA que se centren en la componente intencional y social de los SMAs. Esta tesis se considerará la captura de requisitos, la comprobación de propiedades y el tratamiento de contradicciones sobre estos aspectos.

  - *Adaptación de técnicas procedentes de la TA a la captura de requisitos para SMAs*. Las técnicas de captura de información en software suelen estar demasiado centradas en el sistema a construir y presentan un marcado sesgo hacia el desarrollo [Zave & Jackson 1997, Lamsweerde 2000]. Ello hace difícil considerar características del entorno y aspectos ajenos a las prácticas comunes de la Ingeniería del Software. Para solventar este problema usaremos técnicas ya probadas en el estudio de personas y sus organizaciones para tratar con esos aspectos, y definiremos sus conexiones con en el resto del proceso de desarrollo para permitir el trasvase de información, como sugieren entre otros [Nuseibeh & Easterbrook 2000, Malsch 2001, Sommerville 2001, Saam & Schmidt 2002].
  - *Comprobación de propiedades*. La mayor parte de los mecanismos de comprobación de propiedades en la ISOA fueron concebidos para aspectos diferentes de las intencionales y sociales. Se pretende proporcionar métodos centrados en aprovechar las características particulares de estos aspectos y el conocimiento disponible sobre ellos.
  - *Integración de las contradicciones de los modelos en el proceso de construcción del software como un elemento director de la evolución*. El estudio de las *contradicciones* [Leontiev 1978] de un SMA siguiendo los procesos de la TA permite detectar conflictos y aportar soluciones mediante el análisis de las especificaciones del sistema. Este análisis se realiza con el conocimiento de la TA sobre cómo funcionan las sociedades humanas adaptado a SMAs. El ciclo continuo de detección de conflictos y resolución hace evolucionar la especificación del sistema.
- *Síntesis de la TA en un lenguaje de diseño*. La integración de la TA en el desarrollo de SMAs mencionada en los objetivos anteriores requiere un medio para expresar conceptos de la TA de acuerdo con las prácticas ingenieriles. Pese a las capacidades que proporcionaría un eventual uso de la TA sobre especificaciones de SMAs, existe un obstáculo no trivial en su aplicación: la TA está descrita mediante lenguaje natural. Ello hace difícil aprovechar directamente el conocimiento contenido en su investigación de una forma automatizada. Se requieren lenguajes adecuados para la ISOA que expresen el comportamiento de las sociedades según lo detalla la TA.

Las técnicas basadas en la TA de esta tesis se han implementado en una herramienta de ayuda al desarrollo de SMAs integrada en el entorno *INGENIAS Development Kit* (http://ingenias.sourceforge.net). Esta herramienta recoge el conocimiento de la TA sobre los contextos sociales y su utilización. La aplicación desarrollada ofrece una interfaz con APIs que le permite coordinarse con las herramientas de modelado de las metodologías para SMAs.

1.4. Método de trabajo

Para abordar los objetivos precedentes se ha seguido el siguiente plan de trabajo:

1. *Estudio del estado del arte* en cuanto a técnicas de ayuda al desarrollo para metodologías software, centrado en las usadas en el paradigma de agentes y las adaptadas de las ciencias sociales. Se han considerado esencialmente las técnicas para la captura de requisitos y las destinadas a la verificación y validación de las especificaciones, puesto que son las más cercanas en cuanto a objetivos con las propuestas en esta tesis. En las distintas técnicas se han analizado la actividad del proceso software a la que están orientadas, la forma de aplicación y los resultados que proporcionan.
2. *Estudio de la TA y sus herramientas* como posible alternativa y complemento de algunas de las técnicas vistas en el paso anterior.
3. *Determinar los puntos de aplicación de la TA en el proceso software para el desarrollo de un SMA* y las técnicas aplicables en dichas fases.
4. *Establecer la infraestructura necesaria para aplicar las técnicas de TA* sobre un proceso software orientado a agentes. Se trata de evitar la creación de una nueva metodología de agentes. Puesto que lo que se pretende es proporcionar herramientas de ayuda al desarrollo en las metodologías existentes, se ha de determinar cómo se hará la conexión y el trasvase de resultados entre ellas y los métodos de la TA.
5. *Adaptación de las técnicas de TA* seleccionadas al paradigma de agentes y a los usos de trabajo propios de la Ingeniería del Software. Esta adaptación se realizará en función de la infraestructura referida en el punto anterior. En este paso es preciso definir los procedimientos de trabajo adaptados al proceso software.

1.5. Estructura del documento

Los resultados de esta tesis y la experimentación que los apoya son recogidos en el resto del documento que se organiza en siete capítulos como se indica a continuación.

El capítulo segundo repasa algunas técnicas relevantes usadas actualmente como ayuda al desarrollo de software en cuanto a la captura de requisitos y la comprobación de propiedades en las especificaciones. Esta revisión se centra en los métodos usados para la ISOA y en precedentes de aplicación de las ciencias sociales a la Ingeniería del Software en general. El objetivo es determinar la clase de resultados que están proporcionando estas técnicas y sus limitaciones. Con ello se razona la necesidad de un enfoque alternativo a los actualmente usados en la ISOA y se determina qué requisitos debe reunir una propuesta integradora de los aspectos sociales e intencionales.

El tercer capítulo está dedicado a la adaptación de los conceptos manejados por la TA al paradigma de agentes. Se determinan las correspondencias entre las abstracciones de ambos grupos, en las cuales se basa la infraestructura necesaria para el intercambio de información entre la TA y las especificaciones de los SMAs. Este

capítulo también presenta las estructuras usadas para describir *propiedades sociales* y el método para localizarlas en las especificaciones de SMAs.

El capítulo cuarto presenta la adaptación de las técnicas de la TA que se han identificado como adecuadas para la captura de requisitos. Aquí se recogen los procesos necesarios para obtener de los clientes la información de entrada al proceso software, los requisitos, y plasmarla en abstracciones adecuadas para el trabajo en el resto del ciclo de desarrollo.

El capítulo quinto detalla cómo llevar a cabo la evolución de las especificaciones hasta el sistema final. El proceso está guiado por las contradicciones en la información. Los desarrolladores detectan conflictos que explican al cliente y le plantean posibles soluciones en términos de su propio dominio. Las respuestas del usuario proporcionan al proceso la realimentación necesaria para modificar los modelos y realizar nuevas iteraciones de desarrollo.

El capítulo sexto aborda cómo se han plasmado las técnicas previas en herramientas de ayuda al desarrollo que complementan a los entornos de modelado de las metodologías. Estas herramientas son las encargadas de guiar a los desarrolladores en el ciclo de vida con el conocimiento de la TA.

En el capítulo séptimo se muestra un caso de estudio realizado para validar la propuesta de esta tesis. El caso de estudio describe la aplicación de las técnicas de los capítulos previos en la especificación de un SMA.

Finalmente, el octavo capítulo recoge las conclusiones sobre la propuesta a la vista de los resultados obtenidos en la experimentación y el trabajo futuro.

Capítulo 2. Estado del arte

Una de las aportaciones más relevantes del paradigma de agentes es contemplar los sistemas como sociedades de actores intencionales. Considerar estos aspectos implica analizar sistémicamente múltiples facetas del SMA que entrelazan los niveles individual y social. Este capítulo se centra en cómo se ejecuta y en qué se basa la captura de requisitos y la comprobación de propiedades sociales, que son los aspectos considerados en este trabajo de tesis. La revisión considera la forma en que se llevan a cabo estas actividades en la ISOA. También se contemplan ejemplos que proceden de las ciencias sociales para ilustrar conceptos relacionados con esta investigación. Finalmente, el capítulo termina con una revisión de las limitaciones en las propuestas vistas, que detalla algunos de los aspectos que ha de abordar la presente propuesta.

2.1. Introducción

El paradigma orientado a agentes [Newell 1982, Shoham 1993] usa como abstracción fundamental la de los *agentes*, entidades que tienen objetivos que persiguen satisfacer y que pueden alcanzar mediante la ejecución de una o varias tareas. Los agentes pueden interactuar con otros en esta persecución de sus propósitos dando lugar entonces a los SMAs [Sykara 1998].

Basándose en estos conceptos, la investigación en agentes ha producido modelos teóricos, lenguajes de programación y arquitecturas software de agentes y SMAs. Se trata de numerosos elementos cuya integración en la construcción de un sistema no es trivial. De ahí la necesidad de las técnicas de ayuda para la construcción de SMAs que se engloban en la Ingeniería del Software Orientada a Agentes (ISOA) [Jennings & Wooldridge 2000].

A pesar de la variedad de metodologías en la ISOA [Iglesias *et al.* 1999, Wooldridge & Ciancarini 2000, Gómez-Sanz 2003, Gómez-Sanz & Pavón 2004], existen un conjunto de limitaciones que se han verificado en las propuestas existentes y que hacen pensar en la necesidad de suministrar notaciones y mecanismos específicos para trabajar con las *propiedades sociales* de los SMAs. Por *propiedades sociales* de los SMAs entendemos la información relacionada con las componentes intencionales y sociales de estos sistemas, tales como organización, cognición de los agentes, normas sociales o motivación. Las limitaciones en torno a estas propiedades aparecen ligadas esencialmente con las siguientes cuestiones:

- *Necesidad de una visión global de los aspectos sociales e intencionales*. La ISOA propugna como uno de sus principios de análisis la necesidad de contemplar de una forma sistémica las diferentes facetas de los SMAs [Maes 1994]. Con ello se

indica que contemplando de forma aislada los aspectos del sistema no se pueden capturar sus influencias mutuas. Las *propiedades sociales* de un SMA no son una excepción. Su comprensión y modelado requieren análisis que consideren integradamente múltiples características, tanto en la dimensión vertical (i.e. niveles individual y social) como horizontal (i.e. distintos aspectos en el mismo nivel organizativo).

- *Notación empleada*. El lenguaje de modelado [Nuseibeh & Easterbrook 2000] tiene una gran influencia en la clase de información que se puede observar, describir e incluso concebir. Los aspectos intencionales y sociales del paradigma de agentes requieren lenguajes que faciliten el trabajo con ellos, es decir, su modelado, modificación en el proceso software y la verificación de sus propiedades relevantes [Wooldridge & Ciancarini 2000]. Por ejemplo, algunas propuestas sólo permiten el modelado de ciertos tipos de *propiedades sociales* (e.g. contemplan los motivos de los individuos pero no los de las organizaciones en las que estos se hayan inmersos). Esta visión parcial de las *propiedades sociales* dificulta el tratamiento integrado que persigue esta tesis.
- *Conocimiento experto para los procesos*. Trabajar con una clase de información, por ejemplo los aspectos intencionales y sociales de un SMA, requiere disponer de una notación para expresarla y procesos para manipularla, pero también conocimiento sobre esa información. La notación y los mecanismos para procesar esa notación son sólo la infraestructura para trabajar con la información. El uso relevante de la infraestructura requiere conocimiento adicional acerca de la información: qué podemos representar, cómo representarlo, para qué se puede utilizar o en qué circunstancias se puede aplicar. Un ejemplo de esta situación sería la verificación de propiedades sobre las características organizativas de un SMA. Aun disponiendo de los mecanismos apropiados, los desarrolladores pueden tener dificultades para establecer propiedades interesantes acerca de la comunidad de agentes que no sean triviales. Tradicionalmente los desarrolladores han buscado este conocimiento experto sobre el desarrollo en la literatura propia de la Ingeniería del Software. El paradigma de agentes con sus *propiedades sociales* ofrece nuevas fuentes de información en las ciencias sociales, y la posibilidad de una visión integrada de dichas propiedades en el sistema software y su entorno humano. La mayor parte de las propuestas revisadas ignoran este aspecto o bien se centra en el conocimiento para propiedades muy concretas, por ejemplo relaciones de poder entre los agentes o interbloqueos en el uso de recursos.

Para mostrar estas limitaciones se ha escogido un conjunto de metodologías que tienen en común la definición de un proceso de desarrollo de SMAs, incluir métodos de captura de requisitos y aplicar técnicas de verificación y validación. La elección de estas actividades de las metodologías se debe a que son las más cercanas a los objetivos de esta tesis (ver sección 1.3). Teniendo en cuenta estos criterios, las metodologías que se han seleccionado son: AAII/BDI [Kinny *et al.* 1996], KAOS [Dardenne *et al.* 1993], ALBERT [Dubois *et al.* 1994a], i* [Yu 1999], DESIRE [Brazier *et al.* 1997], Tropos [Castro *et al.* 2001], ADELFE [Bernon *et al.* 2002] e INGENIAS [Pavón & Gómez-Sanz 2003].

El estudio de las limitaciones señaladas llevó a analizar en cada metodología varios aspectos:

- *Lenguajes de modelado*. Se trata de considerar la capacidad expresiva e integradora y la facilidad de uso de los lenguajes utilizados en las metodologías para representar información relacionada con las *propiedades sociales* y realizar su posterior tratamiento.
- *Captura de requisitos y comprobación de propiedades*. Es el análisis de los métodos específicos para realizar estas actividades sobre las *propiedades sociales*. La captura de requisitos y la comprobación de propiedades son las dos actividades en las que se centra esta tesis. Analizar estos métodos permite conocer el nivel de integración en el tratamiento de las *propiedades sociales* en una metodología dada.
- *Guía en el trabajo con las propiedades sociales*. Además de disponer de métodos para trabajar con las propiedades hay que contar con directrices para su uso. Éstas también constituyen un indicador de si la metodología tiene una perspectiva global del tratamiento de las componentes sociales e intencionales.
- *Conocimiento experto*. Incluye la información que la metodología aporta a los desarrolladores acerca de *propiedades sociales* de los SMAs. Esta información puede estar plasmada en elementos predefinidos de la metodología (tales como la arquitectura que propone para los agentes y SMAs o la notación) o con librerías de propiedades aplicables sobre los distintos desarrollos. En este trabajo interesa especialmente el conocimiento experto relacionado con la captura de requisitos y la comprobación de *propiedades sociales*.

La solución a las limitaciones de las metodologías de la ISOA anteriormente expuestas, lleva a plantear nuevas formas de concebir el desarrollo de SMAs. En la línea de [Winograd & Flores 1986, Nuseibeh & Easterbrook 2000], consideramos que las componentes intencionales y sociales relacionadas con los sistemas software requieren métodos distintos de los usados tradicionalmente en la Ingeniería del Software, que son los que se han aplicado en la ISOA. Una fuente de inspiración para estos métodos son las ciencias sociales. Existen características que permiten establecer un vínculo entre los conceptos del paradigma de agentes y los de las sociedades humanas para las *propiedades sociales*. Las sociedades humanas y los SMAs son organizaciones de actores intencionales que interaccionan para satisfacer sus propósitos. Además, las *propiedades sociales* de los SMAs, que constituyen el objeto de estudio del presente trabajo, se centran en aquellos aspectos del sistema software más cercanos a la metáfora de las organizaciones humanas.

Puesto que esta investigación busca integrar métodos de las ciencias sociales en la ISOA, se ha considerado conveniente estudiar precedentes de esta integración en desarrollos de Ingeniería del Software convencional. El motivo principal ha sido encontrar referentes que indiquen cómo se ha logrado salvar las diferencias conceptuales entre dos disciplinas tan distintas, sobre todo en lo que a métodos de especificación y metodología de trabajo se refiere.

El resto del capítulo se estructura como sigue. La sección 2.2 introduce breves definiciones acerca de las *propiedades sociales* de los SMAs y de qué tareas se consideran dentro de la captura de requisitos y la comprobación de propiedades en el resto del capítulo. La sección 2.3 incluye la revisión de varias metodologías de

desarrollo que forman parte de la ISOA y la 2.4 de técnicas adaptadas a la Ingeniería del Software desde las ciencias sociales. En ambos casos se analiza el trabajo con las *propiedades sociales* en cuanto a la captura de requisitos y la comprobación de propiedades, así como las herramientas automatizadas que emplean. Finalmente se hace una revisión de las ventajas y limitaciones de las técnicas vistas en este estado del arte y de las características deseables en una propuesta que resuelva dichos inconvenientes.

2.2. Definiciones

La presente sección comienza recordando qué información se considera al hablar de las *propiedades sociales* de un SMA. Luego discute acerca de los objetivos involucrados en las actividades de captura de requisitos y comprobación de propiedades de un proceso de desarrollo. Asimismo se incluye una breve revisión de las consideraciones a tener en cuenta al realizar dichas actividades y las tareas involucradas.

2.2.1. *Propiedades sociales* de los SMAs

Las propuestas de la ISOA estudiadas a lo largo de esta investigación tienen en común contemplar sus SMAs como conjuntos de agentes que interactúan para alcanzar ciertos objetivos. Los agentes son entidades que siguen el Principio de Racionalidad [Newell 1982]: Cada acción tomada por un agente le ayuda a satisfacer uno de sus objetivos. Este hecho convierte a los agentes en entidades intencionales. Por otra parte, la interacción entre agentes necesaria para lograr algunos objetivos demanda una visión del sistema como sociedad de agentes que se comunican y prestan servicios [Castelfranchi 2000]. Es la componente social del paradigma.

La información relativa a estas componentes intencional y social queda agrupada en este trabajo bajo el nombre genérico de *propiedades sociales* de los SMAs. Al igual que ocurre en las ciencias sociales [Constantine 1995], esta clase de información hace referencia a aspectos tales como la estructura organizativa, las posibles interacciones entre los miembros de la comunidad, la motivación de individuos y grupos, las capacidades cognitivas, las normas sociales imperantes, la educación o los conflictos entre los componentes del SMA. Ejemplos de estas propiedades serían el tipo de organización en el SMA (e.g. jerarquía, comité o mercado), la coordinación (e.g. recursos compartidos, reglas comunes o árbitros), la motivación de los agentes (e.g. egoístas que sólo sirven a su propio propósito o agentes capaces de anteponer los objetivos de la organización a los propios) o los procesos de creación de la sociedad (e.g. si los agentes crean las reglas de su organización o si son capaces de decidir socialmente cuál es el agente más fiable para proporcionar una información).

Las *propiedades sociales* no se refieren solamente a un SMA concreto. También incluyen la información manejada por las metodologías. Esta información se puede encontrar por ejemplo embebida en la arquitectura que proponen para agentes y SMAs.

2.2.2. Captura de Requisitos

De acuerdo con una caracterización frecuentemente referenciada:

La definición de los requisitos es una cuidadosa valoración de las necesidades que un sistema va a cubrir. Debe decir por qué el sistema es necesario, basándose en las condiciones actuales o previstas, que pueden ser operaciones internas o un mercado externo. Debe decir cuáles características del sistema servirán y satisfarán este contexto. Y debe decir cómo se construirá el sistema [Ross 1977]

Esta definición ya contiene los principales ingredientes de los elementos que deben contemplarse en los requisitos del sistema. Una definición más actual afirma que:

La Ingeniería de Requisitos está relacionada con la captura de aquellos objetivos de alto nivel que han de ser alcanzados por el sistema previsto, su refinamiento y operacionalización mediante la especificación de servicios y restricciones, y la asignación de responsabilidades para los requisitos resultantes a agentes tales como humanos, dispositivos y software [Lamsweerde & Willemet 1998].

La captura de requisitos incluye varias actividades en las cuales la interacción con los clientes tiene una importancia primordial [Nuseibeh & Easterbrook 2000]. No sólo se trata de obtener de los clientes conocimiento sobre los objetivos del sistema a construir o su contexto, sino también de que los desarrolladores sean capaces de comunicarles la visión que han obtenido de dichos requisitos y de que ambos grupos puedan negociar sobre ellos si es preciso. Considerar estas actividades en relación con el manejo de las *propiedades sociales* en la ISOA resalta la necesidad de usar lenguajes apropiados para representar dichas propiedades y razonar sobre ellas. Además, estos lenguajes han de ser comprensibles tanto por los clientes como por los desarrolladores, ya que los requisitos son un artefacto construido conjuntamente por ambos grupos.

2.2.3. Comprobación de Propiedades

La comprobación de propiedades consiste en demostrar que los artefactos generados en el proceso software poseen ciertas características. Bajo este epígrafe se consideran dos tipos de demostraciones [Sommerville 2001]:

- *Validación*. Se trata de probar que los artefactos producidos satisfacen sus requisitos.
- *Verificación*. Se pregunta si los artefactos satisfacen ciertas propiedades. Este es el caso de las comprobaciones sobre la ausencia de interbloqueos o la inclusión de un patrón de organización dado en el SMA.

Existen varias revisiones recientes de las técnicas en uso para la verificación y validación de especificaciones en la ISOA. Dos de ellas son [Wooldridge & Ciancarini 2000] y [Gómez-Sanz & Pavón 2004].

Las técnicas de verificación y validación aplicadas en la ISOA que veremos en este apartado no quedan restringidas a las propuestas de carácter formal. Alguna revisiones acerca de la comprobación de propiedades en SMAs, como [Wooldridge & Ciancarini 2000], sólo consideran las propuestas axiomáticas (i.e. verificación deductiva) y semánticas (i.e. comprobación de modelos), que están basadas en notaciones formales. La revisión de esta tesis abarca todos los procesos que puedan emplearse para revisar unas especificaciones, sea cual sea su notación.

En el caso concreto de este trabajo, resulta de especial interés facilitar la representación de las *propiedades sociales* del SMA, su detección en las especificaciones y la descripción de las posibles modificaciones a realizar en dichas especificaciones.

2.3. Ingeniería del Software Orientada a Agentes

Esta sección realiza un recorrido por algunas de las propuestas extraídas de la Ingeniería del Software para la construcción de SMAs. La revisión aborda metodologías que han nacido en la ISOA inspiradas por las experiencias en el desarrollo de SMAs y en la observación de las sociedades humanas. Para ello se contemplan, siguiendo un orden cronológico de aparición, los ejemplos de AAII/BDI [Kinny *et al.* 1996], KAOS [Dardenne *et al.* 1993], ALBERT [Dubois *et al.* 1994a], i* [Yu 1999], DESIRE [Brazier *et al.* 1997], Tropos [Castro *et al.* 2001], ADELFE [Bernon *et al.* 2002] e INGENIAS [Pavón & Gómez-Sanz 2003].

Esta revisión no contempla metodologías procedentes de otros paradigmas distintos del de agentes. Las metodologías de la ISOA han sido construidas frecuentemente sobre otras ya existentes con las adaptaciones necesarias para el paradigma de agentes, incorporando por tanto el conocimiento y experiencia de otros paradigmas. Además hay que tener en cuenta que el paradigma de agentes es considerado muchas veces exclusivamente como una herramienta conceptual, que se utiliza en las etapas iniciales del desarrollo y que admite distintas implementaciones. Debido a ello, a partir de cierto estado de evolución de las especificaciones el desarrollo se realiza siguiendo metodologías propias de otros paradigmas, como la orientación a objetos o técnicas formales.

2.3.1. AAII/BDI

La metodología AAII/BDI [Kinny *et al.* 1996] es una de las primeras propuestas para el modelado de SMAs. Su trabajo incluyó la creación de un proceso software similar a los existentes en metodologías orientadas a objetos y el desarrollo de modelos de arquitectura para SMAs y sus agentes. Los SMAs se modelan en esta propuesta mediante dos puntos de vista: el de los agentes individuales (i.e. vista interna) y el del SMA (i.e. vista externa).

La arquitectura interna de los agentes está inspirada en el modelo cognitivo para los humanos propuesto en [Bratman 1987]. La metodología AAII/BDI [Rao & Georgeff 1991] adapta este modelo para describir el estado interno o mental del agente a través de *Creencias-Deseos-Intenciones* ("*Beliefs-Desires-Intentions*")

(BDI). Las *creencias* reflejan información sobre el estado del mundo o el propio agente, los *deseos* describen su motivación y las *intenciones* definen la estructura de control del agente en términos de planes. El agente trata de satisfacer sus *deseos* mediante la ejecución de planes. Estos planes se desarrollan en función de las *creencias* del agente en cada momento. El modelo externo del sistema se basa en roles, servicios e interacciones.

El trabajo de la metodología AAII/BDI ha tenido una notable influencia en las propuestas posteriores. Por ello su tratamiento de las *propiedades sociales* del SMA resulta prototípico de muchas propuestas de la ISOA.

Los lenguajes usados para representar y analizar las *propiedades sociales* son un formalismo y una notación gráfica [Kinny *et al.* 1997]. El formalismo es una lógica multimodal con árboles temporales ("*multi-modal branching-time logics*"). Estos árboles son estructuras donde cada nodo representa un estado del mundo del agente, con un único pasado y múltiples futuros que a su vez representan alternativas en la ulterior evolución del agente y su entorno. Los árboles expresan el razonamiento del agente, las condiciones del entorno o las normas impuestas por la organización. El uso del formalismo proporciona un gran poder expresivo y permite representar de manera uniforme las *propiedades sociales* con gran versatilidad. La notación gráfica está inspirada por la orientación a objetos. Los modelos gráficos se usan para visualizar parte de la estructura del sistema (e.g. relaciones de herencia y descomposición entre agentes) y sus componentes (e.g. *creencias* de un agente). Como limitación de los lenguajes, los aspectos relativos a la sociedad de agentes no son descritos mediante primitivas específicas, como sí se hace con el estado mental de los agentes. Esto hace que la descripción de los aspectos sociales pueda resultar más compleja.

Otro aspecto de interés son las pautas para buscar nueva información sobre las *propiedades sociales* y realizar verificaciones. AAII/BDI sólo describe tareas del proceso software en lenguaje natural [Kinny *et al.* 1996]. Aunque ésta es la clase de guía incluida en la mayor parte de las propuestas, deja a criterio del desarrollador la localización de la parte de las especificaciones donde actuar y cómo hacerlo.

Sobre la captura de requisitos, AAII/BDI no proporciona mecanismos específicos para llevarla a cabo. Los desarrolladores capturan el conocimiento con las técnicas de su elección y lo plasman en las notaciones de la metodología sin más guía que su experiencia. Dado el mayor peso de la notación formal, los requisitos obtenidos se encuentran en un lenguaje poco adecuado para la discusión con el cliente.

Los mecanismos de comprobación de propiedades están apoyados en los métodos del formalismo (tales como comprobación de modelos [Rao & Georgeff 1995] o demostración axiomática [Rao & Georgeff 1991]). El principal inconveniente de esta formalización de las arquitecturas BDI es la dificultad para manejar *contradicciones* (en el sentido visto en la sección 1.2) [Rao & Georgeff 1995]. No se admiten inconsistencias en el estado mental de los agentes. Considerando que estas *contradicciones* son casi inevitables (por ejemplo objetivos contradictorios o especificaciones aun en desarrollo), la única solución permitida es la de tener mecanismos de resolución de conflictos. Sin embargo, esta solución no siempre es posible (por ejemplo cuando los agentes tienen información imperfecta) y entonces se fuerza el descarte de parte del estado mental.

La última observación es acerca del conocimiento experto sobre las *propiedades sociales* en AAII/BDI. Éste queda limitado a la arquitectura propuesta para los

agentes y al lenguaje de especificación. Son los desarrolladores los que han de establecer *propiedades sociales* de interés, cómo describirlas en los lenguajes de la metodología y cuáles pueden ser las modificaciones en las especificaciones ante su presencia o ausencia.

Frente a las propuestas centradas en formalismos como la de AAII/BDI, el trabajo en esta tesis ha optado por dar un mayor peso a la facilidad de uso y al conocimiento experto sobre *propiedades sociales*. Por ello se ha optado por usar en su vocabulario abstracciones sociales de la TA y un lenguaje gráfico basado en UML. Los conceptos de la TA usados son más sencillos de comprender para los clientes que los inspirados en paradigmas software, ya que se fundan en conocimiento común sobre nuestras propias sociedades humanas. El uso de UML para representar este vocabulario cuenta con la ventaja de estar ampliamente difundido en la Ingeniería del Software. Por otra parte, nuestra propuesta suministra conocimiento experto explícito sobre las *propiedades sociales* extraído de la TA. Este conocimiento aparece como repositorios de propiedades para la captura de requisitos y la comprobación de especificaciones.

2.3.2. KAOS

KAOS [Dardenne *et al.* 1993] es una metodología de adquisición de requisitos. Trata de resolver el problema de determinar el conocimiento relevante sobre el dominio, las necesidades del cliente y su operacionalización en especificaciones de servicios y restricciones. Para ello intenta reaprovechar el conocimiento sobre la captura de requisitos y emplear técnicas que automaticen en lo posible el proceso.

En KAOS, los requisitos se adquieren como instancias específicas para el dominio, de un metamodelo conceptual (ver Fig. 1). Su proceso de captura de requisitos involucra tres niveles de modelado. El *metanivel* se refiere a abstracciones independientes del dominio. El *nivel de dominio* trabaja con conceptos específicos del dominio de aplicación. Finalmente, el *nivel de instancia* emplea instancias específicas de los conceptos del *nivel de dominio*. La información de estos modelos se describe mediante una notación gráfica y una lógica para tiempo real. La notación gráfica representa un subconjunto del lenguaje del formalismo.

Los requisitos del sistema se obtienen gradualmente y se expresan en un lenguaje de adquisición que refleja fielmente la estructura del metamodelo. Los conceptos en KAOS tienen una estructura genérica con dos niveles: una capa semántica externa que declara el concepto, sus atributos y sus relaciones con otros conceptos; una capa interna para su definición formal. El nivel declarativo se usa para el modelado conceptual mediante un lenguaje visual, la trazabilidad de los requisitos mediante la navegación de la red semántica de conceptos; y la reutilización de especificaciones. El nivel de aserción es opcional y se usa para el razonamiento formal.

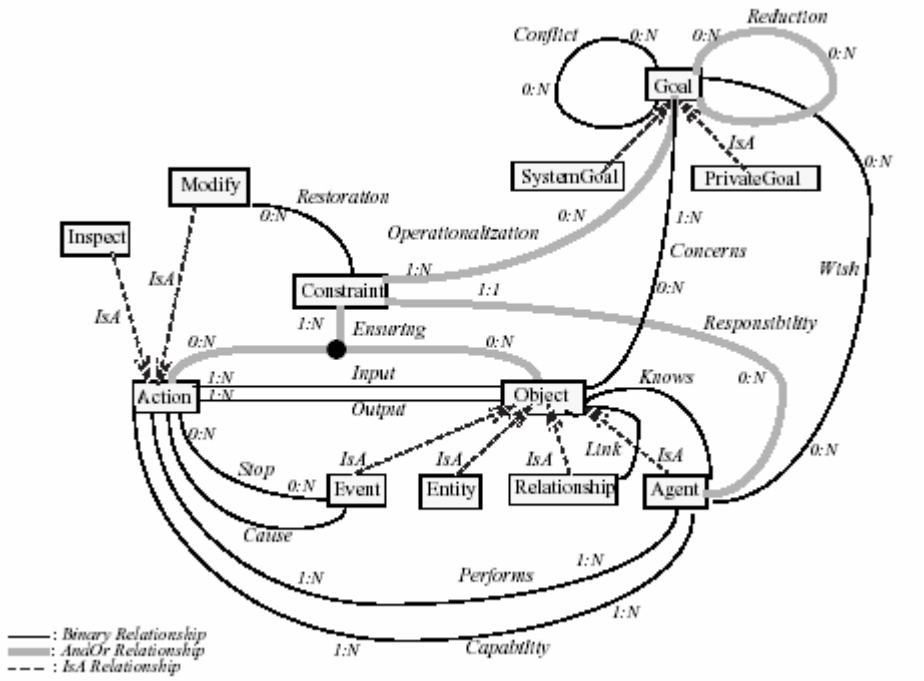


Fig. 1. Porción del metamodelo conceptual de KAOS.

A continuación se examina cómo se tratan las *propiedades sociales* del SMA con esta propuesta.

KAOS emplea dos lenguajes de modelado, uno gráfico y otro formal. Los conceptos del lenguaje gráfico son un subconjunto de los del formal por lo que pueden ser traducidos a éste. Se trata de un lenguaje gráfico muy rico que no se limita a contemplar los aspectos estructurales o dependencias básicas del SMA. Buena parte de las propuestas existentes en la ISOA (por ejemplo ALBERT en la sección 2.3.3, i* en 2.3.4 y ADELFE en 2.3.7) sólo incluyen en sus modelos gráficos relaciones de herencia entre agente, agregación entre componentes, dependencia jerárquica o dependencias para la satisfacción de objetivos. Por el contrario, KAOS contempla una taxonomía de conceptos más amplia. En cuanto al formalismo, se trata de una lógica para tiempo real que aporta especificaciones de grano fino sobre la componente gráfica. En todo caso, la mayor parte del modelado de KAOS se puede realizar con el lenguaje gráfico, lo que facilita la tarea de los desarrolladores. KAOS no incluye conceptos relativos a la sociedad de agentes.

La guía en el proceso de captura de requisitos viene dada por el metamodelo e incluye preguntas-cuestiones y validación de las entradas. El metamodelo puede recorrerse de distintas formas que determinan el orden de adquisición de instancias de los conceptos y las relaciones que se pueden usar. De este modo se da un procedimiento preciso sobre qué abstracciones se deben usar en cada momento aunque no existen pautas sobre cómo se identifican en el dominio real.

La detección y verificación de propiedades confía en la notación formal y sus mecanismos asociados. La herramienta GRAIL/KAOS [Bertrand *et al.* 1998] también puede ser usada para realizar verificaciones básicas manualmente.

El conocimiento experto de KAOS forma parte de su metamodelo, sus lenguajes y librerías. Aquí hay que destacar que al basar la definición de su vocabulario en metamodelos, KAOS permite modificar su base conceptual con mayor facilidad que otras propuestas para adaptarla a nuevo conocimiento, ya que sólo requiere cambiar dicho metamodelo. Por otra parte, KAOS contempla librerías de conocimiento acerca del metamodelo (por ejemplo mediante meta-restricciones) y dominios (tales como redes de teléfonos o administración). Este conocimiento puede ser concretado para nuevos problemas proporcionando guía en el proceso y conceptos relevantes para las especificaciones. No obstante, este conocimiento no puede considerarse centrado en las *propiedades sociales* del SMA, sino que hace más bien referencia al diseño del sistema como artefacto software.

En relación con KAOS, el marco conceptual de nuestra propuesta es más reducido. Si bien esto dificulta capturar ciertos conceptos, facilita la comprensión de la información, especialmente por los clientes. Otro punto diferenciador de las técnicas con la TA es que se centran en suministrar conocimiento experto inspirado por las ciencias sociales, que incide especialmente en los aspectos intencionales y sociales de las especificaciones. Además, esta investigación opta por usar ese conocimiento mediante un proceso de encaje de patrones para la verificación de propiedades. Este proceso es más sencillo de comprender que las pruebas formales de KAOS.

2.3.3. ALBERT

ALBERT [Dubois *et al.* 1994a] es una propuesta para la captura y verificación de requisitos en SMAs. ALBERT se inspira en la orientación a objetos para especificar los agentes. Incluye un lenguaje formal basado en la lógica temporal y una representación gráfica que permite realizar parte del modelado sin recurrir al formalismo.

Las especificaciones con ALBERT contemplan dos niveles:

- *Nivel de Agente*. Se asocian al agente un conjunto de posibles comportamientos sin tener en cuenta el comportamiento de otros agentes.
- *Nivel de Sociedad*. Se consideran las interacciones entre agentes lo que da lugar a restricciones en el comportamiento de los agentes individuales.

La declaración de un agente en ALBERT incluye la descripción de la estructura de su estado y la lista de las acciones que pueden afectarle. El comportamiento de un agente se define mediante un conjunto de *vidas*. Una *vida* es una secuencia potencialmente infinita de estados y cambios en estos mediante acciones.

Las sociedades son descritas como jerarquías de agentes. Es decir, ALBERT sólo contempla relaciones de poder entre sus agentes.

Sobre las declaraciones anteriores se introducen restricciones que controlan los posibles comportamientos del SMA. Estas restricciones se describen con el formalismo.

ALBERT es un ejemplo de aproximación formal complementada con elementos para facilitar su uso y aprendizaje, en este caso una representación gráfica y herramientas software. Pese a ello, ALBERT tiene importante limitaciones acerca del lenguaje empleado, su proceso, el conocimiento y las herramientas que emplea al considerar el tratamiento de las *propiedades sociales*.

ALBERT emplea dos lenguajes: un formalismo basado en la lógica temporal y un lenguaje gráfico. Al igual que ocurre con muchas aproximaciones que combinan varios lenguajes, se puede considerar que existe un lenguaje principal y otro auxiliar cuya capacidad expresiva queda lejos de la del principal. En este caso, el lenguaje gráfico permite sólo representar propiedades estructurales de los agentes, tales como los componentes del estado del agente, qué componentes de ese estado son controlados por el agente o la jerarquía en el SMA. El resto de las especificaciones ha de realizarse con el formalismo. El formalismo por su parte no cubre algunas propiedades de interés. Por ejemplo, su visión de las sociedades de agentes se limita a jerarquías de estos y no considera otras posibles organizaciones como la comunidad de expertos, el mercado o la comunidad científica [Sykara 1998]. De todas formas, el tratamiento integrado de las *propiedades sociales* es posible gracias a su expresión, más o menos compleja, en el formalismo.

Hay que resaltar que el formalismo, como los propios autores reconocen, no es un lenguaje adecuado para la discusión con los clientes sobre los requisitos. La solución de ALBERT a este problema de comunicación pasa por el uso del animador de especificaciones para que los clientes puedan comprender los requisitos. Sin embargo esta herramienta no permite transmitir toda la riqueza de las especificaciones.

En cuanto a los métodos, tanto la captura de requisitos como el trabajo con las *propiedades sociales* depende de las tareas descritas en el proceso software y de las herramientas de modelado. El proceso software es muy simple y se describe en lenguaje natural, aunque se está trabajando en un proceso más detallado. Por otra parte, las herramientas de verificación específicas de ALBERT sólo incluyen el animador de especificaciones [Dubois *et al.* 1994b]. Con ello, la verificación de *propiedades sociales* parece confiarse casi por completo a procesos manuales.

El conocimiento experto en ALBERT puede considerarse cristalizado en su arquitectura de agentes y SMAs y en sus lenguajes, al igual que ocurría con la propuesta del AAII/BDI (ver sección 2.3.1). A este respecto hay que señalar que el formalismo de ALBERT es conceptualmente más rico que el de AAII/BDI. ALBERT incluye, por ejemplo, constructoras específicas para indicar obligaciones, la percepción del agente o relaciones de causalidad entre acciones. No obstante, en ambos casos han de ser los desarrolladores los que determinen las *propiedades sociales* y el modo de usarlas ya que las metodologías carecen de librerías de propiedades aplicables en el desarrollo.

Estas limitaciones de ALBERT resultan muy similares a las de AAII/BDI. Una vez más indicamos que en la propuesta con TA se ha optado por emplear un lenguaje más limitado, con conceptos sociales que resulten más cercanos a los clientes, y una representación gráfica que facilite su comprensión. De esta manera se potencia una colaboración real entre los desarrolladores, que habrán de construir el sistema, y los clientes, que son los que comprenden el dominio real del problema. Por otra parte, nuestro uso de conocimiento experto se centra en librerías de *propiedades sociales*. Estos repositorios dan flexibilidad en cuanto a las propiedades consideradas, ya que

pueden ser modificados por los desarrolladores según los proyectos y su experiencia. En el estado actual de la investigación se incluyen librerías de captura de requisitos y contradicciones extraídas de la TA.

El problema de la guía del proceso software es resuelto mediante las *contradicciones* en la propuesta con la TA. La detección y solución de *contradicciones* en las especificaciones da las pautas para la evolución de las mismas.

2.3.4. i*

i* [Yu 1999] es un marco de modelado concebido para razonar acerca de sistemas intencionales con características sociales y organizativas complejas. Trata de representar cómo las personas se expresan y razonan acerca de motivaciones, recompensas, diferentes formas de alcanzar objetivos y relaciones sociales.

El concepto central de i* son los *Actores*. Estos son entidades semiautónomas, en el sentido de que no son completamente conocibles (no se puede razonar cuantitativamente sobre su comportamiento) o controlables. Los actores tienen objetivos, creencias, habilidades y compromisos, y dependen entre sí para alcanzar sus objetivos, realizar tareas o conseguir recursos. En i* se distinguen dos tipos de objetivos: *hard* (aquellos que pueden ser evaluados cuantitativamente) y *soft* (que sólo puede evaluarse cualitativamente). Para poder trabajar con estos objetivos *soft* se definen relaciones de contribución.

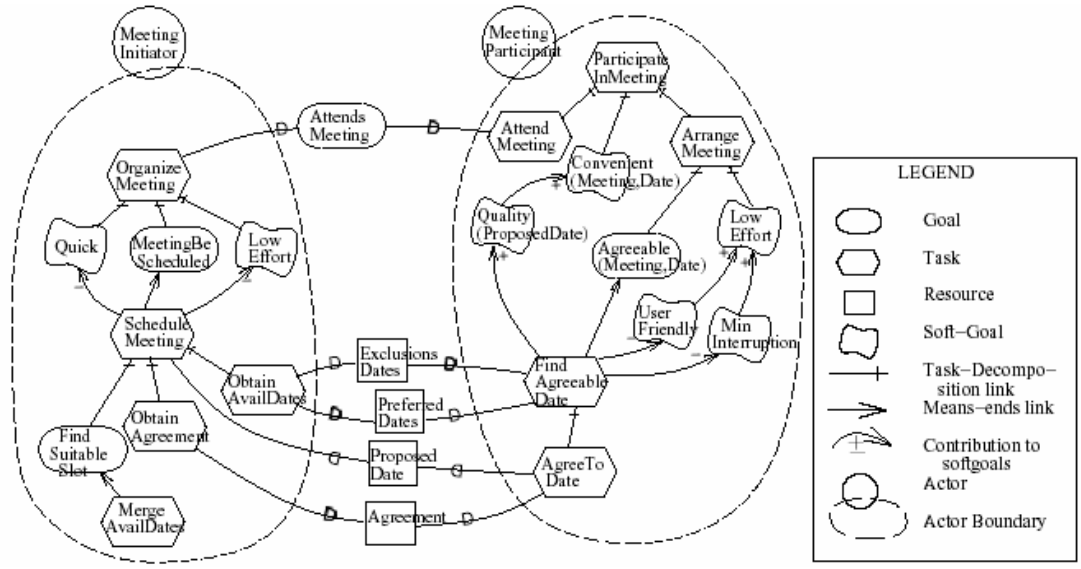


Fig. 2. Modelo de Razonamiento Estratégico de i*.

El trabajo para describir un sistema con i* [Yu & Mylopoulos 1994] comienza representando actores y sus objetivos, con enlaces entre ellos para representar que un actor depende de otro en alguna manera para conseguir alcanzar un objetivo. Esta información se recoge en los Modelos de Dependencia Estratégica ("*Strategic*

Dependency Model"). Después se determina cómo estos objetivos pueden realmente ser satisfechos mediante las contribuciones de otros actores y la utilización de recursos. Los diagramas asociados a esta información son los Modelos de Razonamiento Estratégico ("*Strategic Rationale Model*") (ver Fig. 2).

i* ha tenido una amplia repercusión en la ISOA. Sus conceptos han sido embebidos en un proceso software completo llamado Tropos [Perini *et al.* 2001] (ver sección 2.3.6). A continuación se analiza su tratamiento de las *propiedades sociales*.

i* emplea un lenguaje gráfico de modelado. Su principal ventaja es el carácter intuitivo de sus conceptos y la simplicidad de su utilización. Abstracciones como actores, objetivos o dependencias son fácilmente comprendidas porque representan conceptos asimilables a los que se encuentran en el mundo real de clientes y desarrolladores. Sin embargo, éste mismo lenguaje es demasiado limitado:

- *Primitivas insuficientes*. i* posee un conjunto demasiado reducido de primitivas y carece de mecanismos de extensión, por lo que no permite representar muchos aspectos de las *propiedades sociales*. Por ejemplo, sólo contempla relaciones de descomposición y dependencia.
- *Falta de mecanismos de abstracción*. Cuando i* aborda problemas reales genera unos pocos diagramas complejos que resultan difícilmente comprensibles.
- *Especificaciones de grano demasiado grueso*. Con las primitivas disponibles no es posible reflejar aspectos como las interacciones necesarias para satisfacer los objetivos.

En cuanto a sus procesos, pese a que i* está orientado a la captura de requisitos no ofrece guías regladas para llevar a cabo esta actividad. Su ayuda en esta fase se limita a proporcionar un vocabulario y diagramas para el modelado de requisitos y a una somera descripción de algunas tareas. Acerca de la comprobación de propiedades, i* se limita a aquellas verificaciones que se programen en su herramienta OME [OME 2000]. i* tampoco incluye conocimiento experto para trabajar con las *propiedades sociales*.

Como conclusión, señalar que i* comparte ciertas similitudes con nuestra propuesta. También usa un vocabulario reducido, centrado en las abstracciones sociales y representado mediante un lenguaje gráfico. Como diferencias, las técnicas basadas en la TA de esta tesis sí incluyen guías para la elaboración de las especificaciones (mediante el tratamiento de las contradicciones), mecanismos de razonamiento sobre las *propiedades sociales* (gracias a un proceso de detección y cambio de las especificaciones según dichas propiedades) y contemplan conocimiento experto (fundamentado en la investigación de la TA).

2.3.5. DESIRE

DESIRE [Brazier *et al.* 1997] es un marco de diseño y especificación que permite construir agentes basándose en la composición recursiva de tareas interconectadas. Su semántica formal se basa en una lógica modal temporal. DESIRE incluye un método de desarrollo y herramientas de soporte.

El concepto básico en DESIRE es la tarea. Una tarea se caracteriza en términos de las estructuras de conocimiento que necesita como entradas y produce como salidas.

Esta información de entrada/salida de las tareas se describe genéricamente como predicados lógicos. La información producida durante una ejecución concreta es representada mediante instancias de los predicados anteriores. El intercambio de información entre las tareas se representa con *enlaces de información* que indican qué átomos específicos de los predicados de entrada/salida son intercambiados y las condiciones de activación del enlace.

Sobre este modelado de tareas, DESIRE describe los agentes y el propio SMA como conjuntos de tareas interconectadas organizadas jerárquicamente. La interacción y coordinación entre agentes se especifica en términos de intercambios de información y dependencias de control.

DESIRE es uno de los trabajos más elaborados del área de agentes. Cuenta con un ciclo de desarrollo completo, herramientas, cursos de entrenamiento y ejemplos de uso. Constituye por tanto una opción muy completa para el desarrollo de SMAs. No obstante, su tratamiento de las *propiedades sociales* de los SMAs presenta algunos inconvenientes.

El lenguaje de DESIRE se fundamente en la lógica modal temporal. Los mismos comentarios acerca de la dificultad en el manejo de *propiedades sociales* con formalismos, su capacidad de análisis o la integración del tratamiento acerca de AAII/BDI (ver sección 2.3.1) o ALBERT (ver sección 2.3.3) se aplican en este caso. También incluye un lenguaje gráfico para representar la estructura del SMA (e.g. tareas existentes, conexiones entre ellas o componentes de un agente).

En DESIRE, los propios autores hacen patente una limitación frecuente del binomio lenguaje-arquitectura en las metodologías. Esta limitación consiste en que los lenguajes propuestos tienen una estrecha relación con la arquitectura planteada en la metodología para agentes y SMAs. En el caso de DESIRE se fija una arquitectura específica para los agentes basada en tareas. Si bien ello permite dar un mayor soporte a lo largo del proceso también exige adoptar su filosofía sobre qué es un agente y un SMA y cómo deben construirse. Este problema ha sido abordado ya por algunas propuestas (como las metodologías INGENIAS (ver sección 2.3.8) o MaSE [DeLoach 2001]), aunque centrándose sobre todo en la implementación para diferentes plataformas.

Siguiendo con la revisión encontramos que DESIRE no proporciona técnicas específicas de captura de requisitos. Los desarrolladores realizan esta tarea usando las herramientas que consideren convenientes y plasman los resultados en la notación de DESIRE. En cuanto a las técnicas de comprobación disponibles para las *propiedades sociales*, DESIRE propone un método de verificación compositivo basado en su formalismo, donde se correlaciona la verificación de las propiedades del SMA con la de sus componentes [Brazier *et al.* 1994].

DESIRE es otro caso que tampoco incluye conocimiento experto para sobre las *propiedades sociales*, salvo el que pueda considerarse incorporado en su arquitectura.

Como conclusión, decir que esta revisión de DESIRE ha mostrado un nuevo problema al tratar de exportar la experiencia de una metodología a otras: el estrecho acoplamiento entre las técnicas propuestas y la arquitectura planteada. Nuestra aproximación corrige este inconveniente mediante el establecimiento de correspondencias entre los vocabularios de modelado. Ello permite emplear las técnicas basadas en TA sobre cualquier metodología basada en agentes, como veremos más adelante en el Capítulo 3.

2.3.6. Tropos

Tropos [Perini *et al.* 2001] es una metodología software basada en agentes donde la etapa de adquisición de requisitos inspira los procesos empleados en las demás fases del desarrollo. Éste comienza con los requisitos iniciales y continúa descomponiéndolos hasta que son reducidos a colecciones alternativas de decisiones de diseño, cada una de las cuales es capaz de satisfacer los objetivos iniciales.

Tropos está basado en i* (ver sección 2.3.4) y tiene influencias de KAOS (ver sección 2.3.2). Su lenguaje de modelado está constituido por los conceptos de i* embebidos en un marco con primitivas adicionales para soportar la definición del contexto, generalización, agregación y clasificación [Sannicolo' *et al.* 2002]. Los conceptos del lenguaje se emplean en tres tipos de representaciones: diagramas [Castro *et al.* 2001], anotaciones parcialmente formales y especificaciones formales [Fuxman *et al.* 2001]. Las anotaciones parcialmente formales pueden ayudar en la definición de algunos aspectos de los artefactos del proceso. Las especificaciones formales se usan como base para un amplio rango de técnicas de análisis, incluyendo pruebas de corrección, simulación de procesos y análisis de objetivos.

Tropos es la evolución de i* hacía una metodología de desarrollo de sistemas de agentes dirigida por requisitos. Por ello ha tratado de resolver parte de los problemas anteriormente señalados para i* con las *propiedades sociales* (ver sección 2.3.4).

Su lenguaje visual basado en i* se complementa con una lógica temporal modal, *Formal Tropos*. El lenguaje visual facilita el modelado rápido del sistema con un vocabulario intuitivo y puede ser traducido automáticamente al lenguaje formal. Por otra parte, el formalismo proporciona un marco suficientemente rico como para describir y tratar de forma integrada las *propiedades sociales*. Aunque la incorporación de la notación formal solventa el problema del nivel de detalle señalado para i*, existen otros aspectos sin resolver. El lenguaje sigue sin estructurar la información para facilitar el manejo de grandes especificaciones y existen aspectos de las *propiedades sociales* que no pueden ser representados sin constructoras adicionales. Dada la complejidad de los SMAs que se están modelando, resulta muy difícil abordar el examen de unas especificaciones estructuradas sólo en unos pocos diagramas con un gran número de elementos.

La verificación de propiedades en *Tropos* se basa en el lenguaje formal y se realiza mediante técnicas de comprobación de modelos ("*model checking*") con el verificador NuSMV [Cimatti *et al.* 2000]. Para ejemplos de propiedades y verificaciones véase [Fuxman *et al.* 2001]. En cuanto a la captura de requisitos, la falta de procesos guiados ya señalada en i* sigue sin ser abordada.

Tropos no incluye fuentes de conocimiento experto para la verificación de propiedades. Las propiedades que muestran sus casos de estudio proceden de los requisitos del cliente o del conocimiento y experiencia del equipo de desarrollo. En ningún caso se incluyen *propiedades sociales* complejas que emanan de otros campos de conocimiento, por ejemplo acerca de la organización de la comunidad o las teorías cognitivas. Es decir, no se introducen elementos de análisis que no formen parte de la información ya manejada en el equipo de desarrollo.

La presentación de Tropos ha hecho hincapié en un nuevo problema: manejar la complejidad de las especificaciones de los SMAs. Las propuestas para trabajar con las *propiedades sociales* han de contemplar mecanismos de abstracción que permitan

examinar sólo aquellas porciones de las especificaciones requeridas para cada tarea y al nivel de detalle necesario. La TA contempla este problema a través de varios niveles de análisis que abarcan desde las tareas más simples, realizadas por el individuo de forma casi automática, hasta las actividades complejas que involucran a organizaciones completas. En cada nivel, los desarrolladores pueden especificar el contexto que ha de considerarse para la actividad, y que puede incluir tanto aspectos propios de la actividad como impuestos por el contexto social. Estas abstracciones y procesos de análisis son incorporados en la presente propuesta para trabajar con SMAs.

2.3.7. ADELFE

ADELFE [Bernon *et al.* 2002] es una metodología para el desarrollo de SMAs adaptativos. ADELFE tiene una marcada visión social del sistema en la línea del trabajo de [Ferber 1999]. Más precisamente, esta metodología trata de construir SMAs donde los agentes que los constituyen no son adaptativos por sí mismos pero sí lo es su sociedad. ADELFE está basada en enfoques metodológicos de la Ingeniería del Software Orientada a Objetos: su proceso sigue el USDP ("*Unified Software Development Process*") [Jacobson *et al.* 1999] y su notación usa UML [OMG 2003] y AUML [Bauer *et al.* 2001].

El ciclo de desarrollo de ADELFE contempla las fases de captura de requisitos, análisis, diseño e implementación-prueba. La principal contribución de ADELFE a estos flujos de trabajo es el estudio en la fase de diseño de las llamadas *Situaciones No Cooperativas* ("*Non Cooperative Situation*"). Las *Situaciones No Cooperativas* (NCS) son aquellas en las que el agente reconoce fallos en la cooperación. El tratamiento de las NCSs permite a ADELFE modelar SMAs capaces de adaptarse al nivel de la sociedad de agentes a situaciones imprevistas en el entorno. Un agente puede detectar tres tipos de NCSs:

- Cuando una señal del entorno no es leída ni comprendida sin ambigüedad.
- Cuando la información percibida no induce al agente a un proceso de actividad.
- Cuando las conclusiones que extrae el agente le inducen a actuar sobre el entorno de tal manera que no contribuye a sus objetivos.

En la fase de diseño de ADELFE se rellena en lenguaje natural una tabla por cada NCS identificada en el análisis. La tabla indica cómo debe el agente tratar dicha NCS e incluye su nombre, el estado en el que el agente la puede detectar, una descripción textual de la misma y un conjunto de condiciones y acciones asociadas que permiten al agente eliminar la NCS.

Con esta breve introducción a ADELFE se puede pasar a analizar su tratamiento general de las *propiedades sociales* del SMA.

El lenguaje de ADELFE está basado en la orientación a objetos con los elementos añadidos por AUML. Las *propiedades sociales* han de ser modeladas por tanto en términos de casos de uso, las NCSs y diagramas de clases, objetos, actividad y agentes. Esta representación permite modelar fácilmente la estructura del sistema, sobre todo en cuanto a componentes software, y las interacciones entre agentes. Sin embargo aspectos como las intenciones de los agentes o las normas de la sociedad

sólo pueden ser expresados en lenguaje natural, sin que ADELFE suministre una estructura o conjunto de conceptos precisos. En cuanto a los mecanismos de razonamiento sobre estas especificaciones no se cita ninguno en el trabajo revisado.

Acerca de la captura de requisitos, ADELFE sigue los flujos del USDP. Por tanto se limita a dar ciertas guías en lenguaje natural y diagramas de actividades sobre cómo capturar el conocimiento. Los requisitos se reflejan en la misma notación empleada en el resto del proceso software, que está sesgada hacia el desarrollo de agentes fundamentado en la orientación a objetos. Esta clase de notaciones basadas en un paradigma de desarrollo hace difícil para los clientes comprender dichos requisitos, ya que no están habituados a estos conceptos.

El trabajo con las *propiedades sociales* en las especificaciones se realiza manualmente aunque en [Bernon *et al.* 2002] se habla de la futura disponibilidad de una herramienta de modelado.

El conocimiento experto manejado por ADELFE se centra en las NCSs. La propuesta sobre las NCSs comparte algunas similitudes con las de esta tesis acerca del manejo de *contradicciones*. En ambos casos se trata de detectar situaciones conflictivas en el SMA y dar pautas para su solución. Además, ambas aproximaciones incorporan un catálogo con algunas de las situaciones contradictorias más comunes que pueden ser reutilizadas en distintas aplicaciones. Las diferencias entre ambos enfoques radican en la fuente de las contradicciones, su foco y el tratamiento. Las NCSs de ADELFE emergen del modelado de las entidades software como agentes; es un enfoque orientado a los desarrolladores que trata de analizar el comportamiento y percepción de los resolutotes de problemas del sistema a nivel individual. Por el contrario, las contradicciones basadas en la TA pretenden detectar los conflictos relativos a las intenciones y al nivel social. El análisis se basa en patrones que surgen del estudio de sociedades humanas reales en Sociología y Psicología, donde aspectos como las interacciones entre los individuos, la motivación, la evolución o el aprendizaje están altamente elaborados. Finalmente, la descripción de las NCSs se basa en lenguaje natural. Ello hace difícil su aplicación de manera automatizada sobre el proceso software. La presente propuesta ha optado por representar esta información como patrones estructurales que permiten un tratamiento semi-automatizado de las contradicciones y sus soluciones.

2.3.8. INGENIAS

INGENIAS [Pavón & Gómez-Sanz 2003] es una metodología para el desarrollo de SMAs. Al igual que en ADELFE (ver sección 2.3.7), INGENIAS considera que el SMA ha de ser estudiado como un todo, por lo que el concepto de sociedad [Ferber 1999] adquiere una gran importancia.

El trabajo de INGENIAS es continuación del desarrollado en MESSAGE [Caire *et al.* 2001]. Siguiendo esa propuesta, INGENIAS se basa en metamodelos que definen las primitivas y las propiedades sintácticas y semánticas de los modelos. Por tanto, los metamodelos definen cómo debe ser el sistema generado. INGENIAS descompone el estudio del SMA en cinco vistas definidas por otros tantos metamodelos.

INGENIAS cuenta con una herramienta de desarrollo, el *INGENIAS Development Kit* (IDK) (*http://ingenias.sourceforge.net*). Se trata de un entorno para el modelado

según INGENIAS que soporta la incorporación de nuevos módulos. Dichos módulos son capaces tanto de leer como de manipular las especificaciones en la herramienta.

Estrechamente relacionado con el IDK está el proceso de implementación de INGENIAS. El proceso se basa en el uso de plantillas dependientes de la plataforma de destino. Las plantillas se instancian según los elementos en las especificaciones y de este modo pueden dar lugar a varias implementaciones de forma casi automática.

A su vez, el proceso de implementación de INGENIAS se relaciona con sus propuestas de comprobación de propiedades. Los desarrolladores han de generar módulos para aquellos aspectos del sistema que traten de verificar. Dichos módulos obtendrán la información de las especificaciones y comprobarán las propiedades que tengan codificadas, quizás con la ayuda de herramientas externas. Éste es el caso de la simulación de protocolos entre agentes con JADE [Bellifemine *et al.* 2001], donde el correspondiente módulo comprueba que están definidas las interacciones pertinentes antes de generar el código para el protocolo. Ésta es una solución que sólo es accesible a los desarrolladores y que exige desarrollar soluciones *ad-hoc* para cada propiedad. Como ventaja, los desarrolladores sólo han de conocer las APIs del IDK para construir los módulos. El lenguaje de los módulos (i.e. Java) puede resultar más sencillo de usar para los desarrolladores que algunos formalismos.

A continuación, pasamos a considerar el tratamiento de las *propiedades sociales*.

INGENIAS presenta un lenguaje gráfico para el modelado definido mediante un metamodelo. Destaca su atención a las organizaciones de agentes con primitivas para modelar éstas, sus creencias y su influencia sobre los agentes individuales. Éste es un aspecto frecuentemente olvidado en otras propuestas. También contempla otras *propiedades sociales* como la motivación de los individuos en una colaboración, la satisfacción de objetivos o las relaciones de poder. Como inconveniente, la especificación de grano fino en INGENIAS se realiza en lenguaje natural, aunque se recomienda utilizar un pseudo-código Java. De todos modos, no hay indicaciones sobre cómo relacionar ambas especificaciones para realizar comprobaciones.

Como ocurre en otras metodologías (por ejemplo en ALBERT en la sección 2.3.3 o i* de la sección 2.3.4), la captura de requisitos y la comprobación de propiedades de INGENIAS se confía a las tareas de su proceso software y a la herramienta. Las tareas del proceso están descritas con lenguaje natural y diagramas de actividades. Como ocurre en otras propuestas, INGENIAS no da guías precisas para identificar sus entidades de modelado con los conceptos del mundo real. Por último, tampoco INGENIAS contempla conocimiento experto explícito sobre las *propiedades sociales*.

Frente a la ausencia de guías concretas en casi todas las propuestas de la ISOA sobre el modelado de la información suministrada por los clientes, la aproximación con TA presenta repositorios de configuraciones sociales que asocian preguntas en lenguaje natural y patrones estructurales en UML. De este modo se proporciona ayuda para establecer la correspondencia entre las respuestas en lenguaje natural a las preguntas y estructuras del lenguaje de modelado.

Para facilitar la modificación de las primitivas relacionadas con la TA se ha optado por describir su lenguaje mediante metamodelos. Así se permite adaptar el lenguaje a nuevo conocimiento relativo a problemas concretos u otras disciplinas.

2.4. Ciencias Sociales

En este apartado se examinan algunas técnicas usadas en el desarrollo de software cuyo origen se encuentra en las ciencias sociales. La búsqueda de nuevas técnicas de ayuda para la Ingeniería del Software en estas disciplinas tiene su origen en el reconocimiento de que los requisitos de los sistemas deben contemplar su contexto social como un elemento más [Lamsweerde 2000, Nuseibeh & Easterbrook 2000]. Sin embargo, el contexto social es diferente de los habituales artefactos en la Ingeniería del Software y su estudio requiere nuevos métodos [Winograd & Flores 1986].

Recurrir a las ciencias sociales para tratar el contexto social de los sistemas software resulta aún más natural en la ISOA, porque la metáfora de los agentes y los SMAs resulta muy próxima a las organizaciones humanas estudiadas por esas disciplinas. Como ya se argumentó en el Capítulo 1, los propios SMAs se describen parcialmente en términos de contextos sociales donde actores intencionales interactúan para alcanzar sus objetivos.

Los ejemplos de este apartado mostrarán que las técnicas adaptadas de las ciencias sociales facilitan el trabajo con las *propiedades sociales* de los sistemas software mediante técnicas de captura, tratamiento y comprobación de las mismas. En general se tata de propuestas que, desde el punto de vista de la Ingeniería del Software, podrían clasificarse como metodologías *convencionales* [1] [Fensel 1995]. El motivo es que usan el lenguaje natural y gráficos arbitrarios para representar su información y procesos. La riqueza del lenguaje subyacente les permite trabajar con aspectos que serían de difícil descripción con notaciones más estructuradas.

Como ejemplos de la aplicación de técnicas procedentes de las ciencias sociales en la Ingeniería del Software se verán los trabajos con Etnografía de [Hughes *et al.* 1994] y el Diseño Cooperativo [Bødker *et al.* 1998]. Aunque los ejemplos descritos no proceden de la ISOA, su forma de trabajo y resultados sí podrían ser aplicados en un proceso software para SMAs. Se trata de técnicas orientadas hacia la captura de requisitos y con características para la validación de estos en el producto software.

2.4.1. Etnografía

La Etnografía se encuentra entre las técnicas propuestas para el estudio del contexto social en el desarrollo de sistemas software. Dentro de la investigación en Ingeniería del Software, su utilización es más evidente en ciertos campos como la Ingeniería de Requisitos [Goguen 1992], el CSCW ("*Computer Supported Cooperative Work*") [Hughes *et al.* 1994] o el HCI ("*Human-Computer Interaction*") [Hutchins & Klausen 1992].

Según definiciones clásicas, la Etnografía es "*el estudio empírico de la cultura y la organización social de las poblaciones humanas*" [Ellen 1984]. "*Los etnógrafos tratan de describir otra cultura tan precisamente como sea posible basándose en la*

[1] El término original en el trabajo de Fensel es "*informal specification*". Aquí se ha optado por traducirlo como "especificación convencional" o "método convencional".

perspectiva de los nativos" [Werner & Shoepfle 1987]. Es una disciplina dedicada a la observación, la captura de información y la descripción de actividades humanas.

La Etnografía se centra en el estudio de las situaciones tal y como ocurren en la realidad presente. El desarrollo de software, por el contrario, se centra en la creación de nuevos sistemas. No obstante, la Etnografía resulta de utilidad para comprender el contexto humano del nuevo sistema, con aspectos como las habilidades de los usuarios, la manera en que interactúan o el posible rediseño de su forma de trabajo.

1 Doctor Susanna: Brainview, listen, there is some bigger problem now-
2 Designer Timo: Oh- *(stops walking for a while)*
3 Doctor Susanna: My measurements just do not work out at all.
4 Designer Timo: Well, let's see- *(sits down at the workstation, taps the keyboard)*
5 Doctor Susanna: The channels go out of tune all the time in the middle of the measurement. *(sits down near Timo)*
6 Designer Timo: As far as I can see, the picture is clear. *(taps the keyboard, looks at the display)*
7 Doctor Susanna: Not in the beginning, when the measurement is only getting started, and when they *(the channels)* are being tuned up in the empty room- *(points to the shielded room. Pekka, the other designer, walks in)*
8 Designer Timo: Aha. *(stands up, turns around and walks to the shielded room)*
9 Doctor Susanna: But then they *(the channels)* all of a sudden go bad. - And when they are being tuned up again, they may soon go out of tune again. *(Telephone rings. Pekka sits down to the workstation. Timo comes out of the shielded room. Timo and Pekka are talking together. Pekka opens a box and takes out a circuit board)*
10 Designer Timo (to designer Pekka): Well, maybe you can change it now.
11 Designer Pekka: I'll do it right away.
12 Doctor Susanna: But this time the troubles started immediately. *(Timo and Pekka are talking together, overlapping Susanna's speech. Pekka walks out, Timo sits down at the workstation.)*
13 Doctor Susanna: And now that *(the software)* has broken down also.
14 Designer Timo: *inaudible (at the same time stands up to leave)*
15 Doctor Susanna *(raising her voice)*: Hey! Look here now!
16 Designer Timo *(Timo stops and turns to Susanna)*: Well, yes-
17 Doctor Susanna: When I was doing them- hey come on now, you must take this seriously - it just can't go on that things don't work out! *(Timo sits down and starts to tie his shoelaces)*
18 Doctor Susanna: Every time the same thing, this has happened many times over.

Fig. 3. Ejemplo de registro etnográfico de una observación de participantes.

Los análisis etnográficos se basan en descripciones detalladas de las actividades cotidianas de los actores sociales en contextos concretos. Estas descripciones se realizan en lenguaje natural, dotado en ocasiones de cierta estructura gracias al uso de plantillas para documentos. El lenguaje natural se complementa con otros recursos como gráficos, grabaciones de audio y video, o instrumentos empleados en la actividad de los sujetos. Un ejemplo de esta clase de descripciones puede encontrarse en [Hasu & Engeström 2000], del cual se ha extraído el fragmento del registro de una conversación recogido en la Fig. 3. Obsérvese que la trascripción no sólo incluye la

comunicación verbal entre los participantes sino que también recoge su comunicación no verbal, las tareas en curso durante la conversación o datos del entorno.

Los métodos tradicionales del etnógrafo para obtener las descripciones de actividades incluyen la observación de los participantes en su entorno habitual, la investigación de archivos, las entrevistas y los cuestionarios. En general, los datos sobre una determinada actividad a estudiar se recogen usando varios de estos métodos. Luego los investigadores cruzan e interpretan dichos datos para confirmar la información y detectar inconsistencias. Esta interpretación de los datos depende de varios factores. En primer lugar, depende de las teorías sociales con las que se trabaje (e.g. TA [Vygotsky 1978] o Etnometodología [Livingstone 1988]). La Etnografía es sobre todo una disciplina de observación y no presupone un modelo teórico con el que interpretar las actividades observadas. Los otros factores que influyen en la interpretación de los datos son la habilidad y rigor de los investigadores. Esto se debe a que los procesos de la Etnografía carecen de guías regladas de actuación.

	Etnografía Concurrente	**Etnografía Rápida**	**Etnografía Evaluativa**	**Etnografía de Revisión**
Nivel de detalle	Según el foco del estudio	Comprensión básica	Depende del modelo inicial	Según estudios previos
Información para el desarrollo	Requisitos del prototipo durante el ciclo de desarrollo	Revisión del dominio de trabajo para un prototipo inicial	Implicaciones para el entorno de un modelo inicial	Motivación y ámbito del diseño
Duración del trabajo de campo	12-14 meses. simultáneos a la captura de requisitos	2-3 semanas de estudio previas al análisis	2-4 semanas de estudio con el modelo original antes de la revisión	Sin trabajo de campo, sólo revisión de los estudios previos
Influencia del lugar del estudio	Fuerte e impredecible	Mayor libertad para elegir el lugar	Según el lugar de estudio del modelo previo	No aplicable
Relación estudio-desarrollo	Desarrollo conducido por el estudio	Desarrollo conducido por el estudio	Estudio guiado por el modelo inicial	Estudio según el esquema del diseño
Forma del sistema	Sistemas para trabajo interactivo. Énfasis en el detalle	Sistemas para trabajo interactivo. Énfasis en la estructura básica	Sistemas para trabajo interactivo. Énfasis en la estructura básica	Plataformas generales de soporte para un rango de aplicaciones diferentes

Tabla 1. Roles de la Etnografía en el desarrollo de software.

Sobre el uso de la Etnografía aplicado al desarrollo de software, éste adopta distintas formas. Una revisión de posibles usos y sus características extraída de [Hughes *et al.* 1994] puede verse en la Tabla 1. Como se desprende de la tabla, la

Etnografía pude ser aplicada tanto a la captura de requisitos (i.e. Etnografía Concurrente, Rápida y de Revisión) como a la comprobación de propiedades (i.e. Etnografía Evaluativa). En estos usos de la Etnografía, las *propiedades sociales* aparecen como enunciados en lenguaje natural acerca de situaciones de los individuos o sociedades relativos, por ejemplo, a sus motivos, su interacción o sus relaciones.

En base a esta breve revisión sobre la Etnografía, se puede analizar su utilidad en el análisis de las *propiedades sociales* de SMAs.

La Etnografía usa en sus descripciones el lenguaje natural y gráficos arbitrarios. Como se ha indicado, éste es también el lenguaje con el que se expresan las propiedades a comprobar. Aquí surgen varios problemas relacionados con la interpretación de estas descripciones:

- *Argot propio*. El lenguaje usado por los expertos en Etnografía es complejo y con una gran densidad semántica. Es decir, el vocabulario tiene significados muy precisos que pueden diferir en matices mínimos.
- *Lenguaje de los actores observados*. La comprensión de la situación observada depende de la interpretación de su lenguaje por personas externas a esa situación.

Además, el uso del lenguaje natural dificulta contar con herramientas de ayuda automatizadas y la traducción de la información a lenguajes de diseño.

En cuanto a los procesos, los métodos de la Etnografía establecen *propiedades sociales* a verificar, realizan observaciones y las analizan para comprobar si las hipótesis se cumplen. Los principales inconvenientes de los métodos etnográficos para el diseño de software son:

- *Falta de guía reglada*. Las decisiones sobre el proceso a seguir dependen casi por completo del criterio del investigador, ya que no existen guías claras. Por ello exigen un conocimiento avanzado de la disciplina.
- *Necesidad de expertos para interpretar las observaciones*. El significado de la situación observada depende de la teoría con que la interpreta un experto.
- *Resultados no orientados al diseño*. Se trata de presentaciones discursivas y muy prolijas, que no están dirigidas a las necesidades de los desarrolladores y que no reflejan claramente las decisiones de diseño.

Los estudios de la Etnografía pueden servir como fuente de conocimiento experto en el tratamiento de las *propiedades sociales*. Estos estudios incluyen tanto observaciones generales sobre grupos humanos como casos concretos.

Para concluir señalar que esta tesis comparte algunos de los supuestos de la Etnografía: ambas propuestas prestan atención a la información implícita en la interacción, representada por patrones de comportamiento complejos. En este análisis resaltan la importancia de examinar los antecedentes como fuente de conocimiento sobre la situación. Como diferencias, esta tesis se centra en el proceso de desarrollo de SMAs y adopta las convenciones propias de la Ingeniería del Software. El análisis y el conocimiento experto se aplican sobre especificaciones de SMAs, mucho menos complejas que las organizaciones humanas estudiadas por la Etnografía. Por ello se puede adoptar una visión simplificada del problema, accesible para no expertos en las ciencias sociales. La presente propuesta representa el conocimiento sobre *propiedades sociales* mediante patrones estructurales descritos en UML y usa procesos de detección de patrones en vez de recurrir a la interpretación de expertos.

2.4.2. Diseño Cooperativo

El Diseño Cooperativo [Bødker *et al.* 1998] es una aproximación a la captura de requisitos fundamentada en la implicación temprana y activa de clientes y usuarios en el diseño y evaluación de prototipos del sistema.

El marco teórico del Diseño Cooperativo se fundamenta en la TA [Vygotsky 1978], concretamente en los trabajos de [Bisgaard *et al.* 1989] y [Engeström 1990] acerca del aprendizaje, la construcción de nuevos objetos colaborativamente y el papel de los conflictos en estos procesos. La idea fundamental es que la actividad de diseño es una actividad de aprendizaje. Su estudio debe provocar rupturas en la visión actual que tienen los usuarios y los diseñadores de las tareas, induciendo en el proceso nuevos instrumentos de trabajo.

A juicio de los investigadores del Diseño Cooperativo [Bødker & Grønbæk 1996], la mayoría de los desarrollos de prototipos son dirigidos y orientados exclusivamente por los desarrolladores. La participación de los clientes queda reducida a meras fuentes de información sobre el dominio del problema. Esta clase de prácticas da lugar a una implicación de los usuarios en el desarrollo limitada a la evaluación pasiva del trabajo de los desarrolladores; esta evaluación se basa en demostraciones y comprobaciones de que los programas satisfacen o no las especificaciones.

En contraste con las prácticas anteriores, el Diseño Cooperativo pretende involucrar a los usuarios en el desarrollo del sistema desde los primeros momentos, haciéndoles partícipes en la concepción del mismo. Se busca una confrontación de las diferentes prácticas en la persecución de un mismo objetivo, el desarrollo de la aplicación. Esta forma de trabajo se apoya en el uso de herramientas de diseño flexibles, basadas en computadoras, que soportan la manipulación directa del diseño y permiten simular la funcionalidad del sistema. Estas herramientas permiten establecer un estrecho acoplamiento entre la modificación y desarrollo del prototipo y su evaluación mediante sesiones que representan el trabajo real de los usuarios. La información sobre estas sesiones se recoge en notas y grabaciones. Posteriormente se realizan análisis etnográficos de estas fuentes (como los vistos en la sección 2.4.1) guiados por la TA. Con estos estudios se obtiene información adicional acerca del sistema y su interacción con el entorno, así como modificaciones a los presentes requisitos y nuevas peticiones. Esta estrategia da lugar a un desarrollo iterativo dirigido por las sesiones de trabajo conjuntas entre usuarios y desarrolladores.

Desde el punto de vista del estudio de las *propiedades sociales*, el Diseño Cooperativo ofrece varias características de interés.

Al igual que en otras propuestas sociales, el lenguaje natural juega un papel primordial. Sin embargo, el uso de prototipos de aplicaciones permite obtener requisitos en otros formatos más cercanos a la Ingeniería del Software, como representaciones HTML o programas en lenguajes de *scripts*. Estas representaciones plasman aspectos como acuerdos, división del trabajo, conocimientos y tareas de los clientes. Por ello puede considerase que describen *propiedades sociales* del entorno.

Sobre sus procesos, el Diseño Cooperativo sigue las pautas de la TA y adopta un proceso para el desarrollo de sistemas basado en el estudio y resolución de contradicciones en las actividades de trabajo y diseño. De este modo cuenta con una guía para hacer evolucionar el sistema a partir de los datos sobre el trabajo capturados con estudios etnográficos.

El Diseño Cooperativo presenta también dos importantes limitaciones:

- *Uso de la Etnografía*. Sus estudios sobre la situación de trabajo suelen realizarse con técnicas etnográficas y presentan por tanto las virtudes e inconvenientes ya vistos en la sección 2.4.1.
- *Dificultad en la ejecución de las sesiones de trabajo con los usuarios*. Las sesiones han de conducirse hacia aquellos aspectos productivos para el desarrollo sin coartar la expresión del usuario. Si ocurre esto último se pueden ignorar aspectos relevantes del trabajo habitual de los usuarios. Además, para que las sesiones se aproximen a la realidad del contexto es necesario incluir herramientas y datos que no siempre están disponibles. Por estos motivos, la calidad de los resultados del Diseño Cooperativo es muy variable en función del equipo de desarrollo, el sistema a especificar y las herramientas empleadas.

Nuestra propuesta guarda una cierta relación con el Diseño Cooperativo. En ambos casos se trata de un proceso basado en la TA y sus contradicciones que propugna la necesidad del trabajo conjunto entre clientes y desarrolladores. Por lo demás, nuestra propuesta incide en el uso de lenguajes más estructurados, como UML. De este modo se pueden proporcionar herramientas de ayuda semi-automatizadas para detectar contradicciones. A fin de evitar el conflicto entre dirigir al usuario y que éste realice su labor normal, nuestras herramientas se conciben como asistentes invocados por los usuarios en su proceso de desarrollo cuando desean contar con la ayuda de la TA.

2.5. Conclusiones

En la introducción y en este capítulo se ha argumentado como las llamadas *propiedades sociales* de los SMAs constituyen una de las aportaciones fundamentales del paradigma de agentes a la Ingeniería del Software. Estas propiedades son información relativa a los aspectos sociales e intencionales de los SMAs y su entorno. Al tratarse de una aportación original del paradigma, la ISOA ha tenido que desarrollar técnicas específicas para su tratamiento o adaptar otras ya existentes. La hipótesis con la que iniciamos este trabajo fue que el tratamiento actual de dichas *propiedades sociales* en la ISOA era insuficiente en varios aspectos:

- *Falta de conceptos específicos para representar las propiedades sociales*. Las metodologías vistas no contemplan los conceptos necesarios para cubrir alguno de los aspectos intencionales y sociales de un SMA. Esta carencia es aún más problemática en aquellas metodologías que no cuentan con mecanismos para ampliar el vocabulario de modelado con nuevos conceptos.
- *Falta de una visión global del tratamiento de las propiedades sociales*. Falta de mecanismos para trabajar con distintas propiedades considerando simultáneamente individuos y grupos.
- *Falta de métodos específicos de captura de requisitos sociales*. La captura de requisitos se basa en métodos ideados para otra clase de propiedades, que no son adecuados para obtener la información característica de estas propiedades.
- *Falta de métodos específicos de detección y resolución de conflictos sobre las propiedades sociales*. Se usan métodos para la comprobación de otra clase de

propiedades que no aprovechan el significado propio de los aspectos intencionales y sociales y sus relaciones.

- *Falta de guía en el tratamiento de las propiedades sociales*. Las guías del proceso de desarrollo se centran en los artefactos software y no prestan suficiente atención al estado del SMA como sistema intencional y social.
- *Carencia de conocimiento específico sobre las propiedades sociales*. El conocimiento para usar las *propiedades sociales* en el desarrollo del sistema sólo se recoge implícitamente en la mayoría de las metodologías (por ejemplo en el lenguaje, la arquitectura o el proceso). No existen repositorios de propiedades aplicables a los distintos proyectos según el criterio de los desarrolladores.

Para comprobar esta hipótesis se procedió a analizar varias metodologías de la ISOA acerca de estos aspectos de interés. El resumen de las conclusiones puede verse en las tablas Tabla 2, Tabla 3 y Tabla 4 a lo largo de esta sección.

Lenguaje			
Metodologías	**Lenguajes de modelado[2]**	**Adecuación**	**Capacidad de ampliación**
AAII/BDI	Gráfico + Formal Gráfico pobre	Incluye estado mental No incluye sociedades	Con el formalismo
KAOS	Gráfico + Formal Gráfico rico	Incluye estado mental No incluye sociedades	Con metamodelo y formalismo
ALBERT	Gráfico + Formal Gráfico pobre	Incluye estado mental y sociedades parcialmente	Con el formalismo
i*	Gráfico Gráfico pobre	Incluye estado mental y sociedades parcialmente	No ampliable
DESIRE	Gráfico + Formal Gráfico pobre	Incluye estado mental y sociedades parcialmente	Con el formalismo
Tropos	Gráfico + Formal Gráfico pobre	Incluye estado mental y sociedades parcialmente	Con el formalismo
ADELFE	Gráfico + Formal Gráfico pobre OO	Incluye estado mental y sociedades parcialmente	No ampliable
INGENIAS	Gráfico Gráfico rico	Incluye estado mental y sociedades	Con metamodelo
Etnografía	Natural + recursos arbitrarios	Incluye estado mental y sociedades	Lenguajes arbitrarios
Diseño Cooperativo	Natural + recursos arbitrarios + lenguajes de prototipado SW	Incluye estado mental y sociedades	Lenguajes arbitrarios

Tabla 2. Resumen del tratamiento de las *propiedades sociales* en la ISOA y en las técnicas adaptadas a la Ingeniería del Software de las ciencias sociales. Lenguajes.

[2] Los términos "pobre" y "rico" referido a un lenguaje hacen referencia a su poder expresivo.

La Tabla 2 se centra en la información relativa a los lenguajes de modelado de las metodologías. Incluye los siguientes datos:

- *Lenguajes de modelado*. Tipos de lenguajes empleados y riqueza de su vocabulario en conceptos relativos a las *propiedades sociales*.
- *Adecuación*. Aspectos intencionales y sociales contemplados en la metodología.
- *Capacidad de ampliación*. Mecanismos disponibles para modificar el vocabulario de los lenguajes.

En relación con el lenguaje (ver Tabla 2) encontramos que prácticamente todas las propuestas usan una combinación de lenguajes: gráfico, formalismo, lenguaje natural e incluso puede decirse que lenguajes de programación para los módulos de herramientas en alguna de sus técnicas. Sobre las limitaciones, hay que empezar por apuntar que ninguna de las propuestas vistas cubre todos los aspectos de las *propiedades sociales*. Si consideramos la clase de características incluidas en la literatura sobre agentes [Maes 1994, Sykara 1998, Jennings & Wooldridge 2000], vemos que las propuestas existentes contemplan específicamente sólo unos pocos aspectos. Así, AAII/BDI y KAOS no incluye la motivación de la sociedad o Tropos descuida el estado mental de los agentes y sólo habla de objetivos. En ciertos casos este problema es irresoluble por los desarrolladores ya que los lenguajes no ofrecen mecanismos de ampliación. Éste es el caso de i* y ADELFE. El resto de las propuestas se basan en un metamodelo (e.g. KAOS e INGENIAS) o un formalismo donde se pueden definir nuevos predicados (e.g. ALBERT o DESIRE). No obstante, el hecho de contar con estos mecanismos no supone disponer del conocimiento para ampliar el lenguaje con nuevos elementos relacionados con las *propiedades sociales*.

La Tabla 3 contiene información acerca de los procesos usados en las metodologías de la ISOA con las *propiedades sociales*. Los aspectos considerados son:

- *Guía*. Se estudia la forma en que indican dónde modificar las especificaciones, el motivo por el que hacerlo y qué procesos emplear.
- *Requisitos*. Métodos específicos para captura de requisitos sociales.
- *Comprobación*. Métodos para la comprobación de *propiedades sociales* y la modificación de las especificaciones como consecuencia de la aparición de esas propiedades.

Siguiendo con el análisis de las tablas, se considera ahora la disponibilidad de métodos específicos para la captura de requisitos y la comprobación de *propiedades sociales* (ver Tabla 3). Un problema común a ambas actividades es que la mayoría de las propuestas no establecen métodos específicos para trabajar con las *propiedades sociales* de SMAs. Se limitan a establecer un vocabulario básico y realizar el tratamiento con los mismos mecanismos que para otras propiedades. Aunque así se facilita la integración, hay que recordar que las propiedades sociales hacen referencia a aspectos en los que los desarrolladores, al menos en cuanto al entorno, no son expertos. Una mayor especificidad de los mecanismos podría facilitar su trabajo con estas propiedades. KAOS a través de su metamodelo y librerías sí suministra mecanismos específicos de captura de requisitos y restricciones que ha de cumplir la información. Sin embargo, todo este conocimiento se centra una vez más en aspectos relativamente aislados del metamodelo.

Procesos			
Metodologías	**Guía**	**Requisitos**	**Comprobación**
AAII/BDI	Proceso software	Proceso software	Formalismo
KAOS	Proceso software + recorridos del metamodelo + librerías	Proceso software + recorridos del metamodelo + librerías	Proceso software + formalismo
ALBERT	Proceso software	Proceso software	Formalismo
i*	Proceso software	Proceso software	Programados en herramienta
DESIRE	Proceso software	Proceso software	Formalismo
Tropos	Proceso software	Proceso software	Formalismo
ADELFE	Proceso software	Proceso software	Manual
INGENIAS	Proceso software	Proceso software	Programados en herramienta
Etnografía	Proceso (descripción de métodos)	Métodos propios (manual)	Métodos propios (manual)
Diseño Cooperativo	Proceso (descripción de métodos) + contradicciones	Métodos propios (manual)	Métodos propios (manual) + contradicciones (manual)

Tabla 3. Resumen del tratamiento de las *propiedades sociales* en la ISOA y en las técnicas adaptadas a la Ingeniería del Software de las ciencias sociales. Procesos.

A propósito de los procesos disponibles para el trabajo con las *propiedades sociales* emerge otro aspecto importante de las metodologías tratadas: cómo guían los procesos (ver Tabla 3). Se trata de ver la forma en la que las metodologías dan pautas para decidir dónde y cómo modificar los modelos o qué nueva información se necesita. Por lo general, las metodologías revisadas se limitan a describir las tareas del proceso software en lenguaje natural o con diagramas de actividades. Una excepción a este tratamiento es KAOS. KAOS guía la adquisición del conocimiento mediante recorridos de su metamodelo e incluye librerías con elementos de interés dependientes de los dominios. No obstante, y sin importar lo ricas que sean estas guías, hay asuntos que quedan fuera de la descripción, tales como la forma de traducir la información de los clientes al lenguaje de modelado o consejos para la elección de las arquitecturas de agentes y SMAs.

Por último, la Tabla 4 recoge varios aspectos transversales al tratamiento de las *propiedades sociales* en las metodologías de la ISOA:

- *Integración*. Disponibilidad de mecanismos para trabajar combinadamente con diferentes tipos de *propiedades sociales* en distintos niveles organizativos.
- *Herramientas*. Existencia de herramientas automatizadas de soporte y funcionalidad de éstas.
- *Conocimiento experto*. Inclusión de conocimiento sobre *propiedades sociales* ajeno al proyecto concreto o a la experiencia del equipo involucrado en el desarrollo. En este apartado se ha excluido el conocimiento que forma parte de los lenguajes de modelado de la metodología o de la arquitectura propuesta,

puesto que todas las metodologías revisadas lo incluyen. El conocimiento experto de este apartado se refiere a librerías de conocimiento experto que los desarrolladores pueden o no aplicar en sus proyectos con una metodología.

Miscelánea			
Metodologías	**Integración**	**Herramientas**	**Conocimiento experto**
AAII/BDI	Formalismo	No	No
KAOS	Metamodelo + Formalismo	GRAIL/KAOS Modelado + informes	Librerías de dominio y metamodelo
ALBERT	Formalismo	Modelado + animación	No
i*	No	OME Modelado + módulos	No
DESIRE	Formalismo	No	No
Tropos	Formalismo	Ver i*	No
ADELFE	No	No	NCSs
INGENIAS	Metamodelo	IDK Modelado + módulos	No
Etnografía	Conceptual	No	Investigación + casos
Diseño Cooperativo	Conceptual	Prototipado rápido	Investigación + casos + TA

Tabla 4. Resumen del tratamiento de las *propiedades sociales* en la ISOA y en las técnicas adaptadas a la Ingeniería del Software de las ciencias sociales. Miscelánea.

Estrechamente relacionado con los lenguajes y los procesos está la posibilidad de un tratamiento integrado de las *propiedades sociales* (ver Tabla 4). Este tratamiento consiste en la disponibilidad de procesos como guías, captura de requisitos, comprobación de propiedades o modificaciones de los modelos del SMA, que consideren simultáneamente diferentes tipos de propiedades o varios niveles organizativos. De la revisión podemos concluir que esta integración se confía generalmente a la posibilidad de traducir todas las especificaciones a un mismo lenguaje (e.g. el formalismo en AAII/BDI o el lenguaje gráfico en INGENIAS). Ahora bien, una vez traducidas las especificaciones, la utilidad del tratamiento unificado depende de la habilidad de los desarrolladores para obtener y describir propiedades de interés. Esta tarea no es en absoluto trivial porque se trata de establecer patrones sociales complejos que no forman parte del conocimiento habitual de la Ingeniería del Software.

El último tema a considerar es el conocimiento experto (ver Tabla 4). Aquí consideramos repositorios de propiedades sociales que puedan ayudar a los desarrolladores en la captura y tratamiento de *propiedades sociales*. Sólo en KAOS y ADELFE encontramos estas librerías aunque centradas en aspectos muy precisos. Esta falta de conocimiento experto limita el aprovechamiento de estas propiedades, ya que queda limitado a las características dictadas por la experiencia de los desarrolladores o las que surjan en el proyecto.

Como ya indicamos, este trabajo se ha valido de las disciplinas sociales para solventar estos problemas en el tratamiento de las *propiedades sociales* en la ISOA.

Como muestras de aplicaciones similares a las de esta tesis hemos estudiado la Etnografía y el Diseño Cooperativo. La conclusión es que la base conceptual y metodológica que las ciencias sociales ofrecen para trabajar integradamente con las *propiedades sociales* en sus sistemas humanos, puede ser usada en procesos de desarrollo software, en nuestro caso de la ISOA. Además, las investigaciones y casos de estudio de las ciencias sociales constituyen amplios repositorios de conocimiento experto sobre propiedades sociales que podrían ser adaptados a los SMAs. El problema para usar las ciencias sociales radica en que su trabajo se realiza usando sobre todo el lenguaje natural, más expresivo pero también de más difícil interpretación que los lenguajes de diseño, y la aplicación de los procesos depende mucho de la capacidad de los investigadores.

En base a estas observaciones, la adaptación de técnicas de las ciencias sociales a la ISOA que aborda este trabajo cubre los siguientes aspectos:

- *Vocabulario para las propiedades sociales con mecanismos de ampliación.* Adaptado de un marco social que cubra los aspectos sociológicos y psicológicos del individuo y el grupo. Ha de basarse en conceptos suficientemente comunes como para servir de nexo entre clientes y desarrolladores.
- *Representación del vocabulario anterior y métodos con ella para usarlos en procesos software de SMAs.* Las propiedades han de poder ser utilizadas sobre distintas metodologías de la ISOA. Se ha de soportar la captura, comprobación de propiedades y modificación de las especificaciones.
- *Guía en el proceso*. La propuesta ha de indicar a los desarrolladores cuándo usar una *propiedad social*, cómo usarla y los resultados esperados.
- *Conocimiento experto*. Librerías de *propiedades sociales* de SMAs disponibles para su uso en proyectos concretos.

Satisfaciendo estos objetivos se cuenta con obtener una serie de ventajas sobre las propuestas existentes en la ISOA. Estas ventajas son:

- *Representación completa de las propiedades sociales.* El vocabulario básico de la teoría social soportará los aspectos esenciales de las *propiedades sociales*. Los mecanismos de ampliación permitirán su adaptación a nuevo conocimiento.
- *Potenciar la colaboración entre clientes y desarrolladores*. El vocabulario se basará en abstracciones sociales de uso común para las organizaciones humanas. Éstas pueden ser entendidas con mayor facilidad que las basadas en paradigmas de desarrollo de software.
- *Mecanismos de tratamiento portables*. Se opta por sacrificar la potencia de análisis de los procesos en aras de la facilidad de aprendizaje, comprensión y uso y una aplicación semi-automatizada.
- *Expresión de propiedades complejas reales y reducción de la dependencia de la habilidad de los desarrolladores*. Esto se logrará mediante la guía en el proceso junto con las librerías de propiedades. Otro beneficio esperado es la posibilidad de razonar sobre las decisiones de diseño y documentarlas mediante estos procesos.

Como se indicó en el prólogo, nuestro trabajo se he centrado en la TA [Vygotsky 1978]. En el próximo capítulo veremos que este marco presenta ciertas características que le hacen adecuado para los objetivos planteados aquí. En base a la TA se

suministrará un vocabulario para *propiedades sociales* y su representación en UML, métodos de captura de requisitos sociales, detección de propiedades y modificación de especificaciones y librerías con conocimiento experto.

Capítulo 3. Formulación de TA para SMAs

Este capítulo comienza justificado la elección de la Teoría de Actividad entre las propuestas de las disciplinas sociales para abordar los problemas en el tratamiento de las propiedades sociales en la ISOA vistas en el capítulo anterior. Se presentarán las características específicas de este marco de estudio que le hacen aconsejable para trabajar con la ISOA. Después se hace una breve reseña histórica sobre la Teoría de Actividad en sus campos de origen, la Psicología y la Sociología. El resto del capítulo introduce la infraestructura básica para aplicar la Teoría de la Actividad en la ISOA y finaliza con algunas consideraciones generales sobre ella.

3.1. Introducción

Las técnicas propuestas en esta tesis para facilitar el desarrollo de SMAs emergen de la Teoría de Actividad (TA) [Vygotsky 1978]. Se trata de un marco conceptual y metodológico que se usa para el análisis de sociedades e individuos en los campos de la Sociología y de la Psicología. La TA es una corriente muy extendida que ha incluido aplicaciones en campos tan diversos como la investigación Psicológica básica [Leontiev 1978], aprendizaje [Zinchenko 1985, Engeström 1987], motivación [El'konin 1977], influencia cultural en las labores cognitivas [Luria 1976, Ilyenkov 1982], Neuropsicología [Luria 1979], CSCW [Bødker & Grønbæk 1996] o Ergonomía y HCI [Bednyi & Meister 1997, Kaptelinin *et al.* 1999]. Buena parte de su investigación ha estado relacionada directamente con el estudio de las *propiedades sociales* en las organizaciones humanas como se aprecia en esta enumeración.

La TA define conceptos para el análisis y relaciones entre ellos. Además establece niveles de contradicciones en las sociedades para estructurar su estudio. Esta infraestructura se utiliza sobre la situación de la sociedad a analizar para identificar los elementos a considerar. Después, los niveles de contradicción proporcionan las pautas para determinar cuáles son las inconsistencias en el sistema. Estas inconsistencias se plasman por ejemplo en la insatisfacción de los individuos, estructuras burocráticas que no sirven al ciudadano, carencias de servicios o desordenes sociales. El seguimiento en el tiempo de la situación de la sociedad y de los elementos identificados en ella, permite establecer la forma en la que se ha solventado la contradicción generando una nueva configuración de la organización.

Existen varios motivos por los que se ha elegido la TA frente a otras propuestas de las ciencias sociales:

- *Cubre las expectativas sobre el marco de estudio de las propiedades sociales.* La TA contempla el tratamiento integrado de las *propiedades sociales*, suministra un vocabulario para representarlas, cuenta con guías y métodos para la captura y

comprobación de dichas propiedades y, por último, sus estudios sirven como fuente de inspiración para el tratamiento de estas propiedades.

- *Existen precedentes en el uso de la TA para el desarrollo de software*. Estos precedentes permiten pensar que se trata de un marco adaptable a la ISOA. En el capítulo anterior ya se vio el ejemplo del Diseño Cooperativo (ver sección 2.4.2). Otras propuestas en la misma línea se encuentran en [Gould *et al.* 2000, McGrath & Uden 2000]. Algunos investigadores han tratado de aproximar el uso de la TA a procesos más estructurados, propios de la Ingeniería del Software, [Barros & Verdejo 2000, Mwanza 2000], aunque siguen presentando una marcada carencia de guía práctica. También la TA ha actuado como inspiración de conceptos en la investigación sobre agentes (e.g coordinación de SMAs [Ricci *et al.* 2002]). No obstante, esta tesis se diferencia de estas propuestas en su mayor cercanía a las prácticas de la Ingeniería del Software y su uso de la TA para construir un marco complementario de la ISOA para el tratamiento de *propiedades sociales*.
- *Cuenta con características que favorecen su aplicación sobre un proceso software*. Entre estas características se cuentan el acuerdo en su comunidad sobre un vocabulario básico y reducido para el modelado o concebir la evolución de los sistemas como un proceso guiado por sus *contradicciones*.

Pese a estas ventajas, la TA también presenta problemas en su adaptación a la ISOA, comunes por otra parte a las propuestas de las disciplinas sociales para la Ingeniería del Software que se vieron en el capítulo anterior. Estos problemas esenciales son:

- *Uso del lenguaje natural para realizar especificaciones y describir procesos*. Casi toda la investigación del área se ha realizado en lenguaje natural. Esta forma de analizar se encuentra bastante alejada de las propias de la Ingeniería del Software fundamentadas en lenguajes de diseño.
- *Diferente marco conceptual*. Pese a que tanto la TA como las metodologías para SMAs tratan con sistemas sociales e intencionales, manejan conceptos diferentes.
- *Diferentes métodos de trabajo*. Los métodos propios de la TA confían mucho en el criterio de los expertos acerca de las tareas que se deben realizar y cómo realizarlas. En comparación con los procesos habituales en la Ingeniería del Software (representados por ejemplo con diagramas de actividades y que incluyen indicaciones de los productos que se deben obtener o de las comprobaciones que se deben realizar sobre ellos), se puede considerar que los procesos de la TA carecen de guías regladas. Adaptar estos procesos a la ISOA es otro de los problemas a resolver para usar la TA en el tratamiento de *propiedades sociales*.

Como se verá más adelante en este capítulo, la base de la solución a estos problemas en la presente tesis ha consistido en expresar los conceptos esenciales de la TA en un lenguaje bien conocido en la comunidad de la Ingeniería del Software, UML [OMG 2003]. Sobre esta representación se ha definido un proceso de traducción entre el vocabulario de la TA y los de las metodologías de agentes. Con la representación UML y el proceso de traducción, se pueden definir procesos genéricos inspirados por la TA y utilizables sobre diferentes metodologías de la ISOA.

Los apartados restantes en este capítulo se organizan como sigue. En primer lugar se hace una introducción a la TA en sus campos de aplicación original. A continuación, la sección 3.3 resume el esquema general planteado en esta tesis para la adaptación y uso de la TA sobre las *propiedades sociales* en la ISOA. Las posteriores secciones desarrollan la parte de este esquema que corresponde a la infraestructura común a las herramientas de la TA para la ISOA. Estas secciones describen la representación con UML de la TA (ver sección 3.4), dan las guías para establecer las traducciones entre sus conceptos y los de las metodologías de la ISOA (ver sección 3.5), introducen las estructuras para representar las propiedades sociales (ver sección 3.6) y el método de comprobación de esta estructuras sobre especificaciones de SMAs (ver sección 3.7). El capítulo termina con algunas consideraciones acerca de las decisiones tomadas para trabajar con la TA en las metodologías de la ISOA.

3.2. Contexto de la TA

La Teoría de Actividad (TA) [Vygotsky 1978, Leontiev 1978] es un marco para el estudio de las diferentes formas de las prácticas humanas como procesos evolutivos con el nivel individual y el social entrelazados. Desde el punto de vista de la TA, el comportamiento de las personas no puede ser comprendido independientemente del marco socio-cultural en el que se encuentran inmersas. Pero la gente no tiene sólo una relación pasiva con su entorno, interactúan activamente con él, cambiando sus objetos y creando nuevas herramientas. Esta interacción compleja de los individuos con su entorno ha sido llamada *actividad* y es contemplada como la unidad fundamental de análisis en la TA. El contexto de la *actividad* recibe el nombre de *sistema de actividad*.

El análisis de una *actividad* se basa en considerar ésta como un objeto de estudio sistémico y en investigar las relaciones que existen entre sus elementos. En ningún caso se trata meramente de descomponer la *actividad* en los elementos que la constituyen, ya que entonces éstos perderían su significado.

La TA emplea dos dimensiones ortogonales para analizar la *actividad*. Por un lado, la *actividad* se examina desde el punto de vista de sus participantes, las interacciones entre ellos y las relaciones de mediación entre estos componentes. Por otro lado, se contempla la descomposición jerárquica de la actividad en otros procesos más simples, distinguidos por su motivación y duración.

La dimensión temporal-evolutiva del análisis está marcada por las contradicciones. Las contradicciones internas del *sistema de actividad* han de ser analizadas como fuente de conflictos, innovación, cambio y desarrollo en dicho sistema. Los conceptos relativos a la descripción de la *actividad* constituyen la base para modelar y analizar estas contradicciones. Para abordar este estudio la TA identifica niveles de contradicción que se definen según las relaciones de mediación donde aparecen en la *actividad*.

En esta sección se empleará un ejemplo clásico, desarrollado en [Leontiev 1981, Engeström 1987] acerca del trabajo de un médico en una clínica de atención primaria, para ilustrar estos conceptos de la TA.

3.2.1. Estructura del *sistema de actividad*

En el contexto de la TA, la *actividad* [Engeström 1987] refleja un proceso (representado en la Fig. 4). Brevemente se puede decir que la *actividad* es un proceso ejecutado por un *sujeto* con la ayuda de *herramientas*, para transformar un *objeto* en el *producto* requerido para satisfacer el *objetivo* del *sujeto*. El análisis de estos elementos constituye una primera dimensión de estudio de una *actividad*.

El *sujeto* es el elemento activo que lleva a cabo la *actividad*. Puede representar tanto a un individuo como a un grupo de individuos. Su punto de vista es el que se adopta para el estudio de la *actividad*. El *sujeto* tiene una necesidad definida, representada por el *objetivo*. No todos los participantes involucrados en una *actividad* comprenden o reconocen necesariamente el motivo de la misma. En este caso, estos participantes no pueden ser considerados *sujetos* activos de la *actividad*.

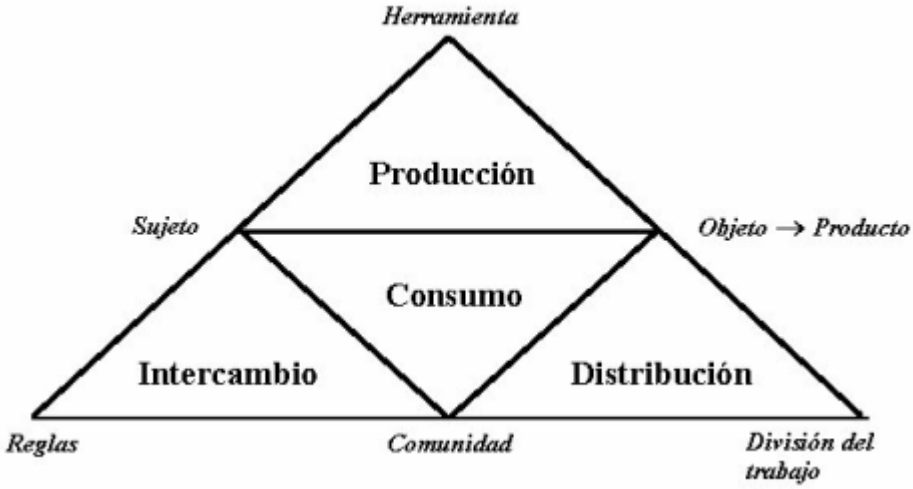


Fig. 4. Conceptos de la Teoría de Actividad y relaciones de mediación [Engeström 1987].

El *objetivo* y el *objeto* determinan el significado de la *actividad* y la distinguen de otras *actividades*. La *actividad* surge [Leontiev 1978] cuando una necesidad del *sujeto* es identificada con un modo de satisfacerla en el mundo real, y entonces este modo de satisfacción se convierte en el *objeto* de la nueva *actividad*. Por tanto, el *objetivo* es satisfecho por un *producto* que se obtiene transformando un *objeto*. El *objeto* puede ser tanto mental como material, en tanto pueda ser compartido para su manipulación y transformación por los participantes en la *actividad*.

La transformación mutua que se establece entre el *sujeto* y el entorno en el curso de una *actividad* nunca es directa, sino siempre mediante el uso de *herramientas*: las *herramientas* siempre median los procesos del *sujeto* sobre los *objetos* [Vygotsky 1978]. Una *herramienta* puede ser cualquier recurso empleado en el proceso, tanto material (por ejemplo una computadora) como intelectual (tal como un plan). La *herramienta* es al mismo tiempo capacitadora y limitadora en la *actividad* [Leontiev 1974]: proporciona al *sujeto* en el proceso de transformación la experiencia recogida

históricamente por su comunidad y cristalizada en la *herramienta*; pero al mismo tiempo también limita la interacción a ser desde la perspectiva de esa *herramienta* particular, esto es, el instrumento hace que otras características potenciales del *objeto* permanezcan ocultas al *sujeto*.

Los anteriores conceptos se centran en la descripción de la *actividad* realizada por un único *sujeto*. Ahora bien, se debe considerar además la conexión de la *actividad* individual con las *actividades* del resto de los *sujetos* en la sociedad, es decir, analizar su nivel social.

El nivel social de la *actividad* se caracteriza por tres conceptos: la *comunidad*, las *reglas* y la *división del trabajo*. La *comunidad* representa a aquellos *sujetos* que comparten un mismo *objeto* [Kuutti 1996]. Las *reglas* median entre el *sujeto* y la *comunidad*, y la *división del trabajo* entre el *objeto* y la *comunidad*. Una *actividad* puede involucrar a varios *sujetos* y cada *sujeto* puede jugar uno o varios roles y tener múltiples motivos [Leontiev 1989]. Las *reglas* especifican cómo los *sujetos* encajan en las *comunidades*. Las *reglas* cubren tanto las normas implícitas como las explícitas, las convenciones y las relaciones sociales dentro de la *comunidad*. La *división del trabajo* se refiere a la organización de la *comunidad* en relación con el proceso de transformación del *objeto* en *producto*.

Como una generalización de todos los conceptos anteriores se suele hablar de *artefactos*. Un *artefacto* puede ser, por tanto, cualquier elemento que aparezca en el modelado de *actividades*.

El modelo también sugiere la posibilidad de analizar múltiples relaciones dentro de la estructura de la *actividad*, los llamados *momentos* [Marx 1973]. La *Producción* crea el *objeto* que satisface las necesidades asociadas a la *actividad*; la *Distribución* divide dicho *objeto* de acuerdo con las leyes sociales; el *Intercambio* realiza distribuciones adicionales de acuerdo con las necesidades individuales; en el *Consumo*, el *producto* abandona el ámbito social de los demás triángulos y se convierte en el *objeto* directo de las necesidades individuales que satisface mediante su consumo. Por tanto, la *Producción* aparece como el punto de partida, el *Consumo* como la conclusión y la *Distribución* y el *Intercambio* como los eslabones intermedios. No obstante, la tarea esencial siempre consiste en estudiar la *actividad* como un todo sistémico.

Estos conceptos para el modelado de un *sistema de actividad* pueden ser concretados para el ejemplo del médico que trabaja en la clínica de atención primaria [Leontiev 1981]. El *objeto* de su trabajo son los pacientes con sus problemas de salud. Entre los *productos* de su *actividad* se incluyen las recuperaciones de dichos pacientes y las mejoras en su salud, pero también los resultados no deseados, tales como la no-satisfacción de los pacientes o las discontinuidades en los tratamientos. Los instrumentos en este caso incluyen *herramientas* como los rayos-X, laboratorios, registros médicos, diagnósticos o conceptos y métodos relacionados con los tratamientos. La *comunidad* consiste en el personal de la clínica. La *división del trabajo* determina las labores y el poder de decisión de los médicos, enfermeros y demás categorías del personal. Por último, las *reglas* establecen cómo se regula el uso de los recursos, las medidas de calidad del tratamiento o los criterios para repartir las bonificaciones en los sueldos.

3.2.2. Descomposición jerárquica de la *actividad*

La segunda dimensión de análisis en la Teoría de Actividad es la descomposición de una *actividad* en otros procesos más simples. Esta estructura jerárquica (ver Fig. 5) se fundamenta en el estudio de las dinámicas internas, transformaciones y desarrollo de la *actividad* [Leontiev 1978].

Los niveles de esta estructura jerárquica difieren en su granularidad y motivación. En cuanto a la granularidad, las *actividades* presentan formulaciones a largo plazo. Los *objetos* de las actividades son transformados en *productos* a través de procesos que suelen constar de varias fases. Para describir estas fases hay una necesidad de contar con procesos de menor duración que las *actividades*. Las *actividades* consisten entonces en *acciones* o en cadenas de *acciones* que a su vez consisten en *operaciones*. Por otra parte, estos tres tipos de procesos (i.e. *actividades*, *acciones* y *operaciones*) también difieren en cuanto a su propósito. Paralelamente a la jerarquía de procesos existe una de *objetivos* que incluye *motivos*, *metas* y *condiciones*. *Motivos*, *metas* y *condiciones* corresponden respectivamente a *actividades*, *acciones* y *operaciones*. El análisis de la TA comienza aislando en el flujo total del proceso de transformación las *actividades*, es decir, aquellos procesos cuyos *objetivos* están asociados a necesidades del *sujeto*, los llamados *motivos*. Después se extraen los procesos que están subordinados a *metas* conscientes necesarias para satisfacer los *motivos*. Estos procesos son las *acciones*. Finalmente, las *acciones* se han de llevar a cabo en unas circunstancias específicas. Las *operaciones* son los métodos para realizar las *acciones* que dependen de las *condiciones* concretas. Frecuentemente, el *sujeto* no percibe conscientemente las *condiciones* de las *operaciones*, en contraste con las *metas* de las *acciones* que sí son conscientes. Las *herramientas* en la *actividad* son *operaciones* cristalizadas.

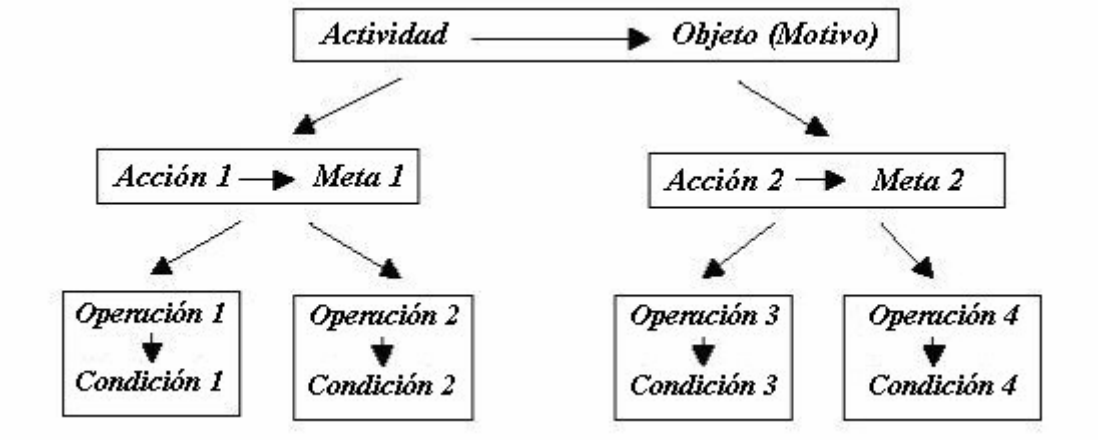


Fig. 5. Estructura jerárquica de la *actividad* y sus fuerzas directoras [Engeström 1987].

En este punto es necesario realizar varias puntualizaciones acerca de las relaciones entre la *actividad*, las *acciones* y las *operaciones*.

Aunque la *actividad* sólo puede existir bajo la forma de cadenas de *acciones* y las *acciones* sólo pueden ser ejecutadas como *operaciones*, no deben reducirse unas a un mero conjunto de las otras, ya que ello violaría el carácter sistémico del análisis de la *actividad*.

Otro aspecto a tener en cuenta es que *actividades* y *acciones* no están relacionadas de forma única. Una misma *acción* puede formar parte de varias *actividades*, incluso simultáneamente. Por otra parte, el *motivo* de una *actividad* puede encontrar su expresión en diferentes *metas* y *acciones*.

También hay que señalar que la clasificación de un cierto proceso en la estructura jerárquica de la *actividad* no es fija [Leontiev 1978]. Así, una *actividad* puede perder el *motivo* que la provocó y convertirse entonces en una *acción* subordinada a una *meta*, realizando quizás como *acción* una relación diferente con el mundo que la rodea. Del mismo modo, la *meta* de una *acción* puede convertirse en una fuerza estimulante independiente, satisfaciendo una necesidad del *sujeto* por sí misma, y entonces transformar la *acción* en una *actividad* separada. Finalmente, una *acción* puede dejar de ser el medio para alcanzar *metas* objetivas y convertirse en una *operación* capaz de realizar varias *acciones*. Del mismo modo, una *operación* cuyas *condiciones* se hacen objetivas para el *sujeto* puede transformarse en una *acción*.

Por último, apuntar que en la estructura jerárquica de la *actividad* es posible la descomposición horizontal, es decir, dentro del mismo nivel. Durante la realización de un *objetivo* en cualquiera de los tres planos (i.e. *actividades*, *acciones* y *operaciones*), puede tener lugar una separación de éste en *objetivos* intermedios. Como resultado de esta separación, el proceso completo se divide en series de procesos secuenciales independientes al mismo nivel. De este modo, un proceso puede convertirse en una fracción de otro más grande o incorporar en sí mismo unidades que previamente eran independientes.

Una vez más, concretamos estos conceptos en el ejemplo del trabajo del médico [Leontiev 1981] basándonos en la presentación en [Leontiev 1978]. Cuando el médico se está formando en su profesión, cada proceso, e.g. auscultar el corazón de un paciente, se constituye en una *acción* con su propia *meta* consciente, i.e. realizar el proceso concreto. Posteriormente, y al adquirir una mayor experiencia, esa *acción* queda incluida en otra. Ahora la auscultación se convierte en uno de los métodos posibles para alcanzar un *objetivo*, i.e. conocer el estado cardiaco del paciente, y pasa a ser una *operación* capaz de satisfacer la *meta* de otra *acción*, e.g. evaluar las condiciones cardiacas del paciente. Es decir, la *meta* de la auscultación no está aislada en la mente del sujeto, sino formando parte de otra *meta*. Para la consciencia del doctor, la auscultación en circunstancias habituales no existe independientemente. Al final, el doctor realiza una *actividad*, i.e. evaluar la salud de su paciente, dentro de la cual incluye varias *acciones*, e.g. reconocer el estado cardiaco del paciente, que lleva a cabo con *operaciones*, e.g. mirar un electrocardiograma o auscultar, de forma prácticamente automática según los medios de que dispone. Generalmente, tales *operaciones* terminan convirtiéndose en las funciones de máquinas.

3.2.3. Contradicciones

La TA analiza los motivos del comportamiento de las sociedades humanas y su evolución a través de las contradicciones presentes en sus *actividades* [Leontiev 1978]. Las contradicciones de la TA representan desviaciones de la norma común en la sociedad y tensiones entre los componentes de las *actividades*, que llevan la semilla de nuevas formas de *actividad*. Estas contradicciones se analizan a través de cuatro niveles en el *sistema de actividad* (ver Fig. 6).

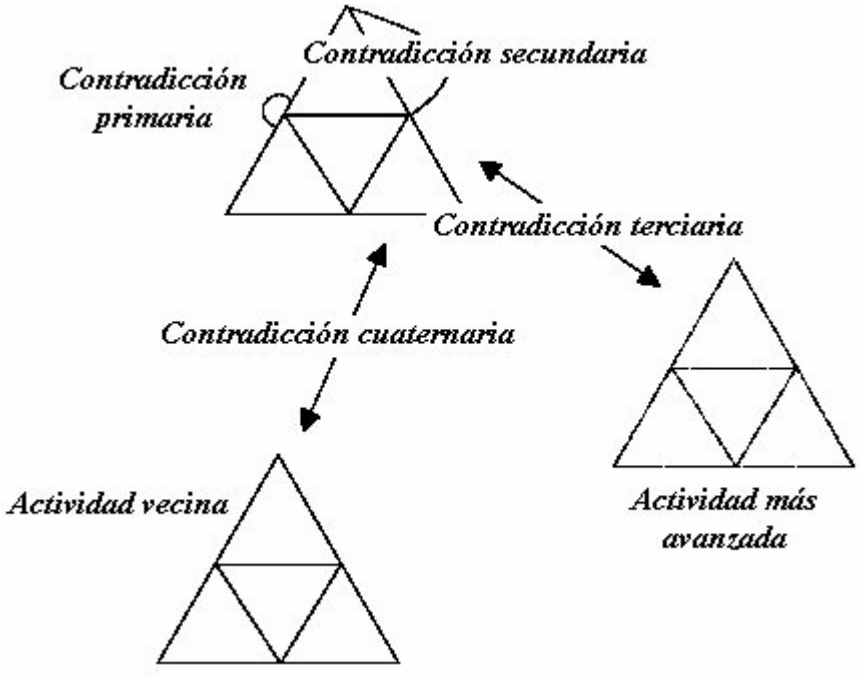


Fig. 6. Los cuatro niveles de contradicción del *sistema de actividad*.

La contradicción básica de la *actividad* humana es su doble condición como producción independiente y al mismo tiempo subordinada a la producción de la sociedad. En cada *actividad* concreta esta contradicción se plasma en el conflicto entre las tareas individuales y la actividad global. De acuerdo con la TA, este conflicto emerge de la *división del trabajo* y adopta formas diferentes en cada momento histórico, según el modelo dominante de organización socio-económica. En las sociedades pre-capitalistas, esta contradicción solía tomar la forma de subordinación por la fuerza del individuo al poder que regía la sociedad. En las sociedades capitalistas esta contradicción se plasma en la diferencia entre el valor de uso y el de intercambio, esto es como mercancía, de los objetos. Estas contradicciones primarias se producen dentro de cada una de las esquinas del triángulo de la Fig. 4. Las contradicciones secundarias son las que aparecen en las relaciones de mediación entre las esquinas del triángulo. Las contradicciones terciarias surgen cuando se

introduce una forma más evolucionada de la *actividad*; se trata de la resistencia al cambio en la forma dominante de la *actividad* frente a una posible modificación. Finalmente, las contradicciones cuaternarias son las que surgen entre la *actividad central* que se está estudiando y sus *actividades vecinas* en el contexto de una red de *actividades* interconectadas. La interconexión entre *actividades* surge porque algunas crean o modifican los elementos del *sistema de actividad* de otras.

Las *actividades vecinas* de una *actividad* incluyen:

- *Actividades objeto*. Aquellas en las que se embeben los *objetos* y *productos* de la *actividad* en estudio.
- *Actividades productoras de herramientas*. Producen los instrumentos empleados en la *actividad central*.
- *Actividades productoras de sujetos*. Son aquellas que afectan al pensamiento, comportamiento y habilidades del *sujeto*. Sería el caso de la educación o de sus experiencias previas.
- *Actividades productoras de reglas*. Son actividades que determinan las normativas que rigen en la *comunidad*. Se trata, por ejemplo, de la legislación o la administración.

Un ejemplo de actividad que contiene las contradicciones señaladas es presentado en [Engeström 1987]. Se trata del trabajo de un médico de atención primaria. Las contradicciones primarias aparecen en cada esquina de su triángulo de actividad, como el doble valor de uso y de intercambio. Éste es el caso de las medicinas que usa. Por un lado se trata de instrumentos para mejorar la salud del paciente pero por otro son mercancías manufacturadas para el mercado y vendidas por un beneficio. En cuanto a las contradicciones secundarias podrían surgir entre las tradicionales técnicas de diagnóstico relativas a la clasificación de los síntomas y la naturaleza cambiante del objeto de estudio, los pacientes. Las contradicciones terciarias emergen cuando el doctor se ve obligado a alterar su manera de trabajo por nuevas directivas que tratan de adaptar su trabajo a concepciones distintas. Incluso aunque los nuevos procedimientos lleguen a implementarse es posible que se enfrenten durante un tiempo a cierta resistencia a modificar los hábitos adquiridos. Para terminar, las contradicciones cuaternarias pueden aparecer con la *actividad objeto*, en este caso el "modo de vida" del paciente. Dentro de la *actividad* de la atención primaria de la salud, el doctor puede recomendar al paciente cambios en su alimentación que sean recibidos negativamente por el paciente.

A partir del análisis de las contradicciones en la *actividad* surge la dimensión temporal de la TA. La *actividad* evoluciona en un ciclo que comienza con las necesidades de los individuos en su forma actual. La incorrecta satisfacción de estas necesidades se plasma en *contradicciones* entre los elementos que componen dicha *actividad*. La emergencia de estas inconsistencias lleva a algunos individuos a iniciar la experimentación de nuevas formas de la *actividad*. Las nuevas formas pasan de los grupos reducidos de *sujetos* a la sociedad. En ella, estas nuevas formas de *actividad* experimentan un proceso de transformación y generalización. La experimentación de la nueva *actividad* en la *comunidad* puede llevar a su consolidación en ella o bien a que sea descartada. Finalmente, si la *actividad* se consolida, comienza un nuevo ciclo evolutivo con la reflexión sobre la ahora dominante forma de *actividad* y sus contradicciones.

3.3. Esquema general de uso de la TA en la ISOA

Extrapolando la clase de análisis de la TA descrito en la sección anterior, el proceso de desarrollo de SMAs se contempla como un ciclo continuo que incluye las siguientes fases:

1. *Recolección de nueva información*. Corresponde a la identificación de la situación de la sociedad según la TA. En un proceso software consiste en el estudio de las especificaciones y puede incluir, en el caso de que no se disponga de suficiente información, la obtención de nuevo conocimiento sobre el sistema y su entorno, es decir, captura de requisitos.
2. *Detección de contradicciones en las especificaciones*. Es la detección de *contradicciones* en las *actividades* de la sociedad. Para un proceso software correspondería a comprobar que las especificaciones cumplen ciertas propiedades, i.e. las *contradicciones*. Se puede identificar por tanto con las técnicas de verificación y validación para la ISOA.
3. *Resolución de contradicciones*. Según la TA sería la experimentación e introducción de nuevas formas de *actividad* para solventar los conflictos identificados. Desde el punto de vista de la ISOA corresponde a modificar las especificaciones para solucionar la contradicción.

El conocimiento de la TA empleado en los tres pasos previos es lo que se denomina en este trabajo *propiedades sociales* de la TA. Como se recordará de la sección 1.2, siguiendo la línea de las disciplinas sociales [Constantine 1995] definimos las *propiedades sociales* de los SMAs como la información relativa a conceptos organizativos, cognitivos, de desarrollo y evolución, o de motivación en estos sistemas. Las *propiedades sociales* de la TA para SMAs describen configuraciones de estos sistemas relacionadas con sus componentes intencionales y sociales. Estas configuraciones inspiradas por la TA incluyen información que las especificaciones de un SMA han de contemplar y propiedades/contextos que pueden aparecer en un SMA así como las posibles evoluciones que puede seguir a partir de ellos.

Los pasos anteriores esquematizan una posible forma de aplicar la TA sobre un proceso software de la ISOA, que es la que se adopta en este trabajo. Realizar este proceso como una técnica complementaria del desarrollo habitual, cuyo uso no suponga alteraciones importantes en las metodologías existentes, implica solventar varios problemas esbozados en los capítulos previos y en la propia introducción de éste. Estos inconvenientes son: el uso del lenguaje natural para el trabajo en la TA; el diferente marco conceptual de la TA y el paradigma de agentes; las diferencias en cuanto a objetivos y forma de aplicación de los procesos de la TA y la ISOA. El esquema general del proceso de solución para estos problemas queda representado en la Fig. 7. Todas las actividades de la Fig. 7 son abordadas en este capítulo con la excepción de las dos últimas, la creación de procesos para la captura de requisitos (ver 0) y para el tratamiento de contradicciones (ver Capítulo 5). A continuación se explican con más detalle estos pasos.

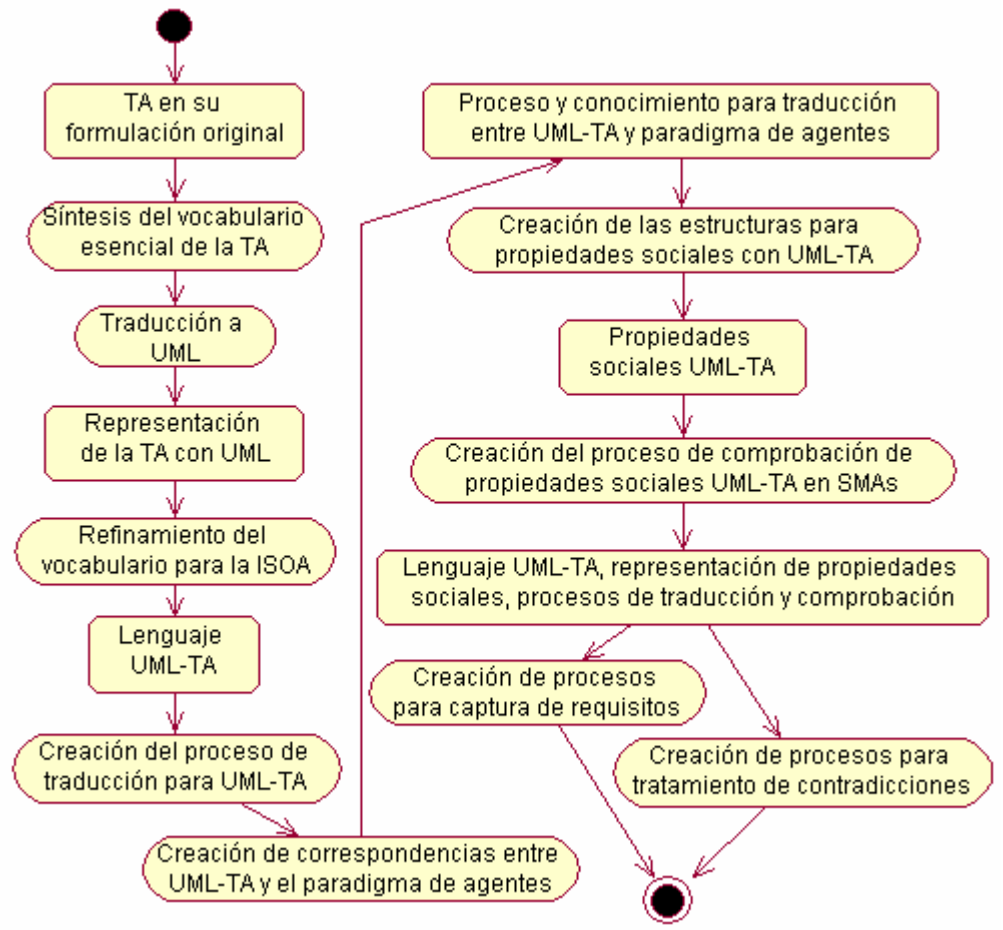


Fig. 7. Adaptación de la TA a la ISOA.

La clase de análisis encontrados en los trabajos de TA es la propia de las disciplinas sociales, como Filosofía, Psicología, Sociología o Pedagogía. Son estudios que trabajan con el lenguaje natural, donde el vocabulario es fundamental y el contenido semántico muy denso. Los términos poseen significados muy precisos y aluden a grupos de otros conceptos con relaciones concretas. Se trata por tanto de textos de difícil lectura e interpretación para los no expertos. Estos estudios incluyen también las descripciones que realizan las personas observadas. Aquí se trata del uso común del lenguaje natural o del realizado por expertos de otras áreas, lo que genera ambigüedad y malentendidos.

La ISOA, en contraposición con la TA, trabaja con lenguajes de diseño, menos ricos que el lenguaje natural pero más estructurados. Puesto que se pretende generar técnicas para la ISOA se ha optado por representar los conceptos de la TA en un lenguaje conocido en la comunidad de la Ingeniería del Software, UML [OMG 2003]. En el resto de este trabajo nos referiremos a este lenguaje como UML-TA.

Obviamente, la menor riqueza expresiva de UML frente al lenguaje natural hace que adaptar el conocimiento de la TA requiera interpretaciones y simplificaciones. Como ventajas, UML ya se usa en los procesos de ingeniería y establece las suficientes restricciones como para automatizar parte del tratamiento de información sobre sus especificaciones. A modo de ejemplo, son de interés en este trabajo la detección de *propiedades sociales* representadas con UML o la modificación de modelos.

Una vez que el conocimiento de la TA ha sido representado en un lenguaje más cercano al desarrollo de software, persiste el problema de aplicar las nociones de la TA sobre las del paradigma de agentes. Para aplicar la TA a procesos software de la ISOA, se ha optado por aislar la traducción entre conceptos de ambos paradigmas mediante correspondencias dependientes de la metodología de agentes empleada. De este modo se permite particularizar las traducciones según se necesite. Esta particularización es necesaria para tratar varios problemas de la traducción:

- *Carencia de una correspondencia biunívoca entre conceptos*. La TA y la ISOA trabajan sobre diferentes marcos conceptuales. Los conceptos de cada grupo no tienen una única posible traducción en el otro.
- *Dependencia de la metodología*. Aunque casi todas las metodologías de agentes comparten ciertos conceptos cuyo origen se encuentra en los trabajos de [Newell 1982] y [Shoham 1993], la interpretación específica de estos y la notación que los plasma cambian.
- *Adaptación al usuario*. La interpretación exacta de una notación depende de sus usuarios. Incluso en los lenguajes con una semántica formal, es el usuario final el que ha de construir los modelos que dotan a la notación de una interpretación.

El uso de correspondencias entre conceptos, ajustables en función del contexto concreto que se considera, permite reflejar el significado correcto en cada situación. Sobre estas correspondencias se apoyan las demás técnicas presentadas en este trabajo.

Acerca de la adaptación de los procesos propios de la TA a la ISOA, las soluciones adoptadas son resumidamente:

- *Comprobación de propiedades sociales*. El proceso de desarrollo basado en TA de esta tesis encontró útil contemplar el trabajo con las *propiedades sociales* en general y no sólo con las *contradicciones*. De esta forma la comprobación de propiedades se puede considerar validación cuando aborda los requisitos y verificación cuando se trata de otras propiedades. El proceso de detección se basa en la utilización de patrones estructurales expresados con el lenguaje gráfico para la TA. Con esta aproximación se da a los clientes y desarrolladores una herramienta sencilla para describir las propiedades a verificar o validar, generalmente de manera visual. Además, la localización de estos patrones en las especificaciones sirve para desarrollar las explicaciones necesarias en términos de la información ya disponible. Por supuesto, las comprobaciones construidas de esta manera no pueden ser tan sofisticadas como las que permitiría el uso de un formalismo. Sin embargo, sí resultan suficientes para comprobar muchas inconsistencias, ambigüedades o falta de detalles en las especificaciones.
- *Captura de requisitos*. Se han adaptado para SMAs las técnicas de captura de información sobre el contexto de las *actividades* en la TA. Estas técnicas se han reelaborado sobre las abstracciones de la TA plasmadas en lenguaje natural y

UML-TA. La elección de estas abstracciones y de la representación se fundamente en la facilidad para su traducción a las de agentes y en que se fundamentan en una base social común y conocida por clientes y desarrolladores. De esta forma se facilita la comunicación entre clientes y desarrolladores, que juega un papel primordial en la captura de requisitos.

- *Guía del proceso de desarrollo*. La guía en la evolución de las *propiedades sociales* se basa en el tratamiento de *contradicciones*. Se trata de un proceso cíclico donde en cada paso de detectan y corrigen *contradicciones* de la especificación. Como se señaló anteriormente, la TA identifica patrones en las sociedades, algunos de los cuales corresponden a *contradicciones*, y en base a ellos explica la evolución de las sociedades. Analizando los casos de estudio, la TA también describe cómo las sociedades han resuelto dichas *contradicciones*. Con esta información, en cada ciclo el proceso que se propone detecta *contradicciones* sociales en la especificación y genera su explicación. Las *contradicciones* determinan los puntos de posible evolución de la especificación. Las soluciones asociadas a las *contradicciones* sugieren posibles transformaciones de la información para hacer evolucionar la especificación.

3.4. Lenguaje de la TA en la ISOA

Como ya se ha señalado, uno de los problemas fundamentales de aplicar la TA al desarrollo de software es el lenguaje de trabajo. La Ingeniería del Software [Brooks 1975] se ha enfrentado al problema de la comunicación dentro del equipo de desarrollo y de éste con los clientes a través de lenguajes de modelado que eliminaran en la medida de lo posible la ambigüedad de la comunicación. Aunque parte de los modelos se describen en lenguaje natural, siempre se trata de traducir esta información a otro lenguaje que ofrezca menos posibilidades de interpretaciones erróneas. Otras áreas de conocimiento, tales como las ciencias sociales, no han desarrollado lenguajes en este sentido, por lo que sus técnicas usan el lenguaje natural. Para superar esta diferencia entre la ISOA y la TA, esta investigación ha sintetizado los conceptos esenciales de la TA en un lenguaje más próximo a la comunidad de la ISOA.

El lenguaje de modelado de software seleccionado para representar los conceptos de la TA ha sido UML [OMG 2003]. Los criterios en los que se ha basado esta elección han sido su amplia difusión en la comunidad de la Ingeniería del Software y el hecho de que proporciona mecanismos para definir nuevos elementos en su vocabulario, con restricciones específicas, dentro de un dominio concreto de aplicación. Estos mecanismos son los estereotipos. Se trata de subtipos de elementos base de un modelo pero con un significado y uso particulares. Los estereotipos pueden utilizar valores etiquetados para almacenar sus propiedades que no forman parte del elemento base. Además pueden tener asociadas restricciones de uso particulares tales como las relaciones en las que pueden participar. En UML los estereotipos se muestran como cadenas de texto rodeadas por los símbolos “<<” y “>>”. Los modeladores también pueden crear iconos propios para los estereotipos a fin de mejorar su identificación visual.

La especificación de los conceptos de la TA con UML partió de la descripción gráfica estándar de la TA (recogida en la Fig. 4). Este vocabulario se completó con la adición de nuevos conceptos para modelar aquellos aspectos presentes en los estudios de la TA pero que no están incluidos en la descripción de la Fig. 4. En paralelo con este estudio, se han analizado las correspondencias entre las nociones de la TA y las del paradigma de agentes. Gracias a este análisis se han clarificado los conceptos de la TA y se ha podido describirlos en términos próximos al paradigma de agentes. Además, este estudio ha llevado a identificar algunos conceptos que especializan a los de la TA y que facilitan la especificación de SMAs. El resultado de estos análisis ha generado el conjunto de conceptos de la TA que se ha especificado con UML. El resultado es un lenguaje, UML-TA, que se presenta en la sección 3.4.1.

El otro aspecto de interés al definir el lenguaje UML-TA tiene que ver con las equivalencias entre conceptos. Estas equivalencias se emplean al realizar traducciones o localizar propiedades en unas especificaciones para hacer sustituciones de conjuntos de elementos UML-TA por otros con el mismo significado. Así se reduce el número de patrones a considerar. Estas equivalencias se describen en la sección 3.4.2.

3.4.1. Vocabulario de UML-TA

Este apartado presenta el lenguaje UML-TA, la representación con UML de los conceptos de la TA para el modelado de SMAs. Para diferenciar en el texto los conceptos de las especificaciones de un SMA de los propios de la TA, el resto de esta exposición se referirá a los conceptos de la TA como *roles* que juega un concepto de acuerdo con la TA en un *sistema de actividad*.

A continuación se define la representación de los *roles* de la TA incluidos en el triángulo de Engeström (ver Fig. 4) con UML. Sobre el triángulo se han añadido los *roles* de *sistema de actividad*, *objetivo* y *artefacto*. El *sistema de actividad* incluye todo el contexto de una *actividad*. Consiguientemente abarca todos los elementos del triángulo de la TA asociados a una *actividad* dada, así como los posibles refinamientos de estos elementos en otros. El *objetivo* representa las necesidades que satisface el *producto* de la *actividad*. El *artefacto* puede representar a cualquiera de los conceptos de la TA. Además, a fin de poder facilitar la referencia a las relaciones entre los *roles* de la TA, éstas se han hecho explícitas en la representación y se les han proporcionado nombres significativos. Los estudios de TA suelen describir estas relaciones y su significado sólo en sus descripciones textuales, no en los diagramas.

La representación UML de estos *roles* y relaciones se puede ver en las Fig. 8, Fig. 9 y Fig. 10. Las Fig. 8 y Fig. 9 reflejan la definición de UML-TA mediante *perfiles* UML y la Fig. 10 con una notación alternativa simplificada.

Los *perfiles* son uno de los dos mecanismos estándar para extender UML [OMG 2003]. Un perfil consta de un paquete UML, donde se describe qué estereotipos se asocian a entidades pertenecientes al metamodelo de UML, y expresiones OCL, que determinan las construcciones válidas de los diagramas. En este trabajo, las entidades del metamodelo que se estereotipan de forma especial son los *class* y *association*, que representan respectivamente una metaclase y una metaasociación.

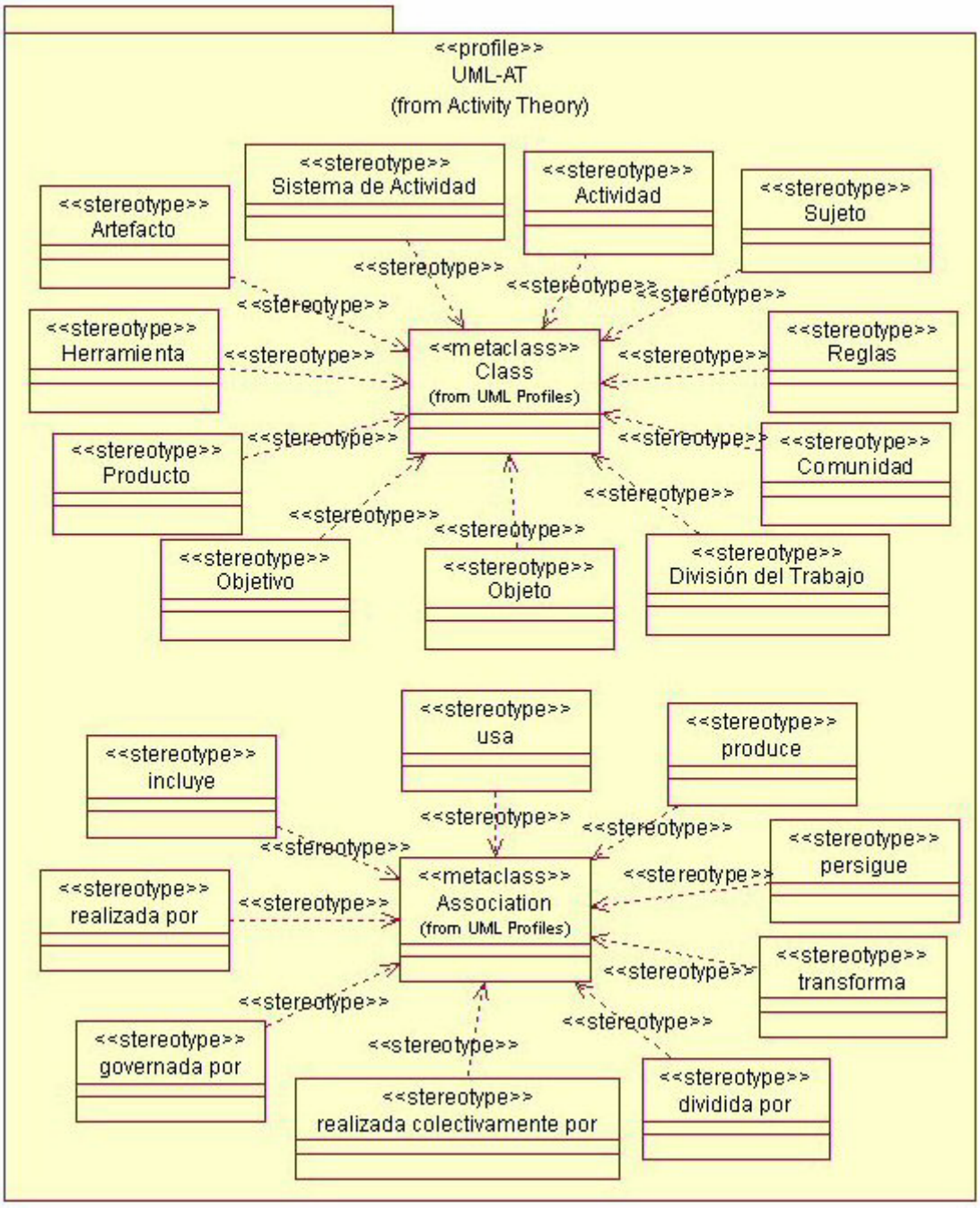


Fig. 8. Definición del lenguaje UML-TA mediante perfiles UML.

```
context UML::InfraestructureLibrary::Core::Constructs::
  Association
  inv:self.isStereotyped("incluye") implies
    self.connection ->
      exists(participant.isStereotyped("Artefacto") and
        multiplicity.min = 1 and multiplicity.max = n)
    and self.connection ->
      exists(participant.isStereotyped("Actividad") and
        multiplicity.min = 1 and multiplicity.max = n)
```

Fig. 9. OCL para la definición de la relación *incluye* en el perfil UML de UML-TA.

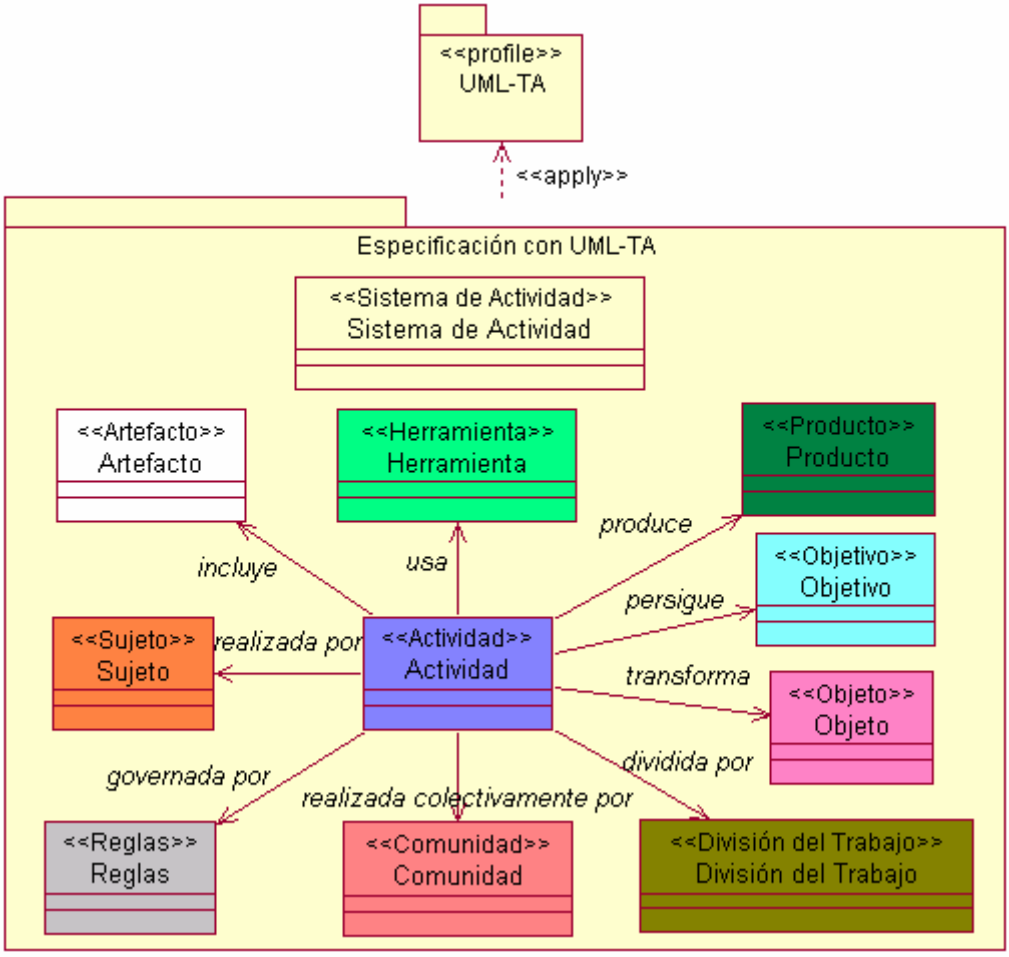


Fig. 10. Conceptos de la Teoría de Actividad y relaciones de mediación representados con UML.

La Fig. 8 describe los estereotipos para clases y relaciones introducidos por UML-TA a propósito del triángulo de la TA. Con esta representación gráfica se define el nuevo vocabulario. El otro elemento a representar son las restricciones que ha de cumplir ese vocabulario, que en este trabajo consisten en definir qué estereotipos puede unir una relación concreta. La Fig. 9 muestra un ejemplo de restricciones con OCL para las nuevas relaciones. Estas restricciones hacen referencia a los estereotipos que puede conectar la relación *incluye* y la cardinalidad de sus participantes.

En la Fig. 10, los conceptos del triángulo de la TA se modelan mediante estereotipos con el mismo nombre. La representación con estereotipos está complementada con un código de colores que facilita distinguir visualmente los *roles*. El estereotipo *artefacto* no tiene asociado un color fijo, puesto que se trata de un *rol* introducido para generalizar las descripciones. Si un *artefacto* puede adoptar varios posibles *roles* en el *sistema de actividad* estudiado se representa con el color blanco. En el caso de que el *artefacto* adopte un *rol* concreto se le puede asignar el color del *rol* particular que está representando, que será uno de los demás de la Fig. 10. Por ejemplo, en el caso de que el *artefacto* se instancie con una *regla*, usará el color del estereotipo *regla*. Las relaciones se dibujan como asociaciones UML entre aquellos conceptos para los que se admitirá su existencia.

Por simplicidad, el resto de esta sección adopta la propuesta de la Fig. 10 para introducir nuevo vocabulario. Se omite la definición explícita de las de entidades y relaciones mediante *perfiles*, utilizándose en su lugar la representación alternativa con estereotipos para las entidades y asociaciones UML para las relaciones.

Los *roles* anteriores y sus relaciones, directamente extraídos del triángulo de la TA (ver Fig. 4), no bastan para describir y razonar sobre las *actividades*. Por ejemplo, no permiten describir las relaciones de contribución entre *objetivos*, la descomposición de procesos o transformaciones específicas de *objetos* como la consumición. Para solventar este problema, y basándonos en la experiencia obtenida en el uso de la TA con SMAs, se introducen conceptos adicionales.

La primera notación que se presenta es la necesaria para cuantificar las relaciones (ver Fig. 11). Se introducen tres posibles adornos aplicables a cualquiera de las relaciones del vocabulario. Estos adornos permiten establecer a cuántas instancias de una relación es aplicable la información recogida en el patrón. El adorno *TODOS* significa que el patrón descrito se mantiene para cada posible *relación* del tipo dado entre el *Artefacto 1* y el *Artefacto 2*. El adorno *NO* puede considerarse la negación del adorno *TODOS* y significa que no existe ninguna *relación* de ese tipo entre los *artefactos Artefacto 1* y *Artefacto 2*. Por último, el adorno *ALGÚN* indica que existe al menos una instancia de la *relación* entre los *artefactos Artefacto 1* y *Artefacto 2*.

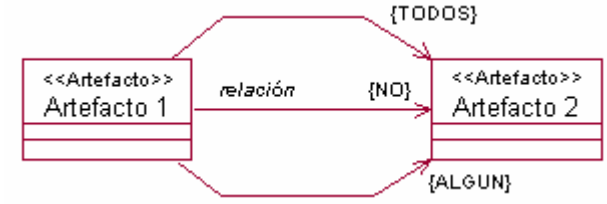


Fig. 11. Adornos para relaciones.

La relación de *cambio de rol* (ver Fig. 12) se emplea para indicar que un elemento de la especificación pasa a jugar dentro de un *sistema de actividad* un *rol* de TA diferente del que jugaba hasta ese momento. Este sería el caso de algunos *artefactos* compartidos entre *actividades vecinas*. Por ejemplo, el *producto* de una *actividad objeto* cambia su *rol* al de *objeto* cuando se le considera en la *actividad central*, aunque sigue tratándose del mismo elemento.

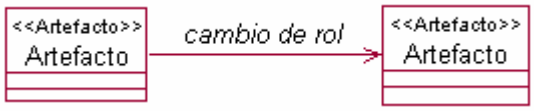


Fig. 12. Relación *cambio de rol*.

La relación de *herencia* entre *artefactos* (ver Fig. 13) hace referencia a que el *Artefacto 1* puede sustituir al *Artefacto 2* en todas sus apariciones. Por tanto, el *Artefacto 1* puede participar en todas las relaciones en las que participa el *Artefacto 2*.

Fig. 13. Relación de herencia.

La relación *juega* entre *sujetos* (ver Fig. 14) es una variante de la relación de *herencia*. Sólo se puede dar entre *sujetos* y tiene una semántica similar a la que normalmente se da a la relación juega entre agentes y roles en el paradigma de agentes. La Fig. 14 refleja que el *Sujeto 1* persigue los mismos *objetivos*, realiza las mismas *actividades* y forma parte de las mismas *comunidades* que el *Sujeto 2*. Esta relación también se puede emplear para los roles de agentes, ya que estos se modelan en este lenguaje como entidades que juegan el *rol sujeto* de la TA.

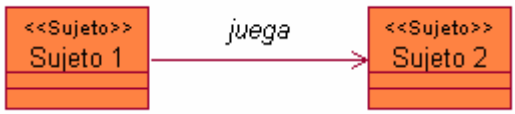


Fig. 14. Relación *juega* entre *sujetos*.

La relación *persigue* se ha definido anteriormente entre los *roles* de *actividad* y *objetivo* (ver Fig. 10). A través de la *actividad*, esta relación asocia a un *sujeto* con el *objetivo* de su necesidad que es satisfecho a través de la *actividad*. A fin de proporcionar una abreviatura apropiada a esta relación, se considera conveniente poder definir también la relación *persigue* directamente entre el *sujeto* y su *objetivo* como aparece en la Fig. 15. Puesto que una *comunidad* puede ser el *sujeto* colectivo de una *actividad*, también se ha definido la relación *persigue* entre una comunidad y su *objetivo* con la misma notación que para el *sujeto* y la *actividad* con sus *objetivos*.

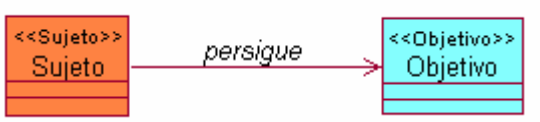


Fig. 15. Relación *persigue* para un *sujeto*.

Esta definición de *persigue* proporciona también el medio para asociar un *sujeto* con *objetivos* satisfechos por *actividades* que él no realiza.

Las relaciones de contribución muestran la influencia de los distintos *artefactos* sobre otros en cuanto a su construcción, satisfacción o realización, es decir, en cuanto al cumplimiento del papel asociado a su *rol* en el *sistema de actividad*. Ejemplos de estas contribuciones serían las relaciones de satisfacción entre *objetivos*, la construcción de *objetos*, el daño a *herramientas* o la ejecución de *actividades*. La nomenclatura de algunas de las relaciones de contribución en esta aproximación está inspirada en el trabajo de i* [Yu 1999]. Las relaciones de contribución consideradas para modelar la información relativa a la TA son: *garantiza*, *esencial*, *contribuye positivamente*, *indefinida*, *contribuye negativamente* e *impide*. Su representación en el lenguaje UML puede ser vista en la Fig. 16. La interpretación de esta representación es siempre que el *Artefacto 1* contribuye al *Artefacto 2*.

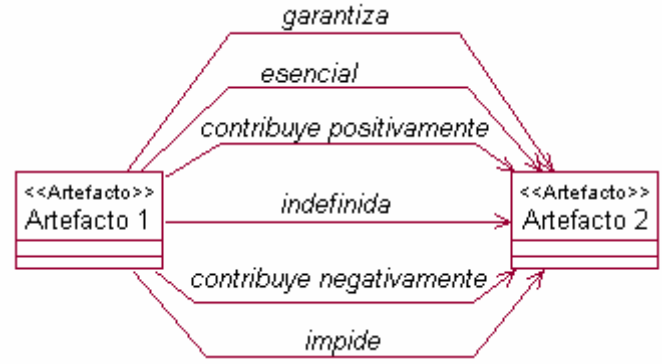


Fig. 16. Relaciones de contribución entre *artefactos*.

Una contribución *garantiza* implica que la presencia del *Artefacto 1* supone siempre que el *Artefacto 2* cumple su *rol*. En el caso de la contribución *esencial*, la presencia del *Artefacto 1* es imprescindible para que *Artefacto 2* cumpla con su *rol*, aunque su mera presencia no garantiza dicho cumplimiento. La *contribución positiva* significa que el *Artefacto 1* ayuda a que el *Artefacto 2* cumpla su *rol* pero no lo garantiza. La *contribución negativa* significa que el *Artefacto 1* dificulta que el *Artefacto 2* cumpla su *rol* pero no lo impide. Una contribución *impide* significa que la presencia del *Artefacto 1* inhibe completamente que el *Artefacto 2* cumpla su *rol*. Finalmente, una contribución *indefinida* indica que el *Artefacto 1* afecta al cumplimiento del *rol* del *Artefacto 2* aunque no se puede determinar cómo o depende de las circunstancias. En todos los casos no citados explícitamente, la presencia o ausencia del *Artefacto 1* no tiene efecto sobre cumplimiento del *rol* del *Artefacto 2*.

En ocasiones, la relación *persigue* entre una *actividad* y su *objetivo* se describe como *contribuye positivamente*. Esta clase de equivalencias son realizadas mediante traducciones entre patrones como se verá más adelante en la sección 3.4.2.

Indicar también que las relaciones de contribución pueden estar implícitas. Si el *producto* de una *actividad* contribuye a un *artefacto*, todos los elementos del correspondiente *sistema de actividad* contribuyen al *artefacto* de la misma forma que el *producto*, salvo que se explicite lo contrario.

En la Fig. 10 se introdujo la relación *persigue* entre la *actividad* y el *objetivo* que ésta trata de satisfacer. Esta relación no implica la satisfacción automática del *objetivo* cuando se ejecuta la *actividad*. A fin de diferenciar diferentes cursos de acción cuando se satisface o falla el *objetivo* se introducen dos nuevas relaciones *satisface* y *falla* (ver Fig. 17). La relación *satisface* indica un contexto en el que la ejecución de la *actividad* ha generado los *productos* necesarios para satisfacer las necesidades del *sujeto* y, por tanto, el *objetivo* asociado a ella. Por el contrario, la relación *falla* corresponde a un contexto en el que la *actividad* no ha podido satisfacer su *objetivo*. La no satisfacción del *objetivo* puede deberse a varias causas tales como la imposibilidad de completar la *actividad* o la destrucción de *objetos* necesarios por otras *actividades*.

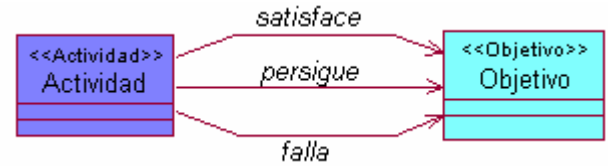


Fig. 17. Posibles relaciones de una *actividad* con el *objetivo* que persigue.

Las *actividades* relacionadas en un *sistema de actividad* son las llamadas *actividades vecinas*. Esencialmente se trata de un conjunto de *actividades* conectadas mediante asociaciones productor-consumidor, donde una *actividad* produce un *artefacto* que forma parte del sistema de otra *actividad*. En ocasiones los *artefactos* que actúan como conectores entre las *actividades* carecen de relevancia para el análisis de la situación. Para evitar la introducción de *artefactos* sin otro propósito que indicar la conexión entre *actividades* se introduce la relación *conecta* (ver Fig. 18). Esta relación significa que la *Actividad 1* genera algún *artefacto* que aparece en el sistema de la *Actividad 2*. Por tanto, también implica cierta precedencia en el tiempo de la *Actividad 1* respecto a la *Actividad 2*.

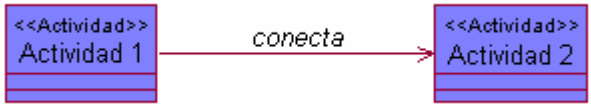


Fig. 18. Relación *conecta* entre *actividades*.

La ejecución de una *actividad* implica la utilización de distintos *artefactos*, por ejemplo la transformación de *objetos* o el uso de *herramientas*. En general se asume que estos recursos son sólo usados y siguen existiendo en el entorno después de la realización de la *actividad*. Sin embargo, en ocasiones es preciso indicar que la

actividad destruye el recurso al utilizarlo. Para representar esta clase de uso se introduce la relación *consume* (ver Fig. 19). Esta relación sólo es aplicable sobre *artefactos* que juegan el *rol* de *objeto* o *herramienta*.

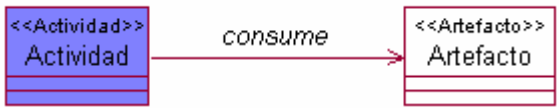


Fig. 19. Relación *consume*.

Otro aspecto a considerar en el vocabulario de la TA son las relaciones de descomposición entre *artefactos* (ver Fig. 20). Los conceptos de la TA pueden ser descritos a varios niveles de detalle. La introducción de estas relaciones permite considerar sólo la información relevante en cada momento del análisis de la *actividad*. Existen tres tipos de relaciones de descomposición: *descompone*, *descompone-Y* y *descompone-O*. La relación *descompone-Y* significa que el *Artefacto* está compuesto por la adición de los *artefactos Artefacto 1* y *Artefacto 2*. La relación *descompone-O* implica que el *Artefacto* puede ser sustituido bien por el *Artefacto 1* o por el *Artefacto 2*. Finalmente, la relación *descompone* puede representar a cualquiera de las dos anteriores.

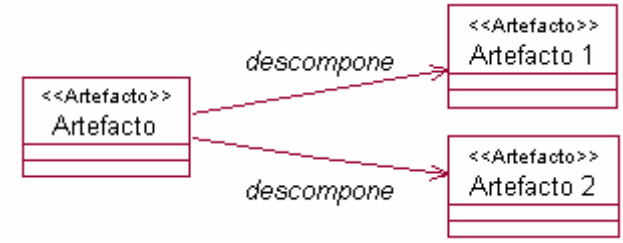


Fig. 20. Relación *descompone* entre *artefactos*.

Un aspecto importante a señalar acerca de las relaciones de descomposición es que sus restricciones y semántica no son exactamente iguales para todos los *roles* que se pueden considerar en un *sistema de actividad*. En el caso de *objetos*, *objetivos*, *productos*, *herramientas*, *sujetos* y *actividades*, la semántica es la introducida como general para esta clase de relaciones. Las relaciones de descomposición para estos *roles* sólo pueden conectar entidades que juegan ese mismo *rol*. Por ejemplo, un *objetivo* sólo puede ser descompuesto en otros *objetivos*. Esta semántica y restricciones cambian para la *comunidad*, las *reglas* y la *división del trabajo* .Una *comunidad* puede ser descompuesta en otras *comunidades* o bien en *sujetos*. En cuanto a las *reglas* y las *divisiones del trabajo*, el elemento básico para su especificación es el *sistema de actividad*. Las descripciones incluidas en estos conceptos pueden representarse como patrones de una *actividad* y sus elementos asociados. Por ello, las *reglas* pueden descomponerse en nuevos grupos de *reglas* o bien en *sistemas de actividad* y, del mismo modo, la *división del trabajo* puede descomponerse en otras *divisiones del trabajo* o bien en *sistemas de actividad*.

Otro aspecto importante de las relaciones *descompone* para trabajar con ellas en patrones es que pueden ser deducidas como ocurría con las contribuciones. Es decir.

Si un *Artefacto 1* se descompone en un *Artefacto 2* y éste en un *Artefacto 3*, se puede deducir la relación *descompone* entre el *Artefacto 1* y el *Artefacto 3*.

El último elemento añadido al vocabulario es la relación *supera* entre *objetivos* (ver Fig. 21). Esta relación se emplea para indicar la prioridad en el cumplimiento de los *objetivos*. Indica que para el *sujeto* el *Objetivo 1* tiene más importancia que el *Objetivo 2*. Así, en caso de conflicto entre ambos, se debería satisfacer el *Objetivo 1* y, sólo si éste se ha satisfecho y no se ve perjudicado, intentar satisfacer el *Objetivo 2*.

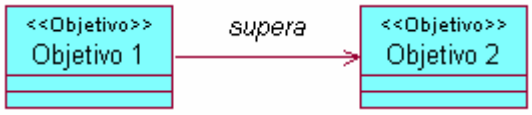


Fig. 21. Relación *supera* entre *objetivos*.

El vocabulario de este apartado se emplea para representar el conocimiento extraído de la TA mediante *patrones estructurales*. Un *patrón estructural* en este trabajo es un diagrama descrito con el lenguaje introducido en el presente apartado. La información de la TA, descrita en lenguaje natural, es interpretada y traducida a su representación mediante *patrones estructurales*. Esta representación no es única y depende la interpretación del traductor. Por ello se admite la posibilidad de múltiples representaciones para una misma información original.

Los patrones pueden incluir elementos con valores variables o fijos, tales como *roles* de TA y nombres de entidades o relaciones. Para indicarlos se sigue la convención de representar entre ‘ “ ’ y ‘ ” ’ (comillas dobles) los valores que deban ser interpretados como literales, i.e. valores fijos, y sin comillas los que han de ser interpretados como variables.

3.4.2. Equivalencias en UML-TA

El apartado anterior ha presentado las primitivas de modelado de UML-TA. Estas primitivas serán usadas para construir patrones estructurales que describirán las *propiedades sociales* para SMAs. El otro elemento que se considera en la definición del lenguaje son las equivalencias entre conceptos.

Las equivalencias entre conceptos aparecen para simplificar el tratamiento de los patrones de las *propiedades sociales*. Indican cuando un conjunto de elementos de un diagrama UML-TA puede ser sustituido por otro sin alterar el significado de la especificación. Con este conocimiento genérico para el lenguaje se evita multiplicar las descripciones de propiedades con patrones que emplean diferentes primitivas pero que tienen idéntico significado. En lugar de describir todas las posibilidades de un patrón, se usa un patrón genérico que puede ser adaptado a distintas situaciones mediante las equivalencias.

Un ejemplo de equivalencia sería el de *actividades* unidas por la relación *conecta* que son sustituidas por una sola *actividad*. La sustitución es correcta porque una serie de *actividades*, vistas en un nivel de detalle menor, pueden ser representadas por una sola *actividad*. Además resulta inviable construir patrones que contemplen cadenas de *actividades* unidas por *conecta* de longitud arbitraria. En su lugar se define un solo patrón con una *actividad* y se adapta a otros casos con la relación de equivalencia.

Original	Sustituto	Observaciones
Artefacto	Cualquier concepto	Ausencia de restricciones explícitas
Cualquier concepto	*Artefacto*	
Sistema de Actividad	*Actividad*	
Actividad	*Sistema de Actividad*	
Artefacto (destino)	*Artefacto* (origen)	*Artefacto* (origen) → *herencia* → *Artefacto* (destino)
Sujeto (origen)	*Sujeto* (destino)	*Sujeto* (origen) → *juega* → *Sujeto* (destino)
Actividad → *satisface* → *Objetivo*	*Actividad* → *garantiza* → *Objetivo*	
Actividad → *persigue* → *Objetivo*	*Actividad* → *contribuye positivamente* →*Objetivo*	
Actividad → *falla* → *Objetivo*	*Actividad* → *impide* → *Objetivo*	
Actividad (origen) → *conecta* → *Actividad* (destino)	*Actividad*	
consume	*incluye* *usa* *transforma*	
Artefacto (destino)	*Artefacto* (origen)	*Artefacto* (origen) → *descompone* → *Artefacto* (destino)
Artefacto (origen)	*Artefacto* (destino)	*Artefacto* (origen) → *descompone* → *Artefacto* (destino)
Artefacto (destino)	*Artefacto* (origen)	*Artefacto* (origen) → *descompone-O* → *Artefacto* (destino)
Artefacto (origen)	*Artefacto* (destino)	*Artefacto* (origen) → *descompone-O* → *Artefacto* (destino)
Artefacto (destino1) + *Artefacto* (destino2)	*Artefacto* (origen)	*Artefacto* (origen) → *descompone-Y* → *Artefacto* (destino1)+*Artefacto* (destino2)

Tabla 5. Equivalencias entre conceptos en UML-TA.

La Tabla 5 describe las equivalencias en el lenguaje UML-TA. Para cada equivalencia se indica el conjunto de conceptos original, su sustituto y observaciones adicionales a la sustitución. Como convención para la Tabla 5, los *roles* de la TA se escriben en cursiva con su primera letra mayúscula y las relaciones en cursiva y completamente en minúsculas. En los casos en los que hay que distinguir entre dos

elementos que juegan el mismo *rol*, se asigna un nombre entre paréntesis a cada uno de esos elementos.

El apartado anterior ha presentado las primitivas de modelado de UML-TA. Estas primitivas serán usadas para construir patrones estructurales que describirán las *propiedades sociales* para SMAs. El otro elemento que se considera en la definición del lenguaje son las equivalencias entre conceptos.

Las equivalencias entre conceptos aparecen para simplificar el tratamiento de los patrones de las *propiedades sociales*. Indican cuando un conjunto de elementos de un diagrama UML-TA puede ser sustituido por otro sin alterar el significado de la especificación. Con este conocimiento genérico para el lenguaje se evita multiplicar las descripciones de propiedades con patrones que emplean diferentes primitivas pero que tienen idéntico significado. En lugar de describir todas las posibilidades de un patrón, se usa un patrón genérico que puede ser adaptado a distintas situaciones mediante las equivalencias.

Un ejemplo de equivalencia sería el de *actividades* unidas por la relación *conecta* que son sustituidas por una sola *actividad*. La sustitución es correcta porque una serie de *actividades*, vistas en un nivel de detalle menor, pueden ser representadas por una sola *actividad*. Además resulta inviable construir patrones que contemplen cadenas de *actividades* unidas por *conecta* de longitud arbitraria. En su lugar se define un solo patrón con una *actividad* y se adapta a otros casos con la relación de equivalencia.

La Tabla 5 describe las equivalencias en el lenguaje UML-TA. Para cada equivalencia se indica el conjunto de conceptos original, su sustituto y observaciones adicionales a la sustitución. Como convención para la Tabla 5, los roles de la TA se escriben en cursiva con su primera letra mayúscula y las relaciones en cursiva y completamente en minúsculas.

Pese a la introducción de este nuevo vocabulario UML-TA, el objetivo de esta investigación no es introducir nuevas primitivas para el modelado de agentes, sino posibilitar el uso de técnicas analíticas de las ciencias sociales en metodologías de la ISOA. Para ello es necesario definir un mecanismo de traducción entre los conceptos de la TA y los utilizados en las metodologías de agentes, tal y como se presenta a continuación.

3.5. Traducción entre la TA y conceptos de agentes

En el apartado anterior se ha introducido una notación basada en UML para representar los conceptos de la TA identificados como relevantes para este trabajo. En el proceso de adaptación reflejado en la Fig. 7, los pasos siguientes para poder utilizar la TA en un proceso software para SMAs son establecer la relación entre los *roles* de la TA y los conceptos de la correspondiente metodología de desarrollo, así como el proceso que usa estas correspondencias para traducir especificaciones entre ambos lenguajes. Éstos son los aspectos que desarrolla esta sección.

3.5.1. Correspondencias entre vocabularios

El requisito necesario para poder establecer correspondencias con la TA es que la metodología de desarrollo conciba sus sistemas en una forma similar a como la TA concibe las organizaciones humanas (ver sección 3.2). Es decir, los sistemas han de ser contemplados como organizaciones de actores intencionales que persiguen satisfacer objetivos mediante la ejecución de tareas en las cuales emplean recursos e interaccionan con otros actores. Este requisito se satisface en la mayoría de las propuestas basadas en el paradigma de agentes. Los SMAs suelen ser descritos como sistemas sociales constituidos por agentes intencionales [Maes 1994, Sykara 1998]. En estos SMAs, los objetivos globales del sistema sólo pueden alcanzarse mediante la interacción de sus componentes [Jennings & Wooldridge 2000]. Para entender el sistema de esta forma hay que considerarlo a nivel de organización y no como una mera agregación de elementos [Demazeau 1995, Ferber 1999]. Además, los componentes de estos sistemas son intencionales, en el sentido de que satisfacen el Principio de Racionalidad [Newell 1982]: Cada acción tomada por un agente le ayuda a satisfacer uno de sus objetivos. Por tanto, como la TA y el paradigma de agentes modelan sistemas intencionales y sociales en términos de abstracciones similares, pueden establecerse relaciones entre dichas abstracciones que guíen un proceso de traducción entre la TA y conceptos basados en el paradigma de agentes.

Para facilitar la construcción de las correspondencias, la Tabla 6 muestra una concreción de los conceptos principales de la TA (ver Fig. 10) en términos de conceptos habituales en el paradigma de agentes. El objetivo de esta reformulación es dar una idea de la clase de conceptos en una metodología para SMAs que pueden corresponderse con los de la TA. Las relaciones presentadas en la sección 3.4.1 no están incluidas en la Tabla 6.

Las definiciones de la Tabla 6 merecen algunas observaciones acerca de la identificación de sus conceptos sobre las especificaciones de SMAs. Se puede observar que la *actividad*, el *sujeto* y el *objetivo* son *roles* de la TA con definiciones muy restrictivas, que pueden ser jugados por pocos elementos de un SMA. Estos *roles* son por tanto un buen punto de partida a la hora de identificar los elementos de un *sistema de actividad*. La *comunidad* es otro *rol* de la TA bastante limitado en cuanto a sus posibles jugadores en un SMA. Sólo puede estar constituida por *sujetos* que comparten participación en algún flujo de trabajo. Por el contrario, el *objeto*, el *producto* y la *herramienta* son *roles* difíciles de identificar en un SMA. Aunque sus definiciones son claras pueden ser desempeñados por elementos muy diversos, prácticamente por cualquiera empleado en el modelado de SMAs, y pueden ser traducidos a casi los mismo grupos de abstracciones. Además, en el caso del *objeto* y la *herramienta*, si se desconoce la semántica precisa de la *actividad* considerada es muy difícil determinar cuál es el elemento transformado en el *producto*, i.e. el *objeto*, y cuál sólo ayuda en la transformación del *objeto*, i.e. la *herramienta*. Las *reglas* y la *división del trabajo* definen patrones de comportamiento en el SMA. En muchas metodologías de SMAs, ambos se definen mediante la asignación de roles (no de la TA sino las entidades de modelado en SMAs) a agentes, bien de forma implícita o explícita. La diferencia entre ambos grupos de normas hay que buscarla en el *objeto* de la *actividad*. La *división del trabajo* determina la organización de la *comunidad* para transformar el *objeto* mientras que las *reglas* son de ámbito más genérico. La

última abstracción a considerar es el *sistema de actividad*. Puesto que un *sistema de actividad* puede estar constituido por múltiples *actividades* interconectadas y descritas con diferentes niveles de detalle, debe considerarse esta abstracción sobre todo como una forma útil de agrupar conceptos que no son relevantes en un nivel de análisis dado.

Conceptos de Teoría de Actividad	Definición en términos del paradigma de agentes
Sistema de Actividad	Conjunto de elementos de modelado que representan una vista total o parcial de cierta información de un SMA.
Actividad	Tarea que persigue un objetivo relevante para algún agente o grupo de agentes en el SMA.
Sujeto	Agente. Por extensión se puede considerar *sujeto* a cualquier elemento activo del sistema que pueda modelarse como capaz de satisfacer el Principio de Racionalidad,. e.g. roles, grupos u organizaciones.
Objeto	Elemento transformado por una tarea y que puede ser considerado como el origen directo del resultado de esa tarea. Se trata de un prerrequisito de la tarea.
Producto	Resultado de una tarea que satisface el objetivo de la misma o constituye la evidencia sobre la que evaluar dicho objetivo.
Objetivo	Necesidad que satisface una tarea.
Herramienta	Cualquier recurso empleado por una tarea que no sea ni su *objeto* ni su *producto*. Se trata de un prerrequisito de la tarea.
Comunidad	Conjunto de agentes que interactúan en los flujos de trabajo de un SMA.
Reglas	Conjunto de normas no incluido en la *División del Trabajo* que se aplica al funcionamiento del SMA.
División del Trabajo	Organización de los agentes para llevar a cabo los flujos de trabajo del sistema. Incluye las capacidades de los agentes, los recursos que controlan y sus relaciones de poder.
Artefacto	Cualquier elemento empleado en el modelado de un SMA.

Tabla 6. Redefinición de los conceptos de la TA en términos del paradigma de agentes.

A partir de la información contenida en la Tabla 6 y con las guías que se acaban de dar, el proceso de creación de las correspondencias consiste en el estudio sistemático de las definiciones de los conceptos en ambos grupos para determinar cuáles pueden ser sus posibles traducciones en el otro paradigma. Obsérvese que la traducción correcta de un concepto puede depender de otros elementos de las especificaciones. Así, elegir entre traducir un elemento por el *rol herramienta* o el de *objeto* depende de si dicho elemento satisface o no un *objetivo* en la *actividad* que se está considerando. Por tanto, la traducción debe considerar correspondencias entre estructuras y no sólo entre elementos individuales.

A modo de ejemplo, los siguientes apartados muestran el estudio sobre correspondencias para las metodologías INGENIAS [Pavón & Gómez-Sanz 2003] y Tropos [Perini *et al.* 2001], ya revisadas en Capítulo 2. INGENIAS es la metodología con la que se ha desarrollado la mayor parte de la experimentación de esta tesis y con Tropos se ha probado que las técnicas propuestas no están limitadas a una metodología concreta de la ISOA. A fin de limitar la extensión de la posterior discusión no se incluyen las relaciones y sólo se consideran las posibles correspondencias entre elementos individuales.

3.5.1.1. TA-INGENIAS

La metodología INGENIAS [Pavón & Gómez-Sanz 2003] (ver sección 2.3.8) describe los SMAs mediante un conjunto de vistas y un vocabulario asociado. Sus principales conceptos aparecen en la Tabla 7 junto con una breve definición.

Los aspectos particulares del vocabulario de INGENIAS son: la distinción entre *organizaciones* (agrupaciones estáticas) y *grupos* (agrupaciones dinámicas en las *organizaciones*); la inclusión de múltiples entidades para reflejar el estado mental de los agentes tales como *creencias*, *consultas de entidades autónomas* o *hechos*.

INGENIAS	**Definición**
Agente	Entidad que sólo maneja conocimiento y que se gobierna de acuerdo con el Principio de Racionalidad.
Aplicación	Dispositivo del entorno que, mediante las operaciones que soporta, permite obtener información de dicho entorno y actuar sobre él.
Consulta de entidades autónomas	Descripción de las asociaciones reales entre *agentes* concretos y *roles* en una *interacción* dada.
Creencia	Información subjetiva disponible para el *agente*. Proviene de su interpretación de los datos recibidos en sus elementos perceptores. Se etiqueta con el nombre de la evidencia e información acerca de lo que se está aceptando como cierto.
Estado Mental	Conjunto de entidades mentales conocidas por el *agente*. Las entidades mentales son: *hechos*, *objetivos*, *creencias* y *eventos*.
Evento	Cambio ocurrido en el mundo que el *agente* capta.
Flujo de Trabajo	Conjunto de *tareas* estructuradas en secuencias o alternativas. El encadenamiento de las *tareas* viene dado por la creación y consumo de los productos de otras *tareas*.
Grupo	Conjunto de *agentes*, *roles*, *objetivos*, *tareas*, *recursos* y sus relaciones, creados dinámicamente en el seno de una *organización*.
Hecho	Información que resulta inherentemente cierta en el entorno, e.g. leyes físicas o teoremas matemáticos, o información resultante de la ejecución de *tareas*.
Interacción	Descripción de un flujo de trabajo considerando la motivación de los participantes y las tareas involucradas.
Objetivo	Estado del mundo que el *agente*, u otra entidad intencional como un *rol* u *organización*, desea o se compromete a alcanzar.
Organización	Conjunto de carácter estático de *agentes*, *roles*, *objetivos*, *tareas* y

INGENIAS	Definición
	recursos cuya estructura es dada por los *grupos* que se crean en ella.
Recurso	Elementos que forman parte del entorno. Se caracterizan por la cantidad disponible y los límites inferior y superior admisibles para ellos. Por debajo o encima de estos límites, el *recurso* se deshabilita.
Rol	Agrupamiento de *objetivos* y estado mental
Tarea	Método para resolver un problema que cuenta con precondiciones y postcondiciones conocidas.
Unidad de interacción	Tipo de mensaje que hace referencia a un acto del habla, como *request*, *inform* o *not-understood*.

Tabla 7. Conceptos de INGENIAS.

Teoría de Actividad	INGENIAS
Sistema de Actividad	Tarea, Flujo de trabajo, Interacción
Actividad	Tarea, Flujo de trabajo, Interacción
Sujeto	Agente, Rol
Objeto	Recurso, Aplicación Interacción, Unidad de Interacción Hecho, Creencia, Evento Estado mental, Consulta de entidades autónomas
Producto	Recurso, Aplicación Interacción, Unidad de Interacción Hecho, Creencia Estado mental, Consulta de entidades autónomas
Objetivo	Objetivo, Estado mental
Herramienta	Recurso, Aplicación Tarea, Flujo de trabajo, Interacción, Unidad de Interacción Hecho, Creencia, Evento Estado mental, Consulta de entidades autónomas
Comunidad	Grupo, Organización
Reglas	Estado mental, Consulta de entidades autónomas Interacción Grupo Hecho, Creencia
División del Trabajo	Estado mental, Consulta de entidades autónomas Interacción Grupo Hecho, Creencia

Tabla 8. Correspondencias entre conceptos de TA e INGENIAS.

Dadas las definiciones para los conceptos de la TA (ver Tabla 6) y de la metodología INGENIAS (ver Tabla 7), se establecen las correspondencias propuestas en la Tabla 8. Una descripción completa del vocabulario de INGENIAS puede encontrarse en http://grasia.fdi.ucm.es. En general puede apreciarse que, debido a lo

reducido del vocabulario de la TA, varios conceptos de INGENIAS se pueden corresponder con las mismas abstracciones de la TA. El caso contrario, donde varios conceptos de la TA pueden traducirse a un mismo concepto de INGENIAS se debe a la posibilidad de que éste juegue varios *roles* en un *sistema de actividad*.

El proceso de creación de las correspondencias comienza identificando aquellos elementos de la TA que resultan más específicos, concretamente el *sujeto*, la *actividad* y el *objetivo*.

Un *sujeto* para la TA es aquel elemento que satisface el Principio de Racionalidad, es decir, ejecuta tareas buscando satisfacer sus objetivos. En esta categoría pueden incluirse dos abstracciones de INGENIAS, el *agente* y el *rol*. Ambos se modelan con objetivos y tareas asociadas y su toma de decisiones siempre satisface el Principio de Racionalidad.

En cuanto a la *actividad*, ésta se define como una tarea que satisface el objetivo de algún *sujeto*. De acuerdo con las relaciones definidas en INGENIAS, el concepto *tarea* entraría dentro de esta categoría. El *flujo de trabajo* no tiene ninguna relación de satisfacción con objetivos, aunque participa en una relación *consume* con ellos. Dado que el *flujo de trabajo* incluye varias *tareas* y no se exige que todas tengan objetivos asociados, la única forma de que los *flujos* y sus *tareas* satisfagan el Principio de Racionalidad es interpretando esta relación de consumo del *flujo* como una relación con los objetivos que persigue globalmente. También es posible considerar que un *flujo* persigue los objetivos de las *tareas* que forman parte de él. Bajo esta óptica, el *flujo de trabajo* también debería ser incluido como una posible traducción de la *actividad*. Por último, una *interacción* de INGENIAS también es una *actividad*, ya que se trata de un conjunto de *tareas* ordenadas que se ejecutan persiguiendo un propósito, que puede ser global para la *interacción* o bien individual para sus participantes.

El *objetivo* de la TA es una abstracción que representa un estado del mundo deseado por el agente. Estos estados pueden ser representados en INGENIAS con *objetivos* que representan por sí solos estos estados. Otra posibilidad permitida en INGENIAS es representar estos *objetivos* de la TA a través de *estados mentales* del *agente*. De este modo se posibilita una descripción más detallada del estado a alcanzar con la notación gráfica de INGENIAS y se reduce el papel de los lenguajes natural y de restricciones en la descripción.

En cuanto a la *comunidad*, INGENIAS soporta de manera explícita la definición de grupos de agentes o roles mediante los conceptos de *grupos* y las *organizaciones*. Estas *comunidades* comparten recursos y objetivos por lo que sus componentes suelen participar en los mismos flujos de trabajo. También existe la posibilidad de tener *comunidades* implícitas, definidas por los *flujos de trabajo*, que no cuentan con una representación en *organizaciones* o *grupos*.

Como ya se ha señalado, *objetos*, *productos* y *herramientas* admiten traducciones muy similares. Cualquier elemento que pueda ser usado en una *actividad* puede desempeñar el *rol herramienta* de la TA. Por tanto entrarían en esta categoría los conceptos de INGENIAS *recurso*, *aplicación*, *tarea*, *flujo de trabajo*, *interacción*, *unidad de interacción*, *hecho*, *creencia*, *evento*, *estado mental* y *consulta de entidades autónomas*. En cuanto al *objeto* y el *producto*, podrían ser traducidos casi a los mismos conceptos que las *herramientas*, ya que la mayoría pueden ser manipulados en una *tarea*. Las traducciones no permitidas son a *tarea*, *flujo de trabajo*, *interacción*

y *unidad de interacción*. Además, un *evento* no puede ser el *producto* de una *actividad*, porque en INGENIAS surgen del entorno y no son generados por *actividades*.

Las *reglas* y la *división del trabajo* pueden ser especificadas en INGENIAS con distintas abstracciones: *estado mental*, *consulta de entidades autónomas*, *interacción*, *grupo*, *hecho* y *creencia*. Todas ellas permiten describir patrones de comportamiento para los agentes del SMA y son por tanto adecuadas para establecer normas del sistema.

Por último, un *sistema de actividad*, en cuanto descripción posiblemente parcial de una o varias *actividades*, puede ser representado explícitamente en INGENIAS por los conceptos *tarea*, *flujo de trabajo* e *interacción*. Por supuesto, también se pueden considerar *sistemas de actividad* implícitos como agrupaciones de cualesquiera elementos presentes en la especificación de un SMA.

3.5.1.2. TA-Tropos

Tropos	Definición
Actor	Entidad con objetivos estratégicos que sigue el Principio de Racionalidad.
Agente	*Actor* con autonomía, habilidades sociales, reactividad y proactividad.
Creencia	Elemento de información que representa el conocimiento del *actor* acerca del mundo.
Capacidad	Habilidad de un *actor* para definir, elegir y ejecutar un *plan* que satisface un *objetivo* en un entorno operativo determinado.
Entidad	Elemento genérico que representa conceptos intencionales y no intencionales de la organización y del entorno.
Objetivo	Representa un interés de un *actor*, un estado del mundo que desea alcanzar.
Objetivo *hard*	*Objetivo* cuyas condiciones de satisfacción están claramente identificadas y pueden ser observadas.
Objetivo *soft*	*Objetivo* sin unos criterios claros de satisfacción.
Plan	Método para satisfacer un *objetivo*.
Posición	*Actor* que representa un conjunto de *roles* generalmente jugados por un *agente*.
Recurso	Entidad física o de información que un *actor* puede proporcionar a otro que la necesite.
Rol	Caracterización abstracta del comportamiento de un *actor* en un contexto dado.
Tarea	Proceso atómico que ejecuta un *actor* para satisfacer un *objetivo*.

Tabla 9. Conceptos de Tropos.

La metodología Tropos [Perini *et al.* 2001] (ver sección 2.3.6) para la especificación de SMAs emplea vistas al igual que INGENIAS. El vocabulario usado en estas vistas procede principalmente de las abstracciones usadas para modelar los

requisitos iniciales de los SMAs. Los conceptos de Tropos pueden verse en la Tabla 9 junto con una breve definición. Al igual que se ha hecho en el caso de INGENIAS (ver sección 3.5.1.1), sólo se contemplan las entidades, quedando excluidas las relaciones de esta exposición. Para una enumeración completa del vocabulario de Tropos puede verse [Sannicolo' *et al.* 2002].

Los aspectos particulares del vocabulario de Tropos son: la diferenciación entre *objetivos hard* (que pueden ser evaluados cuantitativamente) y *soft* (que sólo pueden ser evaluados cualitativamente); la distinción entre *actores* (entidades intencionales), *agentes* (en el sentido de [Jennings & Wooldridge 2000]) y *posiciones* (agrupación de *roles* de Tropos).

A partir de las definiciones de los conceptos de la TA en términos del vocabulario común en el paradigma de agentes (ver Tabla 6) y de la descripción de las abstracciones de la metodología Tropos (ver Tabla 9), se proponen las correspondencias de la Tabla 10 entre ambos grupos de conceptos.

Teoría de Actividad	**Tropos**
Sistema de Actividad	Plan
Actividad	Capacidad, Plan, Tarea
Sujeto	Actor, Agente, Posición, Rol
Objeto	Entidad, Creencia, Objetivo, Objetivo *hard*, Objetivo *soft* Plan Recurso
Producto	Entidad, Creencia, Objetivo, Objetivo *hard*, Objetivo *soft* Plan Recurso
Objetivo	Objetivo, Objetivo *hard*, Objetivo *soft*
Herramienta	Actor, Agente, Posición, Rol Entidad, Creencia, Objetivo, Objetivo *hard*, Objetivo *sofá* Plan Recurso
Comunidad	Actor, Agente, Posición, Rol
Reglas	Entidad, Creencia Capacidad, Plan Posición
División del Trabajo	Entidad, Creencia Capacidad Posición

Tabla 10. Correspondencias entre conceptos de TA y Tropos.

Las entidades de Tropos que pueden jugar el *rol* de *sujeto* de la TA son todas las que se comportan según el Principio de Racionalidad. Se trata de *actor*, *agente*, *posición* y *rol*.

Las *actividades* serán aquellas tareas que persiguen objetivos. En el caso de Tropos estas abstracciones son las de *capacidad*, *plan* y *tarea*.

Señalar también que la TA no traslada la distinción entre *objetivos soft* y *hard* de Tropos. En la TA todo el razonamiento sobre los objetivos es cualitativo.

Por último, y al igual que ocurría en INGENIAS, en las correspondencias de Tropos también se aprecia que los conceptos de *objeto*, *producto* y *herramienta* abarcan un amplio espectro de usos en la TA. Por ello admiten correspondencias con numerosas abstracciones de Tropos, concretamente con todas las que puedan ser manipuladas por abstracciones que juegan el *rol actividad* de la TA.

3.5.2. Consideraciones para el proceso de traducción

Tras introducir la forma de establecer las correspondencias entre los conceptos básicos de la TA y los de las metodologías de SMAs, se puede definir el proceso de traducción entre lenguajes. Para ello se han de tener en cuenta ciertas reflexiones, surgidas a propósito de las correspondencias, que se discuten a continuación:

- *Traducciones específicas para las metodologías de SMAs*. Es posible establecer una correspondencia con conceptos genéricos del paradigma de agentes, de amplia aceptación, para aplicar las técnicas de la TA sobre procesos software. Sin embargo, el vocabulario empleado en el modelado no es el mismo en todas las metodologías de agentes y cada una confiere matices semánticos particulares a sus abstracciones. Como se ha visto en la sección anterior, las correspondencias consideran el vocabulario específico de las metodologías para incluir en las traducciones esos matices. En todo caso, estas correspondencias sólo se tienen que construir una vez para una metodología de SMAs dada. Luego se pueden reutilizar en distintos proyectos para traducir modelos de la metodología al lenguaje de la TA y viceversa.
- *Correspondencias muchos a muchos*. Las correspondencias entre conceptos de la TA y los de las metodologías de agentes no son biunívocas; por el contrario, una entidad de cualquiera de los dos grupos suele admitir varias traducciones en el otro. Ello se debe a dos motivos:
    - *Los lenguajes no tienen el mismo nivel de detalle en todos los aspectos*. Así, mientras que para la TA sólo se habla de *sujetos*, es posible que la metodología de SMAs distinga entre *agentes* y *organizaciones*. Del mismo modo, una *tarea* en un SMA puede ser para la TA, según el punto de vista y sus motivos, elementos tan dispares como una *actividad*, una *herramienta* o un *objeto*.
    - *Multiplicidad de roles en un sistema de actividad para un elemento de un SMA*. Un concepto de SMAs puede jugar varios *roles* de la TA en un *sistema de actividad* y, por tanto, tener varias traducciones a la TA. El *rol* de la TA depende de la porción del sistema que se está analizando y del punto de vista que se está adoptando.
- *Traducción de estructuras*. El proceso de traducción no considera sólo elementos individuales, sino que puede ser necesario traducir estructuras complejas. La traducción pone en correspondencia grupos de elementos de un lenguaje con grupos del otro lenguaje a cuya definición se ajustan. Puesto que las abstracciones son diferentes, los grupos significativos no tienen que estar constituidos necesariamente por un único elemento. Por ejemplo, las tareas individuales de una secuencia de tareas, donde sólo la última tiene asociada un

objetivo, no pueden ser traducidas a *actividades*; sólo la secuencia completa de tareas satisface una necesidad y puede ser por tanto traducida a una *actividad*.

Estas consideraciones sobre el proceso de traducción se reflejan en el método que se describe en la sección siguiente.

3.5.3. Proceso de traducción

Partiendo de estas consideraciones se ha optado por el proceso de traducción de las especificaciones basado en niveles que se puede ver en la Fig. 22. El proceso tiene como parámetros las especificaciones a traducir y correspondencias entre estructuras organizadas en *niveles de traducción*.

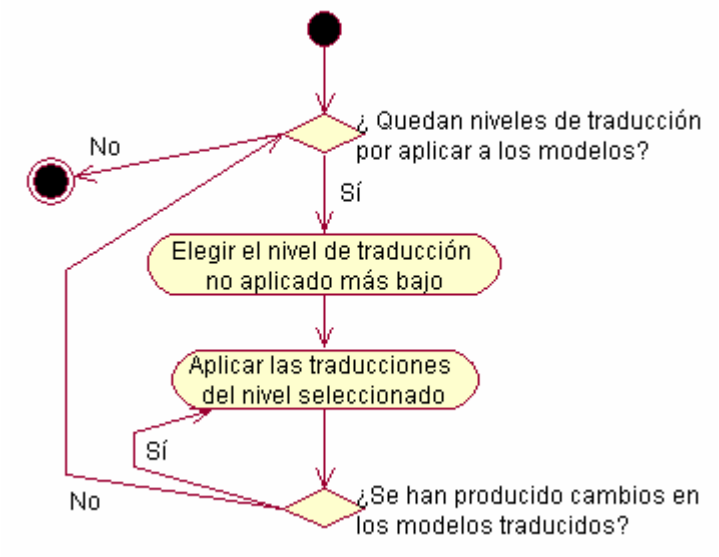


Fig. 22. Proceso de traducción entre lenguajes.

Un *nivel de traducción* agrupa un conjunto de correspondencias y tiene asociado un número de orden. Los *niveles de traducción* se aplican según su número de orden, comenzando por los ordinales más bajos. Las correspondencias dentro de un nivel dado se aplican secuencialmente, sin importar el orden entre ellas. Las correspondencias de un nivel pueden hacer uso en sus traducciones de todos aquellos conceptos que sean traducidos por correspondencias incluidas en niveles de traducción más bajos que el suyo. Son estos prerrequisitos de traducción los que determinan el número de orden de los niveles.

Realizar el proceso de traducción con *niveles de traducción* cubre las necesidades especificadas en la sección 3.5.2:

- *Traducciones específicas para las metodologías de SMAs*. Las correspondencias, que son dependientes de la metodología, son un parámetro del proceso de traducción.

- *Correspondencias muchos a muchos*. Un *nivel de traducción* incluye varias correspondencias aplicadas sin orden prefijado. Dentro de un nivel se pueden incluir varias traducciones posibles para un mismo grupo de conceptos original.
- *Traducción de estructuras*. La traducción de estructuras se puede realizar directamente mediante su especificación con una correspondencia entre dos grupos de elementos. En los casos complejos se puede realizar a través de varias iteraciones con traducciones parciales. Estas iteraciones pueden necesitar como requisitos que ya estén traducidos ciertos elementos. El control de estos requisitos se hace agrupando los pasos de traducción en niveles. Este es el caso de las relaciones y las entidades que asocian. En lugar de traducir las relaciones considerando todos los posibles casos de tipos entidades que pueden asociar, se pueden traducir en primer lugar las entidades y en una segunda iteración las relaciones atendiendo a las entidades ya traducidas.

Aunque en esta sección se ha discutido sobre la traducción entre el lenguaje de la TA y otros basados en el paradigma de agentes, el proceso presentado puede ser usado para dar equivalencias dentro de un mismo lenguaje. Así, en el caso citado en la sección 3.4 acerca de la equivalencia de la relación *persigue* entre una *actividad* y su *objetivo* con la relación *contribuye positivamente*, podría emplearse este mecanismo de traducción para reflejarla en las especificaciones.

3.6. Capturando información con patrones estructurales

Los apartados previos de este capítulo han introducido el marco original de la TA, un vocabulario para representar sus conceptos y guías para traducir dichos conceptos a los propios de las metodologías de agentes. En este punto ya se está en condiciones de abordar la descripción de las *propiedades sociales* descritas en la TA en una forma aplicable al desarrollo de SMAs.

La sección 3.3 definió las *propiedades sociales* de la TA para SMAs como configuraciones que describían 1) información que debía estar presente en unas especificaciones, 2) contextos sociales que podían aparecer en un SMA y 3) posibles cambios en las especificaciones si aparecían estos contextos. Desde el punto de vista del desarrollo de software, esta información puede ser aplicada de diferentes formas en el tratamiento de las configuraciones de un SMA:

- *Propiedades patrón*. Se trata de configuraciones que se desea preservar en el sistema. Se considera que existe un problema en las especificaciones con una *propiedad patrón* cuando éstas no la satisfacen o lo hacen de una manera parcial. Ejemplos de *propiedades patrón* serían los requisitos del usuario que han de mantenerse a lo largo del desarrollo hasta el sistema final.
- *Propiedades anti-patrón*. Definen configuraciones a evitar en el sistema. En este caso existe un problema en las especificaciones cuando una porción de ellas satisface la *propiedad anti-patrón*. Dentro de esta categoría se considerarían las contradicciones que la TA describe para los sistemas sociales.

- *Propiedades descriptivas*. Son configuraciones que se describen con el propósito de facilitar el descubrimiento o incorporación de características en los sistemas. Éste sería el caso de las propiedades que representan organizaciones sociales prototípicas que puede adoptar un SMA, tales como una jerarquía, un mercado o una comunidad de expertos.

A fin de facilitar su posterior tratamiento (e.g. detección de las propiedades y modificación de los modelos según éstas), se quiere dar una representación uniforme a los tres tipos de *propiedades sociales*. Además, la representación elegida ha de satisfacer tres objetivos esenciales para el desarrollo de SMAs. En primer lugar, debe preservar la utilidad de la información que representa como vehículo comunicativo compartido entre desarrolladores y clientes. Ésta ha sido una de las ventajas perseguidas con la introducción de la TA y sus abstracciones sociales. También debe facilitar el uso de las propiedades en el proceso de desarrollo con la notación UML introducida y con mecanismos que faciliten la escalabilidad en las definiciones (es decir, la descripción de propiedades complejas partiendo de otras más simples). Finalmente, la representación debe soportar la incorporación de información relativa a cómo resolver los problemas relacionados con las propiedades, tales como la disconformidad con los requisitos o la aparición de contradicciones en las especificaciones.

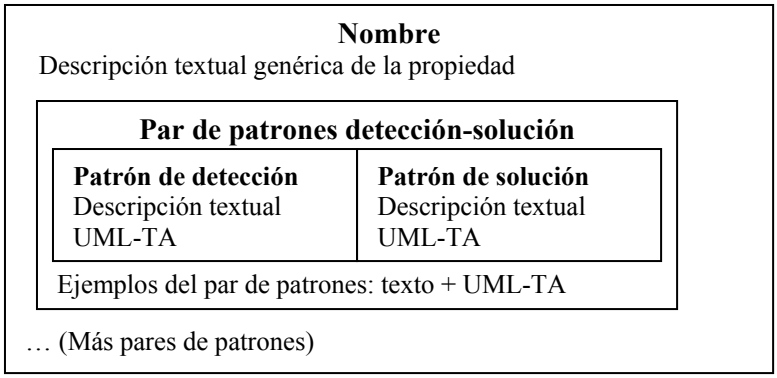


Fig. 23. Representación de las *propiedades sociales* de la TA.

Para cumplir los propósitos previos, las *propiedades sociales* se describen con estructuras como la reflejada en la Fig. 23. Cada estructura incluye el nombre de la propiedad que representa y una descripción textual genérica de ella en términos del vocabulario de la TA. Esta descripción ayuda a clientes y desarrolladores a comprender la configuración que representa la propiedad. Además, las estructuras incluyen un conjunto de pares de patrones, donde cada par está constituido por un patrón de detección y otro de solución. Los patrones incluyen una descripción UML-TA que actúa como un marco cuyas ranuras se rellenan con la información obtenida en los procesos de tratamiento de las *propiedades sociales*. La estructura contempla conjuntos de pares de patrones porque se considera que la misma propiedad puede admitir diferentes representaciones, que corresponden a distintas interpretaciones de la fuente textual original.

Como ya se ha indicado, los pares de patrones comprenden un patrón de detección y otro de solución. Estos patrones tienen diferentes misiones en el tratamiento de las propiedades.

Un patrón de detección describe un conjunto de entidades y relaciones que representan la propiedad. Este tipo de patrón es un marco a instanciar con información de la especificación. Si un conjunto de artefactos de la especificación encaja en este patrón se dice que la propiedad se satisface. Para ciertas propiedades también es posible contemplar correspondencias parciales que señalarían, por ejemplo, información que falta en las especificaciones.

En cuanto al patrón de solución, se trata de un componente opcional en los pares. Así, las propiedades orientadas a capturar información pueden no incluirlo. Tanto en el caso de propiedades a mantener como a evitar, el patrón de solución es una reorganización de los elementos del patrón de detección que tiene asociado, quizás con componentes adicionales. Los patrones de solución pueden referirse tanto a correspondencias totales como parciales de los patrones de detección. En el caso de una ausencia total de correspondencia no tiene sentido definir un patrón de solución.

Hay que señalar que los patrones de detección y solución no están asociados en pares fijos. Por el contrario pueden ser reutilizados o combinados para describir nuevas situaciones.

Cada patrón en la definición de una *propiedad social* tiene dos representaciones, una textual y otra con UML-TA. La forma UML se usa para los procesos semi-automáticos de captura de información y comprobación de *propiedades sociales*. En ellos actúa como un marco para recoger información. Los identificadores presentes en sus ranuras pueden ser tanto valores fijos como variables. Esto permite, por ejemplo, prefijar algunos valores de los patrones antes del proceso de detección o combinar patrones a través de valores compartidos. Por otro lado, la forma textual está orientada a dar información adicional acerca de los patrones. Esta forma ayuda a los clientes a comprender el significado de la notación UML usada y permite que tanto los clientes como los desarrolladores conozcan la interpretación social del patrón.

Para ilustrar la clase de descripciones que se están considerando se introduce en esta sección un ejemplo de *propiedad anti-patrón*. Se trata de la contradicción de *Doble Vínculo* en la TA, que será tratada con mayor profundidad en el Capítulo 5 sobre el proceso de diseño guiado por contradicciones. La descripción textual de los patrones se introduce aquí como parte de la explicación de los mismos.

El *Doble Vínculo* representa una situación en la que todas las *actividades* que puede realizar el *sujeto* afectan negativamente a alguno de sus *objetivos*. Puesto que el *sujeto* sigue el Principio de Racionalidad [Newell 1982], ha de elegir una *actividad* que le ayude a alcanzar alguno de sus *objetivos*. Sin embargo, esa *actividad* perjudica al mismo tiempo a otros de sus *objetivos*. Por tanto, haga lo que haga el *sujeto* no puede satisfacer sus necesidades. El patrón de detección que recoge esta información aparece en la Fig. 24.

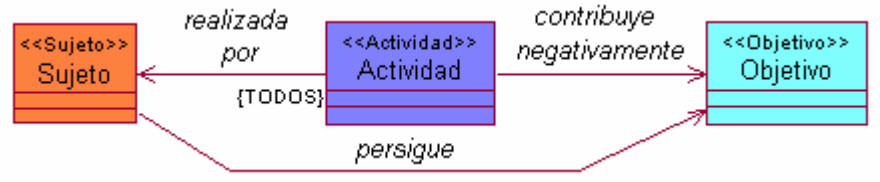


Fig. 24. Ejemplo de patrón de detección de una *propiedad social* con el *Doble Vínculo*.

En el momento de utilizar el patrón de la Fig. 24 para detectar una propiedad, algunas de las etiquetas "Sujeto", "Actividad" y "Objetivo" podrían ser reemplazadas por valores concretos, i.e. nombres de entidades de las especificaciones del SMA. De esta forma se puede comprobar la propiedad sólo para ciertos elementos de las especificaciones. Para aquellas etiquetas que se dejasen como variables se consideraría que se pretenden recuperar todas las posibles soluciones en las especificaciones.

En cuanto al patrón de solución asociado al *Doble Vínculo*, esta contradicción puede resolverse descomponiendo la *actividad* original en otras *actividades* que reduzcan sus efectos negativos aunque satisfaciendo sus *objetivos*, tal y como se muestra en la Fig. 25. El patrón de solución conserva en este caso todos los elementos de su patrón de detección. La solución al problema queda reflejada con la introducción de las nuevas *actividades* que descomponen a la original. Estos nuevos elementos se enmarcan en una elipse de trazo discontinuo.

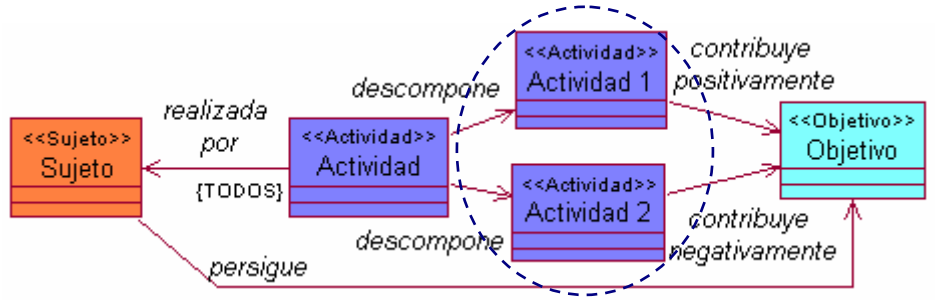


Fig. 25. Ejemplo de patrón de solución de una *propiedad social* con el *Doble Vínculo*.

El patrón de solución del *Doble Vínculo* en la Fig. 25 propone una posible forma de mitigar la contradicción. En la comprobación de las propiedades, esta solución se instancia haciendo uso del patrón de detección y su correspondencia con las especificaciones. De este modo, la solución se puede expresar en términos de las abstracciones usadas por los propios clientes y desarrolladores en sus modelos.

Aunque este apartado ha introducido el mecanismo de representación con patrones a propósito de las *propiedades sociales* relacionadas con la TA, se trata de una herramienta genérica. Parte de la información presente habitualmente en las comprobaciones de SMAs, tales como que una tarea ha de tener asociado un objetivo o que los agentes han de ser capaces de cumplir sus objetivos, puede ser representada declarativamente con estas estructuras y ser comprobada de la misma forma que las *propiedades sociales*.

3.7. Método de comprobación de *propiedades sociales*

Los patrones anteriores se usan en un proceso de detección y resolución de propiedades sociales en modelos de SMAs (ver Fig. 26). Este proceso es el núcleo de los métodos de uso de la TA en la ISOA que se verán en los 0 (captura de requisitos) y Capítulo 5 (evolución de las especificaciones con contradicciones).

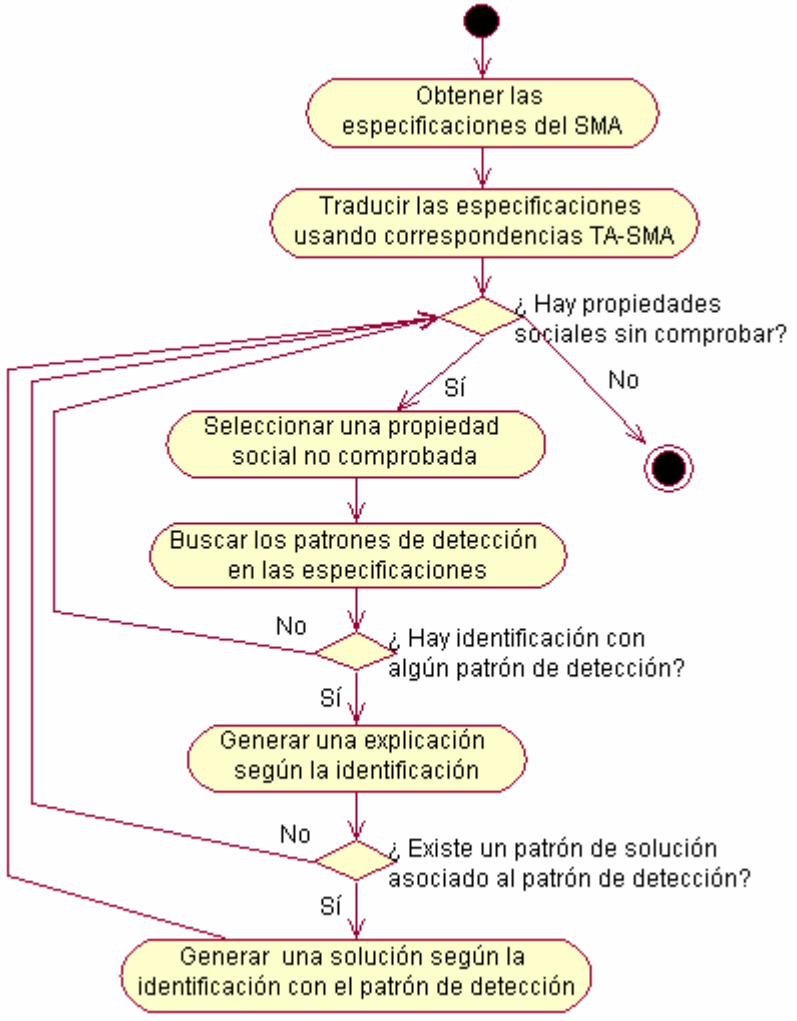


Fig. 26. Diagrama de actividades del método de comprobación de *propiedades sociales*.

El proceso de comprobación de *propiedades sociales* necesita tres grupos de datos:

- *Especificaciones del SMA*. Descripción de un SMA para el que se desean comprobar *propiedades sociales*.
- *Correspondencias*. Se trata de las asociaciones para la traducción entre el vocabulario de la TA y el de la metodología del SMA a comprobar. Estas correspondencias han sido introducidas en la sección 3.5 de este mismo capítulo.
- *Propiedades sociales*. Es el conjunto de configuraciones cuya presencia se desea comprobar en las especificaciones. Estas propiedades son descritas con las estructuras vistas en la sección 3.6.

El método de comprobación incluye las siguientes etapas:

1. *Obtener las especificaciones del SMA*. Se trata de los modelos que describen el SMA a comprobar.
2. *Traducir las especificaciones del SMA al lenguaje de la TA*. Las correspondencias describen cómo las estructuras definidas en un lenguaje pueden ser escritas en el otro. Como ya se ha visto, la traducción no es una tarea trivial ya que las correspondencias entre las estructuras y sus traducciones no son biunívocas. En general, las correspondencias reflejan correspondencias "muchos a muchos" entre estructuras. Como consecuencia, al aplicar el método es preciso mantener referencias desde las estructuras traducidas a las originales para recordar la traducción seleccionada.
3. *Para cada propiedad social, buscar instancias de sus patrones de detección en las especificaciones*. El proceso de detección de las propiedades en los modelos es de detección de patrones ("*pattern matching*"). Los patrones son marcos que describen conjuntos de entidades y relaciones. Sus ranuras pueden estar rellenas con variables o valores fijos. La detección de propiedades consiste en recorrer los modelos traducidos buscando grupos con la misma estructura y valores en las ranuras que los de los patrones de detección. En esta búsqueda se pueden emplear las equivalencias entre conceptos de UML-TA vistas en la sección 3.4.2. Cuando se produce una correspondencia se considera que la propiedad se satisface. Las correspondencias parciales también son posibles y pueden representar situaciones conflictivas, por ejemplo un requisito que no es preservado.
4. *Cuando se produce una detección, proponer el patrón de solución adaptado como alternativa*. Los patrones de detección tienen asociados patrones de solución que representan propuestas para resolver problemas o mejorar las especificaciones. Una vez que se ha establecido la correspondencia con el problema, el patrón de solución se expresa mediante la reestructuración y refinamiento de los elementos en una nueva configuración del modelo.

Los pasos de detección y solución del proceso anterior pueden hacer uso de las definiciones del lenguaje UML-TA y de las equivalencias que se definan. Las entidades de los patrones no tienen que aparecer por tanto literalmente en las especificaciones, sino que pueden aparecer a través de transformaciones dadas por las definiciones del lenguaje. Por ejemplo, si en un patrón una *actividad* satisface un *objetivo* pero en las especificaciones satisface dos, puede usarse la relación *descompone* y considerar que hay un *objetivo* de la *actividad* que se descompone en los dos de las especificaciones.

Para terminar la presentación del método, remarcamos algunas de sus características:

- *Genericidad*. El método propuesto es genérico en el sentido de que no se exige que el SMA esté descrito en términos de las abstracciones propias de la TA. Sólo se requiere que el marco conceptual de la metodología esté fundamentado en las abstracciones propias del paradigma de agentes.
- *Semi-automático*. Este método permite a los clientes y a los desarrolladores comprobar sus modelos con *propiedades sociales* de una forma prácticamente automática, como se verá en el Capítulo 6 dedicado a la implementación. La interacción con el usuario es necesaria para determinar los valores de las variables en los patrones, decidir si la detección en las especificaciones tiene sentido en el problema real o la mejor manera de modificar los modelos.
- *Polivalente*. El método presentado puede considerarse bajo diferentes puntos de vista en función de las *propiedades sociales* seleccionadas. Si estas propiedades son información que se pretende introducir en el SMA, se trata de un método de Ingeniería de Requisitos. Si se trata de requisitos del sistema ya introducidos cuya presencia se desea comprobar, el método puede ser descrito como de "validación". Cuando estas propiedades representan otro tipo de información, el método puede considerarse como "verificación".

Para concluir, señalar que el proceso de comprobación presentado en esta sección no está relacionado con la técnica formal conocida como "comprobación de modelos" ("*model checking*") [Bérard *et al.* 2001]. La "comprobación de modelos" parte de la representación del sistema a analizar como una máquina de estados [Gurevich 1984] (e.g. estructuras de Kripke) mientras que las propiedades que se desean comprobar se expresan típicamente como fórmulas de lógica temporal. El sistema satisface la propiedad si la correspondiente fórmula es verdadera en el modelo del sistema, es decir, en la máquina de estados. El algoritmo propuesto en este trabajo se basa sin embargo en la correspondencia entre patrones estructurales. Tanto el sistema como la propiedad a comprobar se representan mediante grafos. La comprobación consiste en demostrar que el grafo de la propiedad es un subgrafo del grafo que representa al sistema.

3.8. Conclusiones

Este capítulo ha presentado la Teoría de Actividad en sus campos de investigación de origen, la Sociología y la Psicología. En su descripción se han destacado varios aspectos relevantes para el análisis de SMAs.

En primer lugar, la TA concibe todas las *actividades* que ocurren en una organización como sociales. La forma de actuar de los *sujetos* depende de su contexto e influye a su vez en éste. Por tanto, se propugna un análisis sistémico de las organizaciones, ya que se mantiene que no basta con analizar sus partes aisladamente. Esta clase de estudio entronca con los propuestos para los SMAs por diversos autores,

que también establecen la necesidad de contemplar los sistemas como una organización y no como una mera agregación de elementos.

En segundo lugar, se subraya el carácter intencional de las *actividades*. Los *sujetos* se embarcan en la realización de *actividades* porque han de cumplir sus *objetivos*, es decir, satisfacer sus necesidades. Esta motivación de los sujetos se sitúa en la misma línea que la de los agentes inteligentes, que han de comportarse de acuerdo con el Principio de Racionalidad de Newell.

Finalmente, en la TA se contempla la dimensión temporal y evolutiva de las actividades. Las *actividades* no sólo se realizan a lo largo del tiempo sino que también evolucionan. Aspectos como las contradicciones o el aprendizaje de los sujetos sólo se conciben analizando las *actividades* a lo largo de un eje temporal. También en los SMAs se incluye esta transformación del sistema en el tiempo como un elemento necesario del análisis. Además, el estudio del sistema evolucionando en el tiempo como consecuencia de sus contradicciones también responde al proceso de desarrollo. Los artefactos del proceso surgen y cambian a lo largo de las iteraciones para cubrir nuevas necesidades, tales como la especificación de ciertos aspectos o incluir nueva información sobre los requisitos del cliente.

Las conclusiones extraídas de la exposición anterior han sido que la TA resulta un marco social adecuado para tratar las *propiedades sociales* en la ISOA. La TA tiene conceptos y técnicas aplicables al estudio de SMAs y su análisis social, espacial y temporal de la *actividad* cubre aspectos de interés tanto del SMA como del proceso software que lo desarrolla.

Tras elegir la TA para trabajar con la ISOA, el siguiente paso ha consistido en crear la infraestructura para utilizar el conocimiento de la TA en un proceso software para SMAs. Esta infraestructura consiste en un lenguaje basado en UML para los conceptos de la TA, las correspondencias TA-SMA, la estructura para representar *propiedades sociales* y un método de comprobación que utiliza estos elementos.

El primer problema en la aplicación de la TA es que todo su conocimiento está descrito en lenguaje natural. A fin de facilitar su uso en el contexto de un desarrollo software, sus principales conceptos han sido especificados con UML, creando el lenguaje UML-TA. Esta especificación sitúa las abstracciones de la TA en un lenguaje de diseño, que puede ser manejado más fácilmente por los desarrolladores y al cual se le pueden asociar herramientas de desarrollo propias de la Ingeniería del Software. No obstante, esta adaptación no está libre de inconvenientes. UML es un lenguaje menos rico que el natural, por lo que se pierden matices de significado en la traducción de las *propiedades sociales* a UML-TA. Además, se puede afirmar que UML-TA resulta menos expresivo que algunos de los lenguajes de diseño manejados en la ISOA (ver Capítulo 2), aunque esa riqueza expresiva se ha sacrificado en aras de una mayor facilidad de uso y aprendizaje. Por último, el proceso de traducción a UML-TA está necesariamente sujeto a la interpretación de las fuentes en lenguaje natural.

En cuanto a la traducción entre la TA y las metodologías de SMAs, se basa en correspondencias. Las correspondencias son relaciones entre estructuras genéricas que determinan la forma de reescribir los modelos descritos en un lenguaje en otro. Estas correspondencias se fundamentan en las características sociales e intencionales comunes a los sistemas analizados por la TA y las metodologías de la ISOA. Las traducciones basadas en estas correspondencias no son únicas. Debido a las

diferencias entre lenguajes, las correspondencias reflejan frecuentemente asociaciones entre conceptos "muchos a muchos", de manera que una estructura en cualquiera de los lenguajes puede corresponder a varias estructuras en el otro lenguaje.

El tercer elemento creado para manejar la información de la TA es la estructura para describir *propiedades sociales*. Ésta comprende dos grupos de patrones, los patrones de detección y los de solución. Los primeros describen la propiedad buscada y los segundos mejoras en las especificaciones en base a la identificación de los patrones de detección. Para convertir las *propiedades sociales* en una herramienta útil a clientes y desarrolladores, cada patrón cuenta con una representación mixta. Por un lado el patrón se describe en lenguaje natural a fin de facilitar la comprensión de su significado social. Por otro lado el patrón cuenta con una descripción en UML-TA que da una interpretación menos ambigua que la forma textual y sirve como marco para introducir la información necesaria en los procesos de desarrollo.

El uso de patrones estructurales incluye variables en su descripción. Esto permite instanciar patrones para comprobaciones de elementos específicos, combinar patrones para describir situaciones complejas y abordar la construcción de contradicciones modularmente.

Por último, los elementos anteriores son utilizados en un proceso de comprobación de *propiedades sociales*. El método traduce las especificaciones de un SMA usando las correspondencias asociadas a su metodología. Luego se buscan identificaciones de los patrones de detección de las *propiedades sociales* con porciones de las especificaciones traducidas. Cuando de detecta una correspondencia, los patrones de solución se instancian con ella para proponer mejoras en las especificaciones.

Como se ha indicado anteriormente, los mecanismos de este capítulo son la base para usar la TA en el desarrollo de SMAs. Fruto del análisis de los estudios y técnicas de la TA, los dos capítulos siguientes presentan herramientas para la captura de requisitos y la dirección y verificación del proceso de desarrollo en función de las contradicciones presentes en sus artefactos.

Capítulo 4. TA y la captura de requisitos

Uno de los aspectos de este trabajo es la aplicación de la TA en la ISOA para la ayuda a la obtención de información sobre las propiedades sociales. Este capítulo plantea la adaptación de la Guía de Actividades de la TA para este propósito. La Guía de Actividades indica cuáles son los aspectos de interés acerca de una actividad mediante preguntas en lenguaje natural. Este capítulo comienza con la presentación de la formulación original de la Guía de Actividades en estudios de HCI. Luego se plantea cuáles son las modificaciones a introducir para su uso en la ISOA. El resultado de estas modificaciones es la Guía de Captura de Requisitos, concebida como una herramienta para guiar la adquisición de requisitos sociales de un SMA y su descripción con el lenguaje de diseño UML-TA. Las restantes secciones presentan el modo de uso y formulación de esta nueva guía, así como un ejemplo de aplicación. El capítulo se cierra con una discusión sobre las características de la Guía de Captura de Requisitos.

4.1. Introducción

En Ingeniería del Software se suele afirmar que:

La parte más difícil de construir un sistema software es decidir precisamente lo que construir... Por tanto, la función más importante que el desarrollador del software realiza para el cliente es la extracción y refinamiento iterativos de los requisitos del producto [Brooks 1987].

La obtención de los requisitos de un sistema software es considerada como una actividad crucial para el éxito de un desarrollo y, a la vez, es también una de las más difíciles de llevar a cabo [Brooks 1975]. La construcción de SMAs no es una excepción a esta afirmación. En tanto que se trata de sistemas software, su desarrollo está sujeto a la identificación de los requisitos al igual que en la Ingeniería del Software convencional.

Uno de los motivos de estas dificultades para obtener los requisitos es que estos han de contemplar las mutuas influencias entre el sistema a construir y el entorno en el que se situará. Este entorno, definido como el mundo real externo al sistema software, suele ser un sistema de actividad humano [Nuseibeh & Easterbrook 2000]. La interacción entre el sistema y las personas hace que sea necesario tener en cuenta el modelado del componente humano al abordar los requisitos. Los modelos empleados para esta tarea han de considerar al individuo en el marco de actividades humanas que responden a patrones regulares [Probert 1999].

Otra de las dificultades viene dada por la necesidad de realizar la captura de requisitos a lo largo de todo el ciclo de vida de un sistema: a medida que avanza el ciclo, los requisitos de los clientes son contextualizados por la arquitectura existente del sistema [Zave 1999, Nuseibeh & Easterbrook 2000]. La ISOA contempla sobre todo metodologías de carácter iterativo e incremental. Éste es caso de las inspiradas, por ejemplo, en el USDP [Jacobson *et al.* 1999] o en los modelos Caos [Raccoon 1995]. En ellas, cada iteración del proceso contempla todas las actividades incluidas en la metodología. Dependiendo de la etapa del desarrollo, se presta mayor atención a unas u otras actividades. El resultado de cada iteración son unas especificaciones modificadas del sistema que constituyen el punto de partida de la siguiente iteración. Con esta clase de procesos, el análisis de los requisitos se realiza en todas las iteraciones, aunque con mayor relevancia en las iniciales. A medida que avanza el desarrollo, los nuevos requisitos están más contextualizados por las decisiones tomadas previamente acerca de la arquitectura del SMA. Considerar este contexto impuesto por el SMA requiere que los desarrolladores traten de manera integrada los requisitos del sistema que tienen su origen en el entorno y la información de alto nivel sobre el SMA.

Por tanto, al capturar los requisitos de un SMA, los desarrolladores han de combinar la descripción del contexto humano del sistema con ciertas características arquitectónicas del mismo. Esto conduce en la ISOA a considerar los lenguajes orientados a agentes para realizar la captura de requisitos. Gracias a que la metáfora natural al tratar los SMAs es la de organizaciones de componentes intencionales [Jennings & Wooldridge 2000], el paradigma de agentes ofrece la posibilidad de modelar de forma integrada ciertos aspectos del entono humano y de los sistemas software. Esta propuesta facilita tratar con el problema planteado para la captura de requisitos.

No obstante, el uso de un lenguaje basado en agentes no resuelve completamente el problema de tratar con los requisitos ya que determinar cuál es la información que se debe capturar y cómo modelarla sigue sin ser una tarea trivial. Los desarrolladores siguen necesitando decidir qué información del sistema y su contexto será relevante para el desarrollo y cuál es la mejor forma de obtenerla y plasmarla en lenguajes de diseño de software. Además, usar estos lenguajes para la obtención de información suscita la necesidad de nuevos métodos de trabajo, ya que el lenguaje empleado en la captura y modelado influye en la forma en la que se conciben y obtienen los requisitos [Lamsweerde 2000].

A fin de proporcionar un soporte teórico, conceptual y metodológico a la captura de requisitos en la Ingeniería del Software, centrado en la comprensión de los aspectos relacionados con las personas y sus organizaciones, resulta natural recurrir a las ciencias sociales [Goguen & Linde 1993]. Como se vio anteriormente, el uso de técnicas procedentes de estas disciplinas se ve facilitado en la ISOA por las características comunes entre las entidades que modelan y las del paradigma de agentes.

Acerca de las técnicas a considerar de las disciplinas sociales, es necesario que se trate de metodologías y herramientas que puedan soportar la evaluación continua de un conjunto de requisitos que evolucionan en una situación altamente compleja y dinámica. Se tratará de métodos de estudio de contextos humanos reales que contemplan su dimensión evolutiva. Además hay que tener en cuenta que buena parte

de la información relativa al componente humano no tiene una realidad tangible, observable por un desarrollador, y no es objetiva, al contrario que la información acerca del mundo físico. En su lugar esta información se basa en las necesidades y apreciaciones subjetivas de individuos, grupos u organizaciones. Los métodos elegidos contemplarán esta componente subjetiva, tanto individual como social.

La investigación en TA ha empleado numerosas técnicas para obtener esta información acerca de los entornos humanos. Algunos ejemplos se encuentran en los estudios etnográficos recogidos en [Engeström & Middleton 1998], las técnicas de análisis cognitivo aplicado para tareas [Militello & Hutton 1998], el diseño participativo [Bødker *et al.* 1998], el método precursor-acción-resultado-interpretación [Hall *et al.* 1995], el diseño contextual [Holtzblatt & Beyer 1993] y la guía de actividades [Kaptelinin *et al.* 1999]. Todas ellas han arrojado resultados relevantes sobre el análisis del entorno de aplicaciones. Así, estas técnicas han sido útiles para comprender aspectos como la motivación de los usuarios, necesidades comunicativas o reticencias en la organización. Sin embargo, su aplicación en proyectos software concretos ha requerido considerables inversiones en tiempo y recursos. Esta clase de estudios suelen requerir largos periodos de tiempo y expertos en ciencias sociales para su realización. Por otra parte, casi todos los experimentos reportados fueron conducidos por científicos procedentes de la Sociología, Psicología o Etnografía, que fueron ayudados por equipos de desarrollo. Esto se tradujo en una falta de orientación real hacia la construcción de software, el campo en que estas técnicas prestarían apoyo. El resultado es que la información de la mayor parte de estos estudios es bastante inaccesible a los desarrolladores porque está orientada a otra clase de expertos. Además son técnicas difícilmente utilizables en proyectos software que no involucren a equipos pluridisciplinares (desarrolladores y expertos en ciencias sociales) y contemplen largos plazos de tiempo (el necesario para completar estos estudios).

El trabajo de esta tesis adopta una vía diferente para aplicar la TA en la captura de requisitos de SMAs. El esfuerzo del presente trabajo se centra en técnicas que puedan ser usadas por los desarrolladores sin suponer una carga prohibitiva de recursos o tiempo para el proyecto. Para satisfacer estos requisitos se ha tratado de proporcionar procesos con guías regladas, que no exijan conocimientos expertos en ciencias sociales, suficientemente próximos a la Ingeniería del Software y que sean al menos parcialmente automatizables. En esta línea, la propuesta para la captura de requisitos parte de la Guía de Actividades como herramienta procedente de la TA.

La Guía de Actividades ("*Activity Checklist*") (GA) [Kaptelinin *et al.* 1999] se usa en la investigación sobre la interacción hombre-máquina (*HCI*), para analizar el impacto sobre las actividades humanas de la introducción de una nueva *tecnología*. El término *tecnología* se entiende en un sentido amplio, pudiendo tratarse de dispositivos, procesos o aplicaciones. Las ventajas que presenta esta guía sobre otros métodos, desde el punto de vista de su aplicación en un proceso software, son:

- *Fundamento teórico en la TA*. La GA emerge como una ayuda para la aplicación de la TA. Se trata de un recordatorio de los puntos que deben ser abordados a la hora de analizar una *actividad*. Por ello, todo su contenido y proceso está fundamentado directamente en los conceptos de la propia TA.
- *Método de aplicación estructurado*. La GA se define mediante una estructura organizada en *áreas*, *aspectos* y *preguntas*. Las *áreas* están relacionadas con los

focos de interés en una *actividad* según la TA. Cada *área* comprende diferentes vistas, los *aspectos*, de una *actividad* social: pasado, estado actual, evolución, aprendizaje... El conocimiento acerca de estos *aspectos* describe el entorno tal y como lo perciben los usuarios. Para recoger la información acerca de los *aspectos*, la GA define *preguntas* relacionadas con estas características. Las *preguntas* pretenden ser recordatorios acerca de la información que se ha de considerar. Esta estructura de la GA suministra una guía ordenada de la información que se ha de solicitar a los usuarios en relación con la *actividad* a analizar, aunque no establece una traducción directa de esa información a los conceptos de la TA.

- *Orientada al cliente*. La GA está descrita completamente en lenguaje natural. De este modo se facilita su utilización sin necesidad de conocer notaciones específicas, pero a costa de dificultar su traducción a lenguajes de diseño o el uso de herramientas de modelado software.

Usando la GA original como base, se ha elaborado una versión de la misma centrada en la captura de requisitos para SMAs y que emplea la representación ya introducida para *propiedades sociales* (ver sección 3.6). Es la Guía de Captura de Requisitos (GCR). Con la GCR se proporciona un método fundamentado en teorías sociológicas para obtener los requisitos, que además puede conectarse con otros procesos basados en la TA para comprobar propiedades. La GCR no trata de ser una solución universal para todos los problemas de captura de requisitos en el desarrollo de SMAs. Representa una forma de trabajar inspirada en la TA, que debe ser usada en combinación con otras técnicas de Ingeniería de Requisitos y que sirve como guía de aquellos puntos de interés acerca de un SMA.

En las siguientes secciones de este capítulo se hace una introducción a la GA tal y como fue propuesta originalmente en la HCI. A continuación se indican las principales modificaciones que se han introducido en la GA para aplicarla en SMAs, se describe un proceso para llevar a cabo esta modificación y se analiza la casuística de *preguntas* que pueden aparecer en la adaptación. Los resultados de este estudio se plasman en la GCR, la versión adaptada a la ISOA de la GA. Por último, el uso de la GCR se ilustra con un caso de estudio. El capítulo termina con la discusión sobre la propuesta de captura de requisitos con la GCR de acuerdo con los métodos propuestos y los resultados del ejemplo.

4.2. Guía de Actividades

La Guía de Actividades (GA) [Kaptelinin *et al.* 1999] es una herramienta analítica aplicada principalmente en el campo de la HCI. La GA orienta el estudio, en términos de los conceptos de la TA, del impacto de la introducción de nueva tecnología y la interacción con ella en las actividades humanas. El término *tecnología* hace referencia a los *artefactos* de un *sistema de actividad* y puede corresponder tanto a un sistema software, como a una herramienta, un proceso o la tecnología de agentes. Ejemplos de uso de la GA para el diseño de software pueden encontrarse en [Gould *et al.* 2000, Hedestig & Kaptelinin 2002]. En [Gould *et al.* 2000] la GA se usa para determinar las

necesidades de interacción usuario-aplicación en un sistema de información para la web. En [Hedestig & Kaptelinin 2002] se emplea para estudiar la implementación de un sistema para la enseñanza a distancia y cómo su implantación debe modificar la forma en la que se realiza el aprendizaje.

El origen de la GA se encuentra en la necesidad de concretar la TA para su aplicación. Como se indicó anteriormente, la TA es un marco conceptual abstracto que no puede ser aplicado directamente a un problema concreto. La TA sólo proporciona las abstracciones necesarias para que los investigadores creen sus propias herramientas acordes con su problemática particular. Con esta idea, la GA surge para ayudar a construir el contexto para la *actividad* del sistema con una aproximación "*top-down*". El término "*top-down*" hace referencia a que el proceso de captura de la información trata de concretar los conceptos de la base teórica que representa la TA en el contexto de una *actividad* especifica. Para lograr este propósito, la GA ha de cubrir un amplio espectro de introspección: comienza examinando el espacio completo de características de interés acerca de la *actividad*, para centrarse después en las características identificadas y examinarlas con profundidad. Este proceso da lugar a una estructura jerárquica que surge de los cinco principios básicos de la TA: orientación al *objeto*; estructura jerárquica de la *actividad* con la diferenciación entre *actividades*, *acciones* y *operaciones*; internalización/externalización; el papel de la mediación; y el desarrollo. Estos principios ya han sido introducidos previamente en la sección 3.2 referida al contexto de la TA. Estas pautas se plasman en [Kaptelinin *et al.* 1999] en la GA para la HCI.

En el campo de la HCI, la GA maneja dos versiones distintas: una para el estudio de sistemas ya en uso y otra para ayudar en el diseño de sistemas a construir. En ambos casos la GA define un conjunto de *áreas* adyacentes a los conceptos de la TA. Cada *área* incluye una descripción de su significado, un conjunto de *aspectos* que representan la información que se considera en ella y un conjunto de *preguntas* acerca de esos *aspectos*. Son estos *aspectos* y *preguntas* los que permiten obtener la información para las *áreas* y concretar los conceptos de la TA a un caso de estudio dado. Algunos ejemplos de *áreas*, *aspectos* y *preguntas* procedentes de la versión original de la GA para sistemas en uso pueden ser vistos en la Tabla 11.

Las cuatro *áreas* cubiertas por la GA son:

- *Medios y fines*. Las relaciones entre los objetivos del usuario y el sistema. Analiza la extensión en que la tecnología facilita y restringe el logro de los objetivos de los usuarios. También considera la influencia de la tecnología en las contradicciones de la organización, es decir, su impacto en promover y resolver conflictos entre diferentes artefactos (por ejemplo entre *objetivos*, en la *comunidad* o de los *objetivos* con las *herramientas*).
- *Aspectos físicos y sociales del entorno*. Integración de la tecnología con los requisitos, *herramientas*, recursos, *división del trabajo* y *reglas* sociales del entorno.
- *Aprendizaje, cognición y articulación*. Esta área considera cómo la tecnología objetivo proporciona conocimiento para soportar la organización de la *actividad* y la adquisición del conocimiento acerca de ella y del sistema. También contrasta los componentes internos de la *actividad* contra los externos y considera el soporte para sus mutuas transformaciones con la tecnología objetivo.

- *Desarrollo*. Cómo el uso de la tecnología ha cambiado la forma previa de la *actividad* y la evolución prevista de la forma actual.

Para descubrir el contexto involucrado en estas *áreas* de la HCI, la GA propone prestar atención a varias vistas de ellas, los *aspectos*. En cada *aspecto*, la GA tiene *preguntas* que permiten descubrir el conocimiento relacionado con él. Estas *preguntas* están concebidas para recordar a los investigadores los puntos esenciales a considerar en el análisis.

VERSIÓN DE USO			
	Áreas	**Aspectos**	**Ejemplos de preguntas**
USO	Medios/ fines	Formas alternativas de lograr los objetivos a través de otros objetivos de nivel más bajo.	¿Es necesario que el usuario cambie constantemente entre diferentes acciones o actividades en su trabajo? Si la respuesta es afirmativa, ¿existen "atajos" que le permitan una transición fácil entre acciones y actividades y, si es necesario, volver a estados, acciones o actividades previos?
	Entorno	Rol de la tecnología existente en la generación de los productos de las acciones objetivo.	¿La tecnología considerada está integrada con otras herramientas y materiales?
	Aprendizaje/ cognición/ articulación	Observación y reflexión a través de la externalización.	¿El conocimiento distribuido externamente está fácilmente accesible cuando se necesita?
		Posibilidades de simulación de las acciones objetivo antes de su implementación real.	¿Proporciona el sistema representaciones de las actividades de los usuarios que puedan ayudarles a fijar sus objetivos y a la auto-evaluación?
	Desarrollo	Cambios anticipados en las acciones objetivo después de que se implemente la nueva tecnología.	¿Las actitudes de los usuarios hacia el sistema se están volviendo más o menos positivas?
			¿Existen efectos colaterales positivos o negativos fruto del uso del sistema?

Tabla 11. Ejemplos de *áreas*, *aspectos* y *preguntas* extraídos de la versión de uso de la Guía de Actividades.

En general, tanto la formulación como los contenidos de la GA pueden ser comprendidos sin necesidad de un conocimiento profundo de la TA. No obstante, los investigadores que emplean la GA han de hacer refinamientos en sus preguntas cuando se las plantean a los usuarios. Se trata de formularlas sin incluir conceptos específicos ni la de la Ingeniería del Software ni de la TA. Las respuestas de los

usuarios son recogidas por los expertos. Estos expertos son los encargados de procesar la información para componer la descripción textual del contexto de la *actividad* en términos de los conceptos de la TA.

Como conclusión, la GA ha sido utilizada en estudios de HCI. Para SMAs será necesario redefinirla por varios motivos:

- *Sesgada a la HCI*. La GA se centra mucho en características que no son primordiales para los SMAs generales, tales como la adaptación de los usuarios al sistema o los estudios etnográficos acerca de las actividades relacionadas.
- *Requiere conocimiento experto en TA*. Especialmente para procesar las respuestas de los usuarios.
- *Falta de detalle*. La GA está lejos de proporcionar el nivel de detalle necesario acerca del sistema estudiado. Se centra sobre todo en aspectos de interacción mientras que incide mucho menos en la funcionalidad u organización del sistema.

En resumen, la GA ayuda a extraer de los usuarios información relevante, pero entender su contenido y trasladar sus respuestas a un lenguaje de diseño o modificarla, no es una tarea trivial sin la base adecuada.

4.3. Adaptando la GA para la ISOA

La GA en su propuesta original [Kaptelinin *et al.* 1999] no es directamente aplicable a las metodologías de SMAs. Su foco de atención se encuentra en la interacción de las personas con los sistemas, incidiendo en unos puntos de interés diferentes de los del desarrollo basado en agentes. Además, la GA está concebida como una guía para aquellos investigadores interesados en analizar una *actividad* con la TA, no para usuarios sin conocimientos básicos de la TA. Sin embargo, no se trata de un cuestionario cerrado, sino sujeto a permanente revisión en su uso y que puede ser alterado al combinarse con otras técnicas de análisis.

A fin de usar la GA con metodologías de agentes, los siguientes apartados analizaran qué características debe cumplir la guía para ser usada como una herramienta de la ISOA, cómo redefinirla para satisfacer estas expectativas y los casos que se pueden presentar al realizar estos cambios.

4.3.1. Modificaciones a introducir

Para aplicar la GA al desarrollo de SMAs e integrarla en un proceso software soportado por la TA se tienen que cumplir varios objetivos. El primero se refiere a los cambios en el contenido de la propia GA, el segundo se relaciona con el modo en el que la GA se integra con el resto de las fases del proceso y, finalmente, el tercero está orientado a facilitar su uso.

Para satisfacer el primer objetivo, la GA tiene que ser transformada en dos direcciones ortogonales, su foco y su nivel de detalle.

Acerca de su centro de atención, la GA tiene que cambiarlo desde los aspectos relacionados con la HCI del sistema a aquellos relevantes para SMAs. Este giro no

implica una guía completamente nueva. Las *áreas* de la versión original siguen siendo plenamente relevantes para el desarrollo de SMAs. Sin embargo, sus *aspectos* tienen que ser reconsiderados para adaptarlos a las consideraciones específicas del diseño bajo el paradigma de agentes, tales como flujos de trabajo y nociones de coordinación o estado mental. Pero el cambio de foco en el análisis también incluye el punto de vista desde el que se realiza el estudio. La GA se centra en la interacción con el sistema y lo contempla como una caja negra monolítica. En oposición, para capturar los requisitos de SMAs a lo largo de todo el proceso software interesa prestar atención también a cómo se concibe la interacción entre los componentes del sistema. La "personificación" que implican las abstracciones del paradigma de agentes permite una aproximación similar al *role playing* centrada en las entidades activas del sistema. Los clientes pueden especificar el sistema pensando en como actuarían si fuesen sus usuarios, agentes, roles u organizaciones.

El segundo eje de cambio es el nivel de detalle. La GA es más concreta que la genérica TA, pero está todavía muy lejos del nivel de detalle acerca de los requisitos del sistema necesario para servir de base al análisis. Para resolver este problema, las *áreas* de introspección y las *preguntas* relacionadas son revisadas para lograr formulaciones más detalladas, que permitan a los desarrolladores obtener visiones de grano fino de los sistemas y sus entornos. La guía modificada considera cuestiones acerca de elementos tales como las tareas en los flujos de trabajo, sus entradas y salidas o los roles de los agentes involucrados.

El segundo objetivo es proporcionar mecanismos para integrar la GA en un proceso software soportado por la TA. En apartados anteriores (ver secciones 3.5 y 3.6) se introdujo la infraestructura para utilizar las técnicas de la TA en el desarrollo de SMAs. Esta infraestructura, basada en el lenguaje UML-TA y la representación con patrones de las *propiedades sociales*, será la empleada en la adaptación de la GA. La redefinición de la GA se basará en contemplar sus *preguntas* como *propiedades descriptivas* de la TA (ver sección 3.6). La representación con estructuras de *propiedades sociales* da indicaciones de cómo reflejar en un lenguaje de diseño el conocimiento sobre los requisitos. Además, el uso de estos patrones con conceptos de TA posibilita la referencia cruzada entre *preguntas*. Toda o parte de la información buscada con una cuestión pueda ser obtenida de otros patrones de *preguntas* ya contestadas. De esta forma los clientes no necesitan responder a todas las cuestiones o rellenar cada concepto en ellas, porque parte de su información ya puede estar disponible desde otras *preguntas*.

El tercer objetivo se relaciona con ayudar a clientes y desarrolladores a comprender las *preguntas* y responderlas. En este sentido, las preguntas en lenguaje natural han sido reformuladas para el dominio de la ISOA, donde los usuarios, clientes y desarrolladores carecen de conocimientos sobre la TA o si los tienen son muy someros. Adicionalmente, las *preguntas* tienen asociados ejemplos de respuestas previas en otros proyectos. Estos ejemplos proporcionan al equipo información adicional sobre la clase de conocimiento que se persigue con las *preguntas* o sus relaciones con el resto de los requisitos.

El siguiente apartado muestra un proceso de modificación de la GA para satisfacer estos objetivos. El proceso establece varios pasos para analizar y cambiar los diferentes componentes de la GA y sus enlaces con el resto del ciclo software.

4.3.2. Proceso de redefinición

Esta sección establece un proceso (ver en la Fig. 27 una vista general que se amplia con las Fig. 28 y Fig. 29) para redefinir la GA original [Kaptelinin *et al.* 1999], a fin de adaptarla a su uso en la captura de requisitos para SMAs dentro de un proceso software soportado por técnicas de TA. La resultante GCR respeta la estructura de la versión original con *áreas*, *aspectos* y *preguntas*. Se produce un cambio en la representación de las *preguntas*, que son vistas como *propiedades descriptivas* (ver sección 3.6). Es decir, las *preguntas* se describen mediante patrones de detección y solución que incluyen una descripción textual y otra con el lenguaje UML-TA.

Brevemente, el proceso de modificación de la Fig. 27 consta de cuatro pasos. En primer lugar, el desarrollador inicializa la nueva GCR con la GA, es decir, añade sus *áreas*, *aspectos* y *preguntas* a la GCR. Después realiza un ciclo en el que asocia un requisito del dominio considerado con un par *área-aspecto* de su GCR. Tras seleccionar este par se construye el grupo de *preguntas* adecuado para tratar la información seleccionada. En caso necesario se pueden añadir componentes nuevos a la GCR. Este proceso se repite mientras que existan requisitos a considerar que no estén incluidos en la GCR. Cuando no hay más requisitos a considerar el ciclo termina. El proceso acaba eliminando de la GCR aquellos componentes de la GA que no han sido usados. El resto de la presente sección estudia con mayor detenimiento estos pasos.

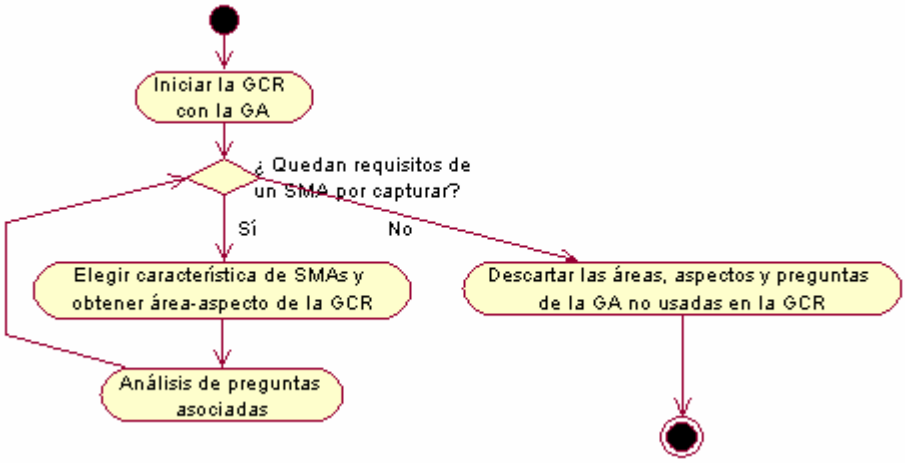


Fig. 27. Proceso de redefinición de la Guía de Actividades en Guía de Captura de Requisitos para SMAs.

Como ya se ha indicado, el método comienza inicializando la GCR con las *áreas*, *aspectos* y *preguntas* de la GA. Se puede pensar en realizar otras inicializaciones, por ejemplo con la GCR vacía, pero usar la GA proporciona un punto de partida que ya incluye características relevantes de los SMAs. En este paso sólo se da la forma textual de las *preguntas* como *propiedades descriptivas*. A continuación se inicia un ciclo de estudio de los componentes de la GCR de acuerdo con los requisitos.

El primer paso es la selección de una característica propia de los SMAs y su *área* y *aspecto* asociados. El desarrollo de este paso puede verse en la Fig. 28. Se comienza seleccionando un elemento de conocimiento requerido para modelar un SMA. Esta información tiene que ser conectada con su correspondiente *área* de la GCR y asociada con alguno de sus *aspectos*. Si este *aspecto* no existe en la GCR, uno nuevo debe ser creado.

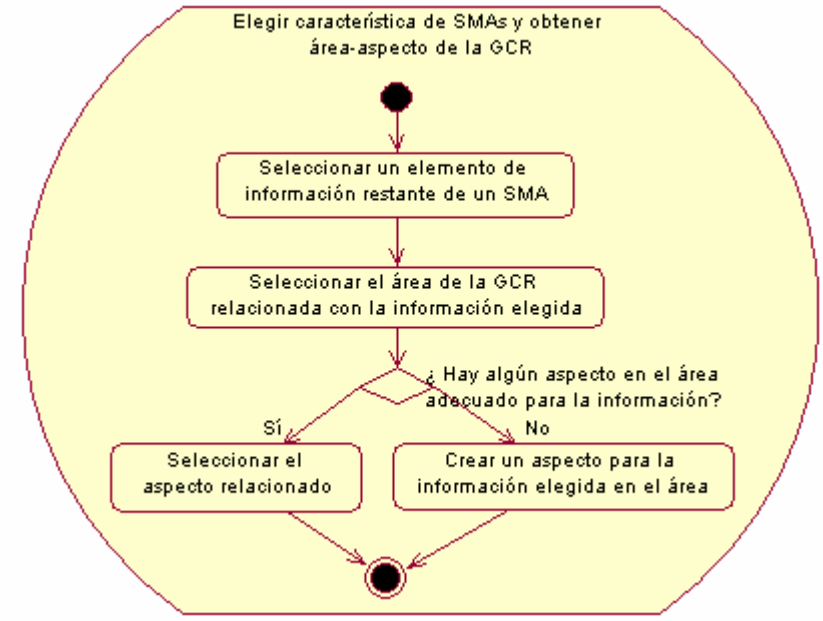


Fig. 28. Selección de la característica de un SMA y obtención de su *área* y *aspecto* asociados en la GCR.

Tras la selección previa se consideran las *preguntas* del *aspecto* seleccionado (ver Fig. 29). Estas *preguntas* se analizan a la luz de su adecuación al doble objetivo de las mismas en este trabajo: recoger el conocimiento requerido para modelar los SMAs; y tener una traducción conocida a conceptos de la TA. El primero se relaciona con la naturaleza de los requisitos y con que la GCR es una herramienta para su captura. Esta faceta se basa en la representación textual de las *propiedades sociales*: las *preguntas* se describen en lenguaje natural con una formulación que no requiera conocimientos específicos acerca de los SMAs o la TA. Esta representación pretende favorecer la discusión entre clientes y desarrolladores acerca de la información solicitada. El segundo objetivo está relacionado con la conexión con el resto del proceso de desarrollo. Aquí, el proceso aprovecha la representación con patrones UML-TA de las *propiedades sociales*. Esta representación actúa como un marco con ranuras que se rellena con las respuestas de los clientes. De esta forma, la información es traducida desde el lenguaje natural al mismo vocabulario de la TA involucrado en el desarrollo. Por tanto, el objetivo del análisis de un *aspecto* dado es obtener, o construir si es necesario, *preguntas* que satisfagan los dos objetivos previos usando

las notaciones propuestas. El análisis puede dividirse en dos pasos, uno para la forma textual y otro para el patrón con UML-TA.

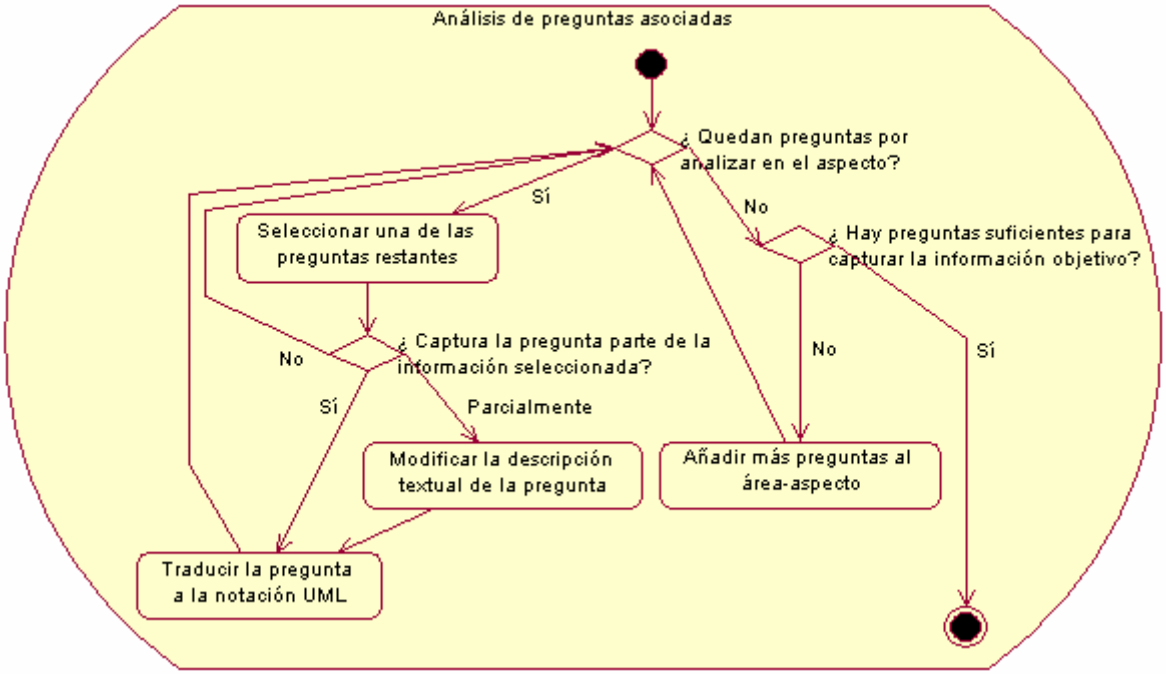


Fig. 29. Estudio de las *preguntas* asociadas a una característica de SMA, su *área* y su *aspecto*.

Al comienzo, los desarrolladores analizan la forma textual de cada *pregunta*. Las formas textuales deben ser comprensibles por no-expertos y tienen que proporcionar elementos de la información requerida sobre su *aspecto*. Hay que tener en cuenta que en muchas ocasiones los usuarios sólo ven la situación desde una perspectiva, por lo que es conveniente contar con múltiples *preguntas* que inciden en informaciones similares desde distintos puntos de vista. Este paso necesita una fase de prueba con personas sin conocimientos ni de la TA ni de SMAs. Las cuestiones que no satisfagan estos criterios deben ser eliminadas o modificadas.

El segundo paso en el análisis es la traducción de la *pregunta* de su forma textual a UML-TA. La forma UML de las cuestiones da a los desarrolladores una idea del tipo de información que se ha de esperar en las respuestas a esas *preguntas*. Además, proporciona las plantillas para poner la información de las respuestas en un lenguaje que pueda ser usado en el resto del proceso software. Por tanto, esta representación debe tener en cuenta no sólo los conceptos de la TA involucrados, sino también cuáles deben ser las ranuras que el usuario puede modificar para introducir la información de la respuesta. Al realizar este paso hay que recordar que la representación de las *preguntas* como *propiedades sociales* (ver sección 3.6) admite múltiples patrones UML-TA que recogen distintas interpretaciones de la forma textual.

Este tipo de traducción de la forma textual a UML es muy similar a aquellas hechas en la conceptualización de bases de datos [Elmasri & Navathe 1994] o en el

análisis orientado a objetos [Booch 1994]. Las principales fuentes de ayuda para realizar esta traducción son los casos de estudio sobre TA ya realizados en Sociología [Vygotsky 1978, Leontiev 1978, Engeström 1987, Kuutti 1996, Bednyi & Meister 1997, Gould *et al.* 2000, Uden *et al.* 2001]. Estos casos no proporcionan correspondencias con lenguajes de análisis como los usados en Ingeniería del Software, pero ilustran cómo interpretar la forma textual de las *preguntas* y sus respuestas a la luz de la TA.

El último paso del ciclo consiste en determinar si las cuestiones resultantes de este análisis son suficientes para capturar toda la información buscada. En el caso de que no lo sean se hace necesario escribir nuevas *preguntas* en la forma ya descrita.

El ciclo de creación de estudio de características, *aspectos* y *preguntas* termina cuando el desarrollador de la GCR considera que su versión de la guía captura el conocimiento que desea. En ese momento pueden existir *preguntas* e incluso *aspectos* de la GA original que no han sido asociados a información del dominio. Estos componentes sin asociación serán eliminados de la GCR final.

Todo este estudio para crear una GCR requiere un profundo conocimiento de la GA, la TA y los SMAs, pero sólo necesita hacerse una vez para un dominio dado. La guía resultante puede ser reutilizada sin cambios en otros desarrollos con similares características, siendo parte de un conjunto ya disponible de conocimientos y técnicas. La GCR presentada en la sección 4.5 de este capítulo es una versión genérica para SMAs, pero es posible crear versiones específicas para dominios concretos como asistentes en el PC o recuperación de información. Un ejemplo de automatización del uso de la GCR aparece en el Capítulo 6 dedicado a las herramientas de ayuda al desarrollo.

4.3.3. Casos en la captura de información

La aplicación del proceso de la sección anterior puede dar lugar a diferentes situaciones en la adaptación de los *aspectos* y *preguntas*. El caso depende de la relación entre la información que se desea capturar y la GA original. Esta sección muestra los tres posibles casos que se han identificado en la elaboración de la GCR. El primero corresponde a la situación en la cual la GA es directamente aplicable al problema con los SMAs. En este caso tan sólo es necesario proporcionar la representación con el patrón UML-TA de la *pregunta* involucrada. En el segundo, la GA requiere algunas modificaciones en su uso con SMAs, no acerca del *aspecto* pero sí de sus *preguntas* relacionadas. Este cambio se traduce en una reformulación de las *preguntas* originales para la GCR. El tercero es un caso de información útil para el modelado de SMAs pero que no está incluida como un *aspecto* de la GA original. En esta situación se hace necesario crear un nuevo *aspecto* en la GCR y establecer las *preguntas* necesarias para capturar su información asociada.

4.3.3.1. Aplicación directa

Aunque la GA está orientada al análisis de la interacción de los usuarios con el sistema, contempla algunos *aspectos* y *preguntas* relevantes para SMAs. Para estos casos, la única adaptación requerida es el estudio de cómo se debe representar la *pregunta* con un patrón en términos del lenguaje UML para la TA. Esta forma UML

debe considerar la información que se desea que proporcione el usuario en su respuesta. El ejemplo de este caso es una *pregunta* sobre contradicciones entre los objetivos.

Uno de los *aspectos* a analizar acerca del *área* de *Medios/fines* es "Potenciales conflictos entre objetivos considerados". Este *aspecto* trabaja sobre las contribuciones negativas dentro del conjunto de objetivos a satisfacer tanto del sistema como del entorno. La *pregunta* "¿Hay conflictos entre los diferentes objetivos del usuario? En caso afirmativo, ¿cuáles son los compromisos y reglas actuales para resolver los conflictos?" está relacionada con este *aspecto*. Tanto el *aspecto* como la *pregunta* contemplan información relevante para un SMA. Incluso aunque la formulación de la pregunta es demasiado evidente para revelar problemas ocultos, representa información que debe ser recogida. Una posible representación gráfica para la primera parte de esta *pregunta* puede ser vista en Fig. 30.

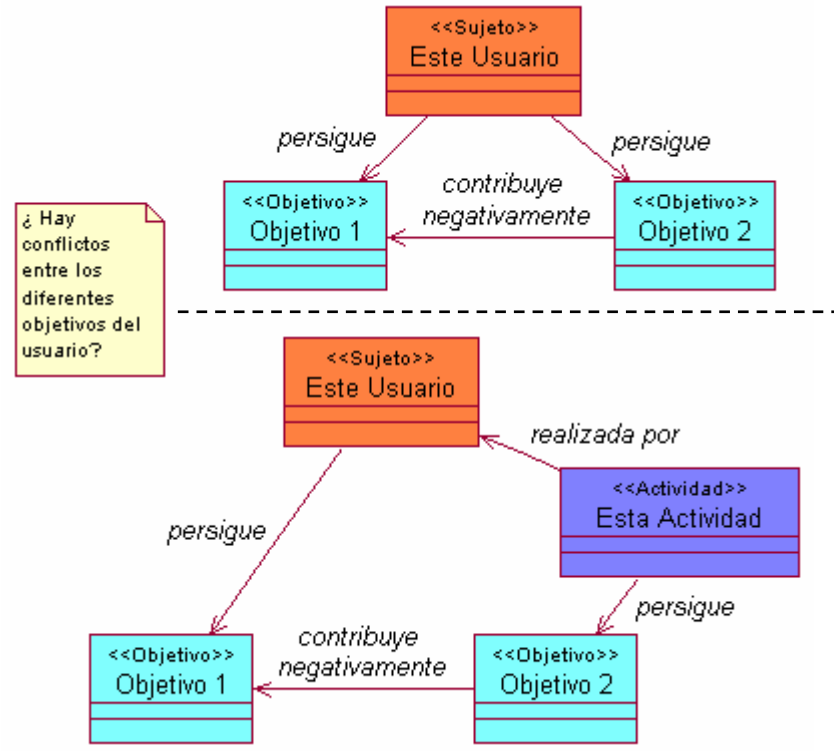


Fig. 30. Ejemplo de representación gráfica de una *pregunta* sobre conflictos entre objetivos.

El diagrama superior de la Fig. 30, muestra que el usuario puede tener conflictos, o *contradicciones* en el vocabulario de la TA, entre sus objetivos porque estos son incompatibles en sí mismos. El diagrama inferior ilustra otra posible fuente de conflictos, que el *sujeto* lleve a cabo una *actividad* que satisfaga un *objetivo* contradictorio con alguno de los del usuario.

4.3.3.2. Corrección de las *preguntas*

La segunda posibilidad acerca la información que se desea obtener de un SMA, es que se encuentre recogida como un *aspecto* de la GA pero su formulación como *pregunta* no sea adecuada. Aquí hay que considerar el distinto foco de la GA original, centrada en aspectos de HCI, y su grano de su estudio, más grueso que el requerido para unos requisitos, con respecto a la GCR. La adaptación requerida incluye en esta ocasión el estudio de cuáles son las *preguntas* que capturan mejor la información necesitada para un SMA. Para estas *preguntas* se debe crear tanto la forma textual como el patrón UML-TA. Un ejemplo de esta situación es el *aspecto* sobre el papel de la tecnología en uso para satisfacer las necesidades de su organización.

Un *aspecto* de interés acerca de un sistema, es el "Rol de la tecnología existente en la generación de los productos de las acciones objetivo" perteneciente al *área* del *Entorno*. Este *aspecto* trata de la importancia que la tecnología en uso (tal como recursos, aplicaciones o agentes) tiene para producir el resultado deseado de las acciones. Con este *aspecto* los desarrolladores pueden evaluar la importancia e integración del SMA en una *actividad* dada. Una de las *preguntas* relacionadas es "¿Se considera la tecnología una parte importante de las actividades del trabajo?". Tal y como está formulada, la cuestión requiere una evaluación subjetiva por parte del usuario que la responde. Además, el término tecnología parece demasiado genérico y sería mejor sustituirlo por uno que aludiera al SMA en construcción, por ejemplo "sistema objetivo". En este caso, la *pregunta* pueda ser sustituida por ésta "¿Cuáles son las actividades de trabajo en las que participa el sistema objetivo?". Esta nueva expresión da una idea de la extensión del sistema en la organización y la importancia de los procesos que la involucran. Su representación puede verse en la Fig. 31.

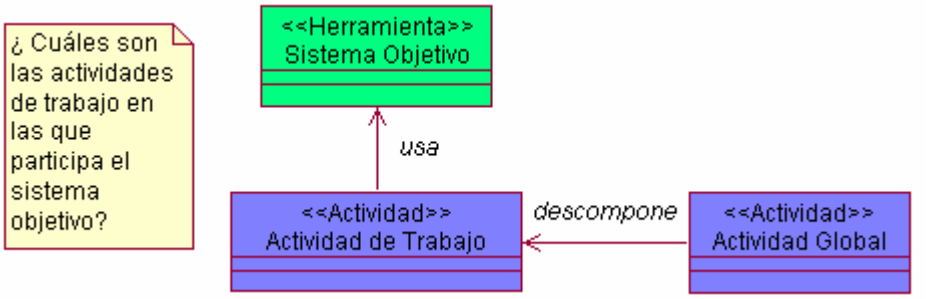


Fig. 31. Ejemplo de representación gráfica del rol del sistema en las *actividades* actuales.

El diagrama representa para los desarrolladores el hecho de que la importancia del sistema para la organización está relacionada con las *actividades* en las cuales participa. Estudiando estas actividades, los usuarios y los desarrolladores pueden dar una mejor evaluación del rol del sistema en la organización.

4.3.3.3. Nuevos *aspectos*

Hay también algunas características importantes de los requisitos de un SMA que no aparecen en absoluto en la GA original. Esta información requiere decidir cuál es la *área* que le corresponde, después crear un *aspecto* para dicha información y finalmente, crear las *preguntas* necesarias como sucedía en la sección anterior. Un

ejemplo de esta situación sería el estudio de los flujos de trabajo en que participan los componentes de un SMA. La funcionalidad de los SMAs es frecuentemente modelada por medio de flujos de trabajo, esto es, secuencias de tareas que usan recursos, consumen elementos (que se relacionan con sus precondiciones) y producen otros elementos (que se relacionan con sus postcondiciones). Esta información es de un grano más fino que la involucrada en la GA original y por tanto no está recogida en ella. A pesar de esto, saber cómo interaccionan los *sujetos* involucrados y sus dependencias es muy relevante para el estudio de SMAs.

La información acerca de los flujos de trabajo en el contexto de un SMA puede ser considerada en el *área* de *Entorno* con *preguntas* como "¿Qué productos se necesitan para llevar a cabo esta actividad?" y "¿Cuáles son los productos esperados de esta actividad?". Su representación gráfica puede ser como la presentada en Fig. 32.

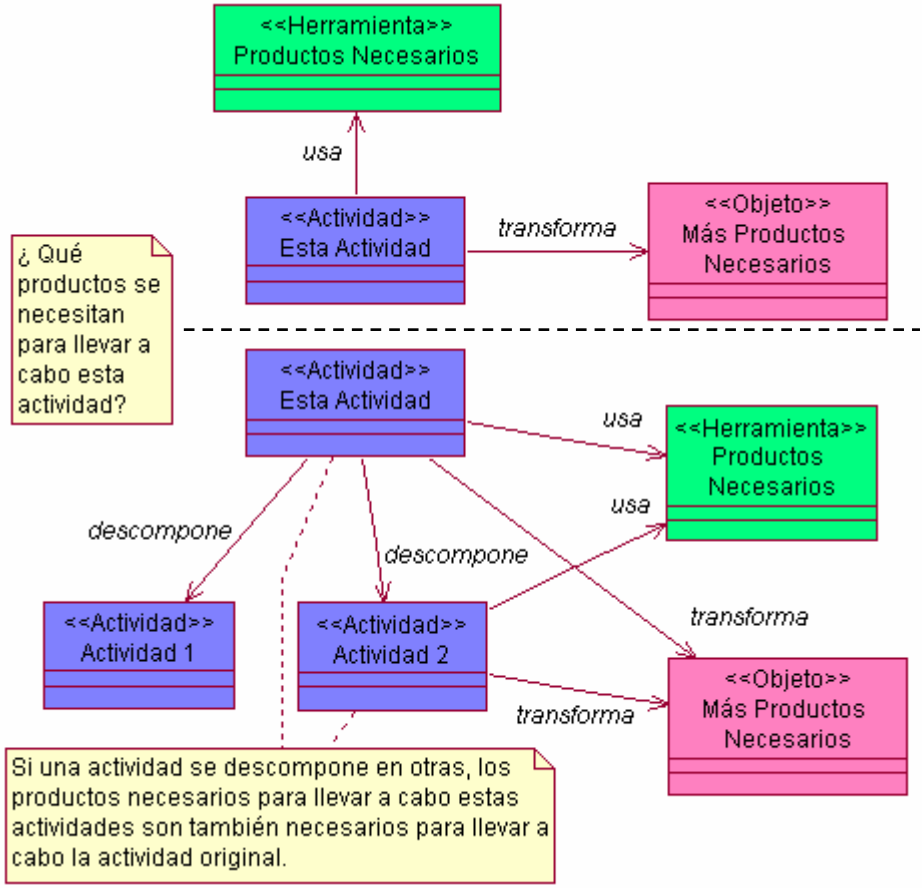


Fig. 32. Ejemplo de representación gráfica de una pregunta sobre los elementos consumidos y producidos en los flujos de trabajo.

La representación gráfica muestra el hecho de que los productos necesarios para llevar a cabo una *actividad* son aquellos directamente usados por la *actividad* o aquellos usados por *actividades* que pertenecen a su descomposición.

4.3.4. Ejemplo de aplicación del proceso de redefinición

Para ilustrar el proceso de creación de la GCR introducido en la sección 4.3.2 vamos a seguir la incorporación de una información sobre SMAs a ella. La información que se desea capturar con la guía es qué datos permiten saber si un objetivo se ha satisfecho o no. Estos datos pueden formar parte de las entidades del estado mental del agente o de su entorno.

El proceso comienza con la inicialización de la GCR con la GA. Se añaden las *áreas* de la GA con todos sus *aspectos* y *preguntas* a una GCR vacía. Las *preguntas* sólo cuentan en este momento con su forma textual.

A continuación se trata de responder a la cuestión de si quedan requisitos por capturar. En este caso la respuesta es afirmativa: queda por capturar la información sobre las evidencias que permiten determinar la satisfacción de un objetivo.

Después hay que determinar el *área* donde se debe contemplar esta información. Para tomar esta decisión se cuenta con las definiciones de *áreas* de la introducción a la GA en la sección 4.2. El *área* de *Desarrollo* se centra en la evolución prevista de las *actividades* por la introducción de nuevos componentes, por lo que no parece adecuada para esta información. Tampoco *Aprendizaje/cognición/articulación*, donde se considera el aprendizaje acerca del componente y la articulación de los actores en torno a él para llevar a cabo las *actividades*, se relaciona con la información sobre los objetivos. El *área* de *Entorno* podría ser adecuada para recoger esta información. Las evidencias sobre la satisfacción de objetivos pueden encontrarse en el entorno. Sin embargo, como esta *área* se centra sobre todo en el contexto externo, no incluiría aquellos casos en los que la evidencia forma parte del estado interno de los agentes o del sistema. La última *área* es la de *Medios/fines*, que considera de una forma genérica los objetivos y sus relaciones con la organización externa. Por ello, ésta parece el *área* más adecuada para contemplar esta información. Esta decisión se ve avalada por el hecho de que los *aspectos* sobre criterios de satisfacción de objetivos o su operacionalización se contemplan en esta *área* en la GA.

Tras elegir el *área*, hay que buscar el *aspecto* que contempla la información que se desea incluir en la GCR o crear uno nuevo si no existe. Como ya se ha indicado, la GA original incluye un *aspecto* sobre *Criterios para la satisfacción o fallo de los objetivos* que es el seleccionado.

Una vez que se ha determinado la información sobre SMAs a incluir y el *área* y *aspecto* donde considerarla, hay que analizar las *preguntas* asociadas. En los trabajos estudiados sobre la GA original, tales como [Kaptelinin *et al.* 1999, Gould *et al.* 2000, Hedestig & Kaptelinin 2002], no se han encontrado *preguntas* sobre este *aspecto*. En el proceso de la Fig. 29, la respuesta a si quedan *preguntas* a analizar del *aspecto* y si las existentes son suficientes para capturar la información considerada es, obviamente, negativa. En este caso el diagrama sugiere añadir *preguntas* en su forma textual. Una candidata evidente podría ser "¿Cómo se puede saber que el objetivo ha fallado?". Con esta formulación puede empezarse el estudio de la *pregunta*.

La primera cuestión requiere saber si la *pregunta* está capturando parte de la información deseada. La respuesta es afirmativa, aunque sería preferible concretar más la formulación. Se considera por tanto que captura el conocimiento parcialmente. El proceso sugiere entonces modificar el texto de la *pregunta*. Se adopta una nueva formulación que es "¿Cómo se puede saber que el objetivo ha fallado? ¿Cuáles son las evidencias/datos que muestran este fallo?", que indica más claramente que se desean conocer datos concretos sobre los que evaluar la satisfacción.

Una vez que se posee una forma textual adecuada, hay que representarla con UML-TA. Realizar esta traducción es fundamentalmente un proceso de interpretación guiado por el conocimiento y experiencia con la TA, UML-TA y su aplicación en la ISOA, como se indicó en la sección 4.3.2. Para los objetivos, la decisión evidente es representarlos como *objetivos* de la TA. El que una evidencia implique el fallo de un objetivo puede representarse como una relación *impide*. La parte de interpretación más difícil en este caso es ver qué evidencias pueden implicar el fallo de un objetivo de un SMA, cuál es su *rol* en la TA y si ese *rol* tiene sentido en una relación *impide* con un *objetivo*. Dependiendo de la metodología de la ISOA, los agentes pueden establecer el fallo de un objetivo en base a evidencias muy diversas, tales como entidades mentales, eventos del entorno, presencia de agentes o capacidades de roles. Estos elementos desempeñan *roles* distintos en un *sistema de actividad*. Por ejemplo, las entidades mentales pueden actuar como *objetos*, los eventos como *herramientas* y los roles como *sujetos*. Para cubrir todos estos *roles* de la TA sólo se puede emplear un *artefacto*. Una vez que se han determinado los *roles* y relaciones de la TA que aparecen en el patrón, quedan por determinar las ranuras modificables del marco. Puesto que la *pregunta* es genérica sobre *objetivos* y *artefactos*, los nombres de estos *roles* serán editables. Además, también será modificable el *rol artefacto* para poder fijarlo a otro *rol* de la TA más concreto si fuese necesario. La representación UML-TA final de la *pregunta* queda como se ve en la Fig. 33. Como se señaló en la sección 4.3.2, la representación UML-TA de una *pregunta* no tiene que ser única porque se pueden tener diferentes interpretaciones de cómo modelar la respuesta a la *pregunta*.

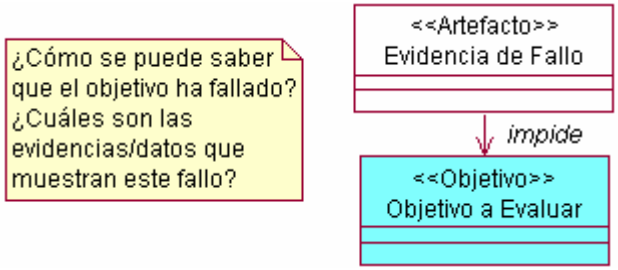


Fig. 33. Representación con UML-TA de una *pregunta* sobre el fallo de objetivos.

Tras realizar la representación con UML-TA, se seguirían analizando otras *preguntas*. Si no quedaran más, se volvería a plantear la cuestión de si las *preguntas* analizadas son suficientes para capturar la información deseada. Si no fuera así, se añadirían más *preguntas* de la misma forma que se ha hecho con el ejemplo que se acaba de ver. Si fueran suficientes se pasarían a analizar nuevas características de los SMAs. El proceso terminaría cuando se hubieran incluido en la GCR todas las características a considerar de los SMAs en estudio.

4.4. Método de aplicación de la GCR

Una vez descrito el proceso para crear la GCR en la sección 4.3, este apartado describe cómo aplicarla. El método de utilización propuesto para la GCR puede verse en la Fig. 34. Este método forma parte de una fase de captura de requisitos en metodologías de SMAs y se encuentra inmerso en un proceso de desarrollo soportado por la TA. El uso de la GCR se muestra en este contexto global para dar una visión más completa de cómo se capturan y refinan los requisitos.

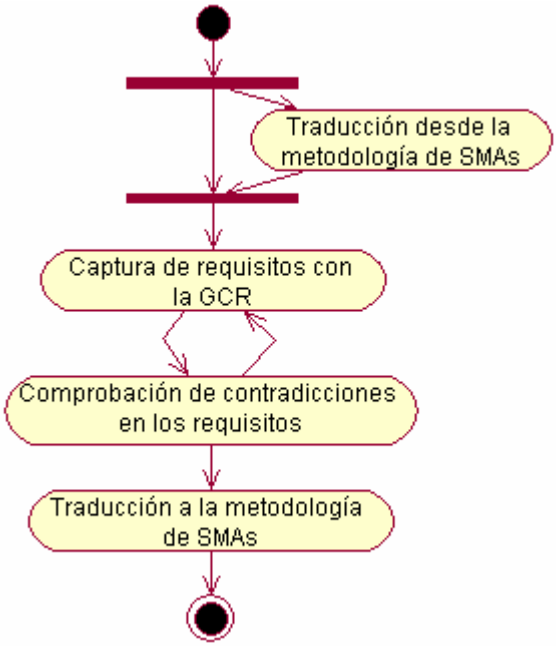


Fig. 34. Método de obtención de requisitos con la Guía de Captura de Requisitos.

El proceso de la Fig. 34 consta de cuatro etapas. El primer paso del proceso consiste en obtener el contexto de los requisitos establecido por la arquitectura del SMA. En la introducción a este capítulo se subrayó que en un proceso iterativo e incremental de desarrollo, los nuevos requisitos debían tener en cuenta los compromisos previamente adoptados en la construcción del sistema. En el caso de que se trate de las iteraciones iniciales del desarrollo no es preciso realizar esta traducción de las especificaciones del SMA a la TA. Después se han de responder las *preguntas* de la GCR y describir las respuestas en UML-TA. Aquí hay que recordar que las *preguntas* de la GCR son *propiedades descriptivas* (ver sección 3.6). Las *preguntas* incluyen patrones en UML-TA que actúan como marcos que se rellenan con las respuestas de los clientes. Estas respuestas se analizan luego siguiendo las *contradicciones* obtenidas de la investigación en TA. La forma de uso de estas *contradicciones* se verá en el Capítulo 5. Las contradicciones contribuyen a guiar el

refinamiento de la información en los requisitos. Se trata de un proceso iterativo y cíclico: las *preguntas* respondidas hacen surgir nuevas contradicciones; resolver las contradicciones requiere nueva información que se ha de obtener respondiendo más *preguntas*. El proceso de refinamiento termina cuando los requisitos se consideran suficientemente completos y sus contradicciones inherentes al SMA y su entorno. Esta valoración es subjetiva, en el sentido de que no se exige responder todas las *preguntas* de la GCR para todos los elementos ni que no existan contradicciones. El desarrollador ha de juzgar si tiene suficiente información para proseguir con el desarrollo. En el momento en que termina el ciclo de refinamiento, las especificaciones con la nueva información se traducen a las abstracciones propias de la metodología de SMAs. Los requisitos traducidos al paradigma de agentes son la base del resto del proceso de desarrollo.

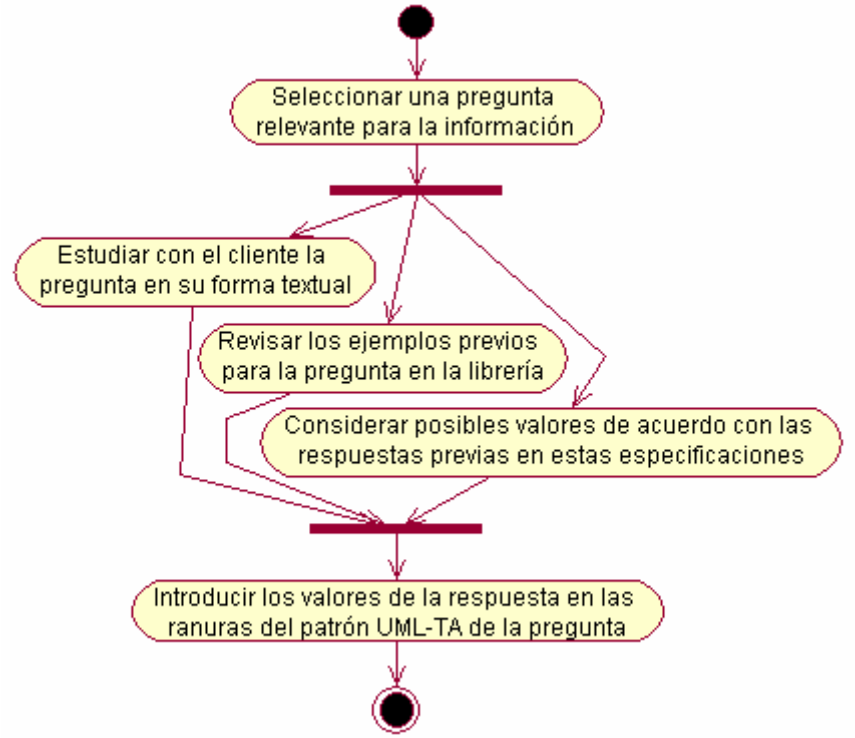


Fig. 35. Proceso de respuesta a las *preguntas* de la Guía de Captura de Requisitos.

La Fig. 35 ilustra con mayor detalle el modo en el que se responden las *preguntas*. Los usuarios y los desarrolladores han de llevar a cabo esta tarea conjuntamente: los primeros conocen el dominio real del sistema; los segundos han de comprender el entorno para implementar la solución. El grupo que realiza la captura de requisitos trabaja seleccionando *preguntas* de la GCR. Dependiendo de la situación, se puede elegir una cuestión sobre características del sistema que no han sido tratadas aun; en otros casos, la *pregunta* corresponderá a características de los requisitos sobre los cuales existen dudas. Las *preguntas* en su forma textual y UML dan a clientes y

desarrolladores el punto de partida para la discusión, para descubrir la información y para ponerla en su marco. En esta tarea se pueden consultar librerías con ejemplos de *preguntas* y respuestas en otros proyectos. De este modo la gente puede hacerse una idea más acertada de la información que se pretende capturar con la *pregunta*. Además, la información ya introducida en la especificación puede ser usada para rellenar automáticamente las ranuras del patrón UML-TA.

El proceso de esta sección no pretende capturar todos los requisitos de un SMA. Es sólo una herramienta para recordar las características importantes de un SMA en los requisitos y proponer puntos de introspección cuando el estudio no se logra completar. Por supuesto, su uso se ha de complementar con otras técnicas de Ingeniería de Requisitos.

4.5. Guía de Captura de Requisitos para SMAs

La sección 4.3 ha introducido cuáles deben ser los cambios en la GA para usarla en la captura de requisitos de SMAs, cómo hacer este cambio y las principales situaciones que pueden surgir. Este apartado muestra el resultado de aplicar ese proceso: la Guía de Captura de Requisitos para SMAs (GCR), es decir, la redefinición de la GA para la ISOA. Los componentes de la GCR están inspirados por la GA y surgen de la experiencia en el trabajo con la TA, la GA y el desarrollo de SMAs. Esta guía no pretende ser exhaustiva en el sentido de recoger todos los posibles requisitos de un SMA. Antes bien, se trata de un ejemplo reducido de viabilidad de este método.

De acuerdo con las líneas expuestas en la sección 4.3, se considera que los *aspectos* y *preguntas* deben contemplar las diferentes perspectivas posibles en el SMA, e.g. un usuario, una organización, un agente o un rol. Para ello, en los puntos donde se considere necesario por la información a capturar, se sustituirán términos como "usuario", "cliente" o "agente" por el más genérico para elementos intencionales de "actor". También los términos "Organización" y "Grupo" pueden referirse tanto a elementos del entorno del futuro sistema como a partes de éste, ya que se modela como un SMA donde estos conceptos tienen sentido como abstracciones de diseño. Cuando el usuario entrevistado responda a *preguntas* que involucren actores, deberá hacerlo considerando que juega el papel asociado a ese actor. También se introduce el término "componente" para referirnos a cualquier clase de artefacto en el modelado –no confundirlo con el *artefacto* de la TA. Así, el "componente" puede ser tanto el elemento tecnológico a introducir como los distintos elementos del SMA o su entorno, incluyendo recursos, normas o estructuras organizativas. En todo caso, las aclaraciones necesarias acerca de posibles ambigüedades se incluirán en la explicación textual.

También siguiendo lo expuesto en la sección 4.3, se trata de capturar la misma información desde distintas perspectivas. El usuario puede concebir una situación dada desde un punto de vista determinado y se han de proporcionar *preguntas* capaces de recoger la información con esa perspectiva.

A fin de facilitar la descripción de las *preguntas* de la GCR, se ha separado la descripción de la guía por *áreas* y, para éstas, una introducción en forma de tabla de su desarrollo. En el desarrollo se explica con mayor detalle la representación textual y

se introduce un posible patrón UML-TA de las *preguntas*. Este desarrollo sólo incluye las *preguntas* marcadas en las tablas con el símbolo "✓". Las *preguntas* elegidas son representativas de todas las *áreas* y han sido elegidas entre las más usadas en la experimentación para mostrar la viabilidad de la GCR. Señalar también que en las tablas se han numerado los *aspectos* y las *preguntas* para facilitar la referencia posterior a las *preguntas*. Así, la *pregunta* i.j.k corresponde al *área* i, *aspecto* j y *pregunta* k.

Los siguientes patrones siguen la convención introducida en la sección 3.4 de representar entre ' " ' y ' " ' (comillas dobles) los valores que deban ser interpretados como literales, i.e. valores fijos, y sin comillas los que se interpretan como variables.

4.5.1. Medios y fines

El resumen de la GCR para SMAs en el *área* de *Medios/fines* está recogido en la Tabla 12. Como se recordará de la introducción a la GA original en la sección 4.2, el *área* de *Medios/fines* hace referencia a la influencia que los componentes que se están proyectando tendrán sobre las necesidades de los actores y sus contradicciones.

VERSIÓN DE DISEÑO **Área de Medios/Fines**	
Aspectos	**Preguntas**
1. Actores del nuevo sistema	1. ¿Quiénes son los actores que se prevé que interaccionarán con el nuevo componente/elemento? 2. ¿Quiénes serán los actores cuyas actividades generen los datos/entradas necesarios para el componente? ¿Quiénes serán los actores que participen en los procesos que generan los datos/entradas necesarios para el componente? 3. ¿Quiénes serán los actores que usen los datos/salidas del componente? ¿Quiénes serán los actores que participen en los procesos que generan los datos/salidas del componente? 4. ¿Quiénes son los actores que interactúan con los anteriores? 5. Dados los objetivos, ¿qué actor debería hacerse cargo de ellos? 6. Dadas las tareas, ¿qué actor debería hacerse cargo de ellas? 7. ¿Qué actores están interesados en la realización de la tarea?
2. Objetivos relacionados con el nuevo componente	1. ¿Por qué desea el actor/organización crear el componente? ¿Cuáles son sus objetivos? 2. ¿Qué esperan los actores que resuelva el componente? 3. ¿Existen inconvenientes para la organización o partes de ella en la construcción del componente? ✓ (ver sección 4.5.1.1) 4. ¿Cuáles son los motivos de los actores para emplear el nuevo componente? ¿Qué beneficios obtienen de su uso? 5. ¿Existen inconvenientes para los actores en la construcción/uso del componente? 6. ¿Cuáles deben ser los objetivos del nuevo componente?

	¿Cuáles deberían ser sus cometidos?
3. Acciones del nuevo componente	1. Al margen de los objetivos identificados, ¿existen procesos/tareas que deba soportar el componente? 2. ¿Qué tareas debería soportar el componente? 3. ¿Cuáles son las estrategias y técnicas de resolución de problemas usadas? 4. ¿Hay acciones, además de las acciones indicadas para el componente, que no estén contempladas pero donde los actores necesitan obviamente ayuda/herramientas?
4. Integración de acciones individuales en otras de más alto nivel	1. ¿Es necesario que el actor cambie constantemente entre la realización de diferentes acciones? ¿Qué objetivos/acciones ha de cubrir así? 2. ¿Existen grupos de acciones que el actor realiza siempre de forma conjunta? 3. ¿Existen "atajos" usados por los usuarios que les permitan una transición fácil entre acciones y, si es necesario, volver a estados o acciones previos?
5. Restricciones impuestas por los objetivos sobre la elección y uso de la tecnología	1. Dados los objetivos involucrados en este desarrollo, ¿cómo deberían ser las herramientas empleadas en él? Es decir, ¿qué características considera aconsejables? 2. ¿Qué características considera desaconsejables en la tecnología que se use para el sistema? 3. ¿Cuáles son las limitaciones de la tecnología actual? 4. Dados los objetivos involucrados en esta actividad, ¿cómo deberían ser las herramientas empleadas en ella? Es decir, ¿qué características considera aconsejables? 5. ¿Qué características considera desaconsejables en los componentes que soporten la actividad?
6. Objetivos que pueden ser cambiados y objetivos que han de permanecer después de introducir el nuevo componente	1. El componente introducirá cambios en el modo de trabajo del actor y en su entorno. ¿Qué objetivos entre los recogidos considera que no deben sufrir cambios? ✓ (ver sección 4.5.1.2) 2. ¿Por qué considera que esos objetivos no deben sufrir cambios? ¿Cuáles son los motivos? ✓ (ver sección 4.5.1.3) 3. ¿Qué objetivos entre los recogidos considera que pueden sufrir cambios? 4. ¿Por qué considera que esos objetivos pueden ser cambiados? ¿Cuáles son los motivos?
7. Refinamiento de los objetivos	1. ¿Cuáles son los objetivos concretos que plasman los objetivos del sistema? 2. A nivel de los usuarios, ¿cuáles son los objetivos que permiten realizar los objetivos del sistema? 3. ¿Cuáles son los objetivos concretos en la labor del actor que plasman los objetivos del componente? 4. ¿Cuáles son los objetivos concretos en la labor del actor que plasman los objetivos de la organización?

	5. ¿Existen formas alternativas de satisfacer los objetivos mediante otros objetivos o una combinación de estos? 6. ¿Existen objetivos que ayuden a satisfacer estos objetivos aunque no garanticen dicha satisfacción? 7. ¿Estos objetivos ayudan a satisfacer otros aunque no garanticen dicha satisfacción? ¿Cuáles son esos objetivos?
8. Operacionalización de los objetivos	1. ¿Qué procesos/acciones permiten satisfacer los objetivos? 2. ¿Qué procesos contribuyen a satisfacer los objetivos? 3. ¿Qué procesos influyen en la satisfacción de los objetivos? 4. ¿Qué procesos pueden impedir la satisfacción de los objetivos?
9. Objetivos de las acciones del componente	1. ¿Por qué ha de realizar/soportar el componente la acción? 2. ¿Cuáles son los objetivos de estas acciones? 3. ¿Contribuyen de forma indirecta estas acciones a otros objetivos?
10. Criterios de éxito o fallo para el diseño	1. ¿Cuándo consideraría que el componente ha sido bien construido? ¿Cuáles serían los criterios para comprobarlo? 2. ¿Cuándo consideraría que el componente ha sido mal construido? ¿Cuáles serían los criterios para comprobarlo? 3. ¿Por qué podría considerar el componente defectuoso? 4. ¿Será el componente capaz de soportar todas las acciones propuestas?
11. Criterios para la satisfacción o fallo de los objetivos	1. ¿Cómo se puede saber que el objetivo se ha cumplido? ¿Cuáles son las evidencias/datos/criterios que muestran su satisfacción? 2. ¿Cómo se puede saber que el objetivo ha fallado? ¿Cuáles son las evidencias/datos que muestran este fallo? 3. ¿Hay evidencias/datos que apunten a que el objetivo se ha satisfecho aun sin garantizarlo? 4. ¿Hay evidencias/datos que apunten a que el objetivo ha fallado aun sin garantizarlo?
12. Contexto de los objetivos	1. ¿Existen circunstancias previas necesarias para intentar satisfacer el objetivo? ¿Cuáles son las evidencias/datos que permiten saber que se da este contexto? 2. ¿Qué datos le permiten pensar que el objetivo puede satisfacerse en este momento? 3. ¿Qué datos le permiten pensar que el objetivo no debe intentar satisfacerse en este momento? 4. ¿Qué elementos piensa que contribuyen *a priori* a que el objetivo se satisfaga? 5. ¿Qué elementos piensa que contribuyen *a priori* a que el objetivo falle?
13. Conflictos potenciales entre objetivos	1. ¿Hay conflictos entre los diferentes objetivos del componente? 2. ¿Hay conflictos entre los diferentes objetivos del actor? 3. ¿Hay conflictos entre los diferentes objetivos de la organización?

	4. ¿Cumplir los objetivos del componente dificulta la satisfacción de otros objetivos de la organización? 5. ¿Cumplir los objetivos del actor dificulta la satisfacción de otros objetivos de la organización? 6. ¿Cumplir los objetivos del componente dificulta la satisfacción de otros objetivos de los actores? 7. ¿Cumplir los objetivos de la organización dificulta la satisfacción de los objetivos del componente? 8. ¿Cumplir los objetivos de la organización dificulta la satisfacción de otros objetivos del actor? 9. ¿Cumplir los objetivos de los actores dificulta la satisfacción de los objetivos del componente?
14. Conflictos potenciales entre los objetivos y los de otras actividades o tecnologías	1. Considerando el resto de la organización, ¿cree que puede haber conflictos con los objetivos de otros grupos? ¿Qué grupos? 2. Considerando el entorno del componente, ¿cree que puede haber conflictos con los objetivos de otros grupos? 3. ¿Qué objetivos de esos grupos serían conflictivos? ¿Con qué objetivos del componente tendrían conflictos? 4. Considerando otras actividades de su organización, ¿cree que puede haber conflictos con los objetivos del actor? ¿Qué actividades? 5. ¿Qué objetivos de esas actividades serían conflictivos? ¿Con qué objetivos del componente tendrían conflictos?
15. Potenciales conflictos entre los objetivos del diseño y otros objetivos	1. ¿Considera que puede haber conflictos entre los objetivos de diseño del sistema y los de la organización o los usuarios? Por ejemplo, con la estabilidad de la organización o con minimizar los gastos. 2. ¿Dificulta la construcción del sistema otros objetivos de su organización? 3. ¿Dificulta la construcción del sistema algún objetivo de los usuarios?
16. Resolución de conflictos entre objetivos	1. En caso de conflictos, ¿cuáles son los compromisos y reglas actuales para resolver los conflictos? 2. En caso de conflictos, ¿puede establecer prioridades entre los objetivos? ✓ (ver sección 4.5.1.4) 3. En caso de conflictos, ¿existen evidencias/datos que permitan al actor elegir qué objetivo satisfacer? ✓ (ver sección 4.5.1.5) 4. ¿Cuáles son las estrategias que emplean los actores habitualmente para resolver los conflictos entre objetivos?

Tabla 12. *Aspectos* y *preguntas* de la Guía de Captura de Requisitos para SMAs en su versión de diseño dentro del *área* de *Medios/fines*.

Los siguientes apartados de esta sección explican con mayor detalle las *preguntas* marcadas con "✓" introducidas en la Tabla 12 para el *área* de *Medios/fines*.

4.5.1.1. Pregunta 1.2.3

La *pregunta* "¿Existen inconvenientes para la organización o partes de ella en la construcción del componente?" trata de hacer patentes las reticencias o efectos negativos, que pueden presentarse en partes de la organización como consecuencia de la incorporación del nuevo elemento. Cuando se introduce un nuevo componente (i.e. una *herramienta* u *objeto*) en una organización, se altera el *statu quo* de dicha organización: cambian las necesidades, las relaciones de poder y los flujos de información [Ruohonen 1996]. Estos cambios pueden hacer que algunos de los actores de dicha organización sean contrarios, o al menos tengan reservas, a la implantación de dicho componente dado que perjudica a sus objetivos.

Las respuestas a esta *pregunta*, dadas por los diferentes clientes entrevistados, irán componiendo una visión conjunta de cuáles pueden ser los escollos organizativos al componente. Se pretende obtener un conjunto de objetivos de actores de la organización que pueden perjudicar a los objetivos relacionados con el componente. Los usuarios han de dar respuestas del estilo de "El *Componente* trata de satisfacer el *Objetivo del Componente* pero éste es contrario al *Objetivo del Grupo* que persigue *Grupo*". Esta información queda reflejada en la Fig. 36.

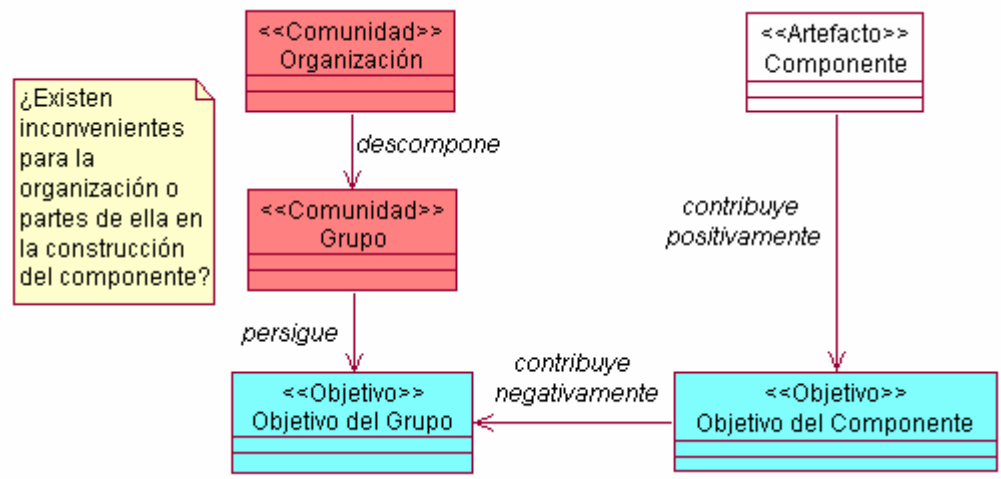


Fig. 36. Guía de Captura de Requisitos: ¿Existen inconvenientes para la organización o partes de ella en la construcción del componente?

La Fig. 36 corresponde a las contradicciones con los *objetivos* que perseguirá el componente, es decir, los *objetivos* que satisfará el componente al jugar su *rol* en el *sistema de actividad*. El diagrama refleja el hecho de que la *organización* puede incluir diferentes *grupos*, cada uno persiguiendo sus propios *objetivos*. El grupo puede ver algunos *objetivos* del componente como perjudiciales para los suyos. Estas contradicciones estarían en el origen de la resistencia de esos grupos o actores al funcionamiento del componente.

Al igual que ocurre con otras *preguntas* de la GCR, existen varias versiones alternativas de la formulación UML-TA de esta *pregunta*. Una primera posibilidad consiste en considerar directamente que la introducción del componente perjudica a

un *objetivo* del *grupo*. En este caso la representación UML no incluiría el *Objetivo del Componente* y tendría una relación *contribuye negativamente* del *Componente* al *Objetivo del Grupo*. Otra posibilidad es considerar que el *Objetivo del Componente* perjudica a la *Organización* en general. El diagrama para esta versión no incluye ni al *Grupo* ni al *Objetivo del Grupo* pero añade una relación *contribuye negativamente* del *Objetivo del Componente* a la *Organización*.

4.5.1.2. Pregunta 1.6.1

Al tratar de capturar los objetivos de un componente surgen algunos que son esenciales para su cometido y otros que no son críticos [Yu & Mylopoulos 1994]. Los objetivos no críticos representan características deseables del componente, que deben ser cubiertas siempre que no supongan un coste prohibitivo o entren en conflicto con los objetivos esenciales. La identificación de los objetivos "vitales" que debe cumplir el componente facilita el razonamiento cualitativo sobre las prioridades de los objetivos. Ello permite luego determinar posibles alternativas de diseño o plantear compromisos entre los actores cuando se necesite negociación entre objetivos.

La *pregunta* "El componente introducirá cambios en el modo de trabajo del actor y en su entorno. ¿Qué objetivos entre los recogidos considera que no deben sufrir cambios?" persigue obtener los requisitos que un actor identifica como esenciales para el componente. De esta pregunta se quieren obtener respuestas como "Para el *Grupo* es esencial que el *Componente* sea capaz de satisfacer el *Objetivo Vital*" quizás en asociación con un objetivo del componente. La representación de esta *pregunta* se ve en la Fig. 37.

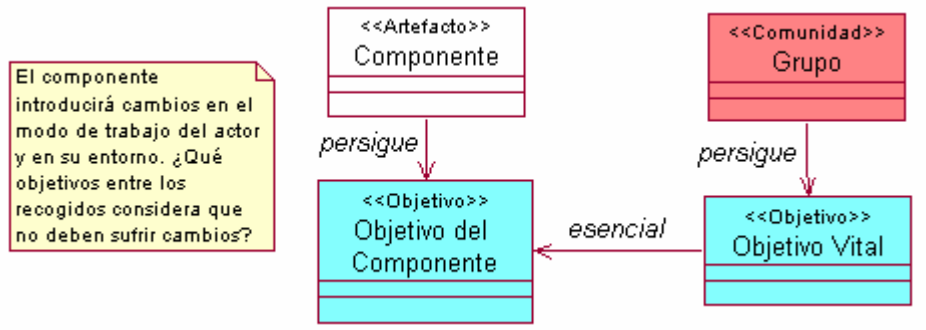


Fig. 37. Guía de Captura de Requisitos: ¿Qué objetivos entre los recogidos considera que no deben sufrir cambios?

La *pregunta* solicita al usuario que determine los *objetivos vitales* de los grupos. La forma de representar que son fundamentales es indicando que estos objetivos se han de satisfacer porque resultan *esenciales* para poder cumplir con algún *objetivo del componente*, es decir, la no satisfacción del *Objetivo Vital* del *Grupo* inhibe la satisfacción del *Objetivo del Componente*.

4.5.1.3. Pregunta 1.6.2

La forma de satisfacer los objetivos de una organización en un momento dado no tiene que ser necesariamente la única posible. Frecuentemente, objetivos que parecen esenciales sólo lo son porque su satisfacción se ve como la única forma posible de cumplir con otros objetivos de más alto nivel [Engeström 1990]. Si se descubren estos objetivos de alto nivel, el abanico de actividades capaces de satisfacerlos puede ampliarse, en lugar de quedar reducido a las soluciones ya en uso. De esta forma, el diseño del sistema cuenta con más alternativas lo que puede mejorar su rendimiento.

La *pregunta* 1.6.2 es una continuación natural de la 1.6.1. La *pregunta* 1.6.1 descubría objetivos esenciales del componente mientras que con la 1.6.2 se trata de determinar cuáles de esos objetivos responden directamente a las necesidades de los actores y cuáles corresponden a la satisfacción de otros objetivos de más alto nivel. La *pregunta* "¿Por qué considera que esos objetivos no deben sufrir cambios? ¿Cuáles son los motivos?" intenta que el usuario reflexione sobre las necesidades últimas a las que responden sus objetivos. Las respuestas a esta pregunta estarían en la línea de "El *Objetivo del Componente* en realidad es una forma de cumplir el *Objetivo Vital*, que es necesario para que el *Grupo* pueda cumplir con el *Objetivo del Grupo*". Esta *pregunta* queda reflejada en la Fig. 38.

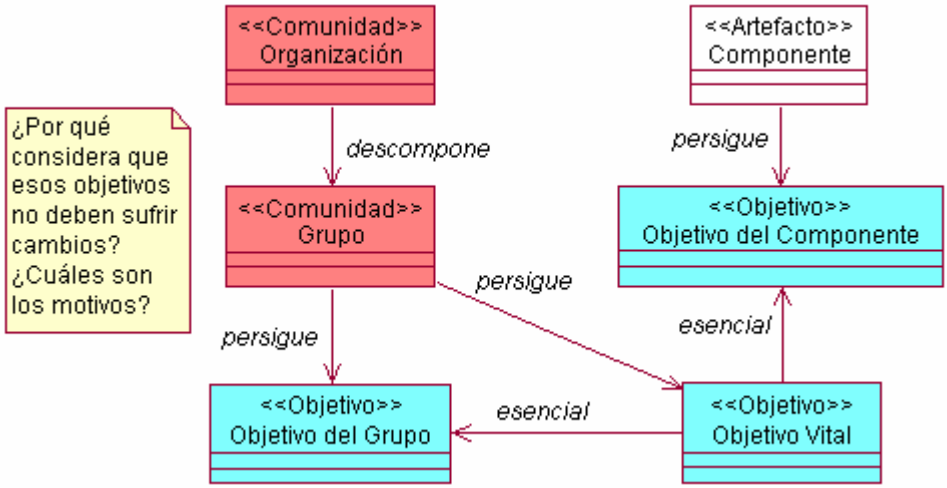


Fig. 38. Guía de Captura de Requisitos: ¿Por qué considera que esos objetivos son esenciales y no deben sufrir cambios?

El diagrama de la Fig. 38 pregunta al usuario si los objetivos que considera "vitales" lo son porque resultan *esenciales* para satisfacer otros *Objetivos del Grupo*. Estos últimos son los objetivos de alto nivel que se desea identificar para poder plantear alternativas en su satisfacción. Esta *pregunta* se contesta a partir de las respuestas a la *pregunta* 1.6.1 (ver Fig. 37) identificando los *Objetivos del Grupo*.

4.5.1.4. Pregunta 1.16.2

Una *actividad* puede involucrar a múltiples *sujetos*, cada uno jugando varios roles y persiguiendo diferentes motivos [Leontiev 1989]. En estas condiciones es normal que surjan contradicciones como conflictos entre los objetivos. Dichos conflictos se resuelven frecuentemente mediante un razonamiento cualitativo sobre los objetivos: se estima que un objetivo es más importante que otro para un actor. La *pregunta* "En caso de conflictos, ¿puede establecer prioridades entre los objetivos?" trata de capturar este conocimiento para aplicarlo a la resolución de conflictos en SMAs. Las respuestas deben ser de la forma "El *Objetivo 1* es más importante que el *Objetivo 2*". El diagrama en la Fig. 39 representa esta cuestión.

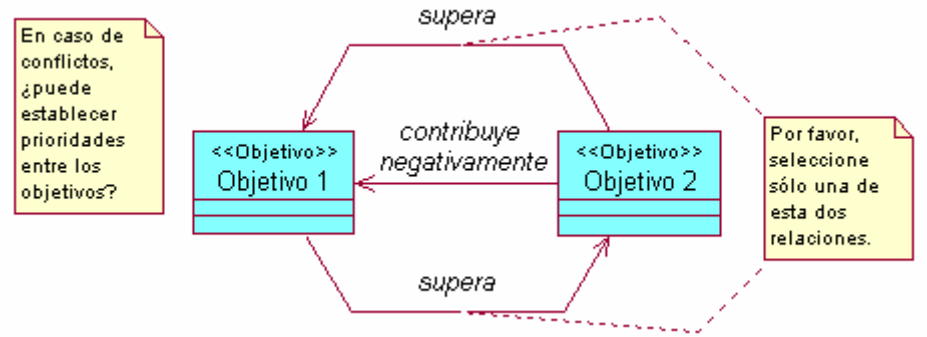


Fig. 39. Guía de Captura de Requisitos: En caso de conflictos, ¿puede establecer prioridades entre los objetivos?

El patrón de la Fig. 39 describe dos *objetivos*, i.e. *Objetivo 1* y *Objetivo 2*, que se afectan negativamente. Al usuario se le interroga sobre si es posible establecer cuál ha de ser el *objetivo* prioritario cuando se persigan ambos.

4.5.1.5. Pregunta 1.16.3

Cuando existen conflictos entre objetivos, no siempre es posible establecer una norma genérica de precedencia entre ellos como se ha hecho en la *pregunta* 1.16.2. En ocasiones, el juicio de cuál debe ser el objetivo satisfecho depende de información disponible en el contexto en el momento de la toma de decisiones. Este contexto puede hacer referencia tanto al entorno del actor como a su estado mental. La *pregunta* "En caso de conflictos, ¿existen evidencias/datos que permitan al actor elegir qué objetivo satisfacer?" pretende capturar esta información para seleccionar objetivos. En este caso se persiguen respuestas de la forma "Cuando existe la *Evidencia para Objetivo 1*, se debe intentar satisfacer el *Objetivo 1*". Su representación en UML-TA puede verse en la Fig. 40.

El diagrama en la Fig. 40 muestra dos objetivos conflictivos, i.e. *Objetivo 1* y *Objetivo 2*. La elección del objetivo a satisfacer no depende de una prioridad fija, sino de la presencia o ausencia de ciertos datos, i.e. *Evidencia para Objetivo 1*. Cuando este dato se encuentra en el entorno se "destruye" el *Objetivo 2* (por ejemplo cuando se le considera como una entidad mental del agente) o se establece una prioridad menor para él, aunque todo ello temporalmente.

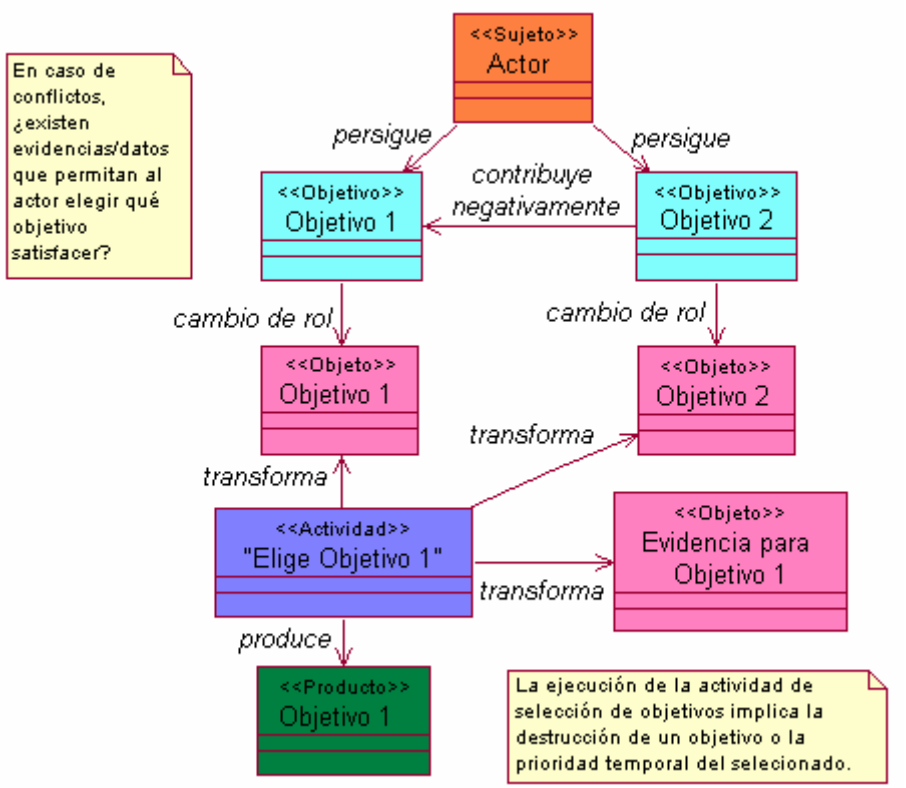


Fig. 40. Guía de Captura de Requisitos: En caso de conflictos, ¿existen evidencias/datos que permitan al actor elegir qué objetivo satisfacer?

Esta pregunta también incluye un patrón "simétrico" al de la Fig. 40 en el que una actividad *Elige Objetivo 2* selecciona el *Objetivo 2* en presencia de los datos *Evidencia para Objetivo 2*. De todos modos hay que recordar que los patrones UML-TA presentados en este capítulo no son las únicas representaciones posibles de estas *preguntas*, sino tan sólo una posible interpretación.

4.5.2. Entorno

La Tabla 13 corresponde al resumen de la GCR para SMAs en el *área* de *Entorno*. En la sección 4.2 sobre la GA original se presentó esta *área* como la relacionada con la integración de nuevos componentes en el contexto de los *sistemas de actividad* de la organización y el entorno.

VERSIÓN DE DISEÑO **Área de Entorno**	
Aspectos	**Preguntas**
1. Rol del componente en la generación de los productos	1. ¿Cuáles son las actividades en las que participará el componente? 2. ¿Existen alternativas al uso del componente en esas actividades? Es decir, ¿se pueden realizar esas actividades usando otros componentes? 3. ¿Existen actividades alternativas para crear los productos que se crearán con las actividades en que participe el componente? 4. ¿Cuáles son las ventajas e inconvenientes de estas actividades alternativas en comparación con las que realiza el componente? ✓ (ver sección 4.5.2.1)
2. Contexto de las actividades del componente	1. ¿Qué productos se necesitan para llevar a cabo esta actividad? 2. ¿La actividad consume los productos que necesita para su ejecución o, por el contrario, los preserva? 3. ¿Cuáles son los productos esperados de esta actividad?
3. Integración del componente con otros componentes	1. ¿El componente considerado está integrado con otros componentes, herramientas o materiales? 2. ¿Las actividades que involucran al componente requieren el uso de otros? 3. ¿Se necesitan los productos de este componente para generar un cierto resultado mediante la integración con otros productos?
4. Acceso a herramientas y materiales necesarios para ejecutar las acciones	1. ¿Existen restricciones en la disponibilidad de los recursos necesarios para llevar a cabo las actividades? 2. ¿Cuáles son las evidencias que señalan que estos recursos están disponibles? 3. ¿Cuáles son las evidencias que señalan que estos recursos no están disponibles? ✓ (ver sección 4.5.2.2) 4. ¿Tienen todos los actores acceso a los recursos? En caso negativo, ¿quiénes sí?, ¿quiénes no? ✓ (ver sección 4.5.2.3) 5. Los actores con acceso al recurso, ¿qué actividades pueden realizar con él? 6. Los actores con acceso al recurso, ¿pueden realizar cualquier actividad con él? En caso negativo, ¿cuáles sí? ¿cuáles no?
5. Componentes y materiales compartidos entre varios actores	1. En el caso de los componentes/herramientas/materiales necesarios para las actividades y compartidos por varios actores, ¿existe un actor que actúe como arbitro para los conflictos de uso? 2. ¿Existen normas definidas para arbitrar el uso de los componentes/herramientas/materiales compartidos?

6. Disposición espacial y organización temporal del entorno de trabajo	1. ¿Son las características del componente consistentes con las del entorno? Por ejemplo, distribución física del componente para una organización con múltiples localizaciones físicas. 2. ¿Existen localizaciones físicas superpuestas? Es decir ¿hay diferentes localizaciones de la organización que quizás hayan de compartir recursos? 3. ¿Se ejecutará el componente en distintas localizaciones físicas? 4. ¿Las actividades que el componente es capaz de realizar dependen de las localizaciones de sus elementos? 5. ¿Las actividades que el componente es capaz de realizar dependen de las localizaciones de sus usuarios? 6. ¿La disponibilidad de recursos depende de las localizaciones de sus componentes? 7. ¿La disponibilidad de recursos depende de las localizaciones de los usuarios/actores? 8. ¿El sistema debe contemplar una dimensión temporal? 9. Si el sistema contempla una dimensión temporal, ¿ésta ha de ser la misma en todo el sistema? 10. ¿Las actividades que el componente es capaz de realizar dependen del momento? 11. ¿La disponibilidad de recursos depende del momento?
7. División de la labor, incluyendo la distribución síncrona y asíncrona del trabajo entre diferentes localizacio-nes	1. ¿Qué actividades realizan qué actores? 2. ¿Qué actores participan en esta actividad? 3. ¿Existen actividades que se pueden realizar en una misma localización a la vez? 4. ¿Existen actividades que no se pueden realizar en una misma localización a la vez? 5. ¿Qué actividades han de concluir los actores antes de que empiecen otras? 6. ¿Qué actividades han de iniciar los actores antes de que empiecen otras? 7. ¿Qué actividades han de concluir los actores antes de que concluyan otras? 8. ¿Cuáles son las actividades cuya ejecución los actores pueden interrumpir para retomarla posteriormente?
8. Reglas, normas y procedimien-tos que regulan la interacción y coordinación social en relación con las	1. ¿Cuáles son los reglamentos de la organización que se aplican a esta actividad? 2. ¿Existen leyes aplicables a esta actividad? En caso afirmativo, ¿cuáles? 3. ¿Diría que existen códigos de conducta implícitos en el grupo? Por ejemplo, consultar ciertas decisiones al director/líder o permitir que algunas elecciones entre los actores se hagan por antigüedad. 4. ¿Las decisiones relativas al funcionamiento del grupo se toman previa asamblea? ¿En comité? ¿Por jerarquía? ✓

actividades	(ver sección 4.5.2.4) 5. ¿Son los conceptos y el vocabulario del sistema consistentes con los del dominio? 6. En caso de respuesta negativa a la pregunta anterior, ¿cuáles son los conceptos inconsistentes?
9. Recursos disponibles para las partes involucradas en el diseño del componente	1. ¿Cuáles son las herramientas actualmente disponibles en su organización para la construcción del componente? 2. ¿Cuáles son las herramientas actualmente disponibles en su entorno para la construcción del componente? 3. ¿Qué herramientas podría adquirir su organización de ser necesarias para el componente?

Tabla 13. *Aspectos* y *preguntas* de la Guía de Captura de Requisitos para SMAs en su versión de diseño dentro del *área* de *Entorno*.

Los siguientes apartados de esta sección explican con mayor detalle las *preguntas* marcadas con "✓" introducidas en la Tabla 13 para el *área* de *Entorno*.

4.5.2.1. Pregunta 2.1.4

El estudio del entorno del futuro componente y las actividades que éste realiza, puede mostrar que algunos de los objetivos que se pretende que cubra son ya cumplidos por otros componentes y actividades. Satisfacer esos objetivos con una nueva herramienta busca cumplir otros objetivos adicionales de la organización que las antiguas actividades no cubrían. No obstante, es preciso tener en cuenta que posiblemente la nueva solución también tendrá desventajas en comparación con la tradicional. Identificar estas ventajas y desventajas es necesario para informar correctamente al cliente y tomar las decisiones adecuadas, bien de diseño o de actuación en el SMA y su contexto. Este sería un caso de *contradicciones ternarias* [Engeström 1987] entre la nueva forma de la actividad, i.e. con el componente, y las formas antiguas.

Tras identificar las actividades alternativas con la *pregunta* 2.1.3, la *pregunta* 2.1.4 "¿Cuáles son las ventajas e inconvenientes de estas actividades alternativas en comparación con las que realiza el componente?" trata de establecer los motivos para elegir una u otra alternativa. Esta pregunta admite múltiples respuestas para las dos actividades. En esencia todas son de la forma "La *Actividad del Sistema* resulta mejor/peor/igual que la *Actividad Alternativa* para el *Objetivo*". Su representación puede verse en la Fig. 41.

La Fig. 41 muestra algunas posibilidades de comparación entre las *actividades* del componente y sus alternativas. El usuario ha de identificar los grupos y sus *objetivos*, así cómo las *actividades* que contribuyen a ellos a fin de poder comparar luego las opciones.

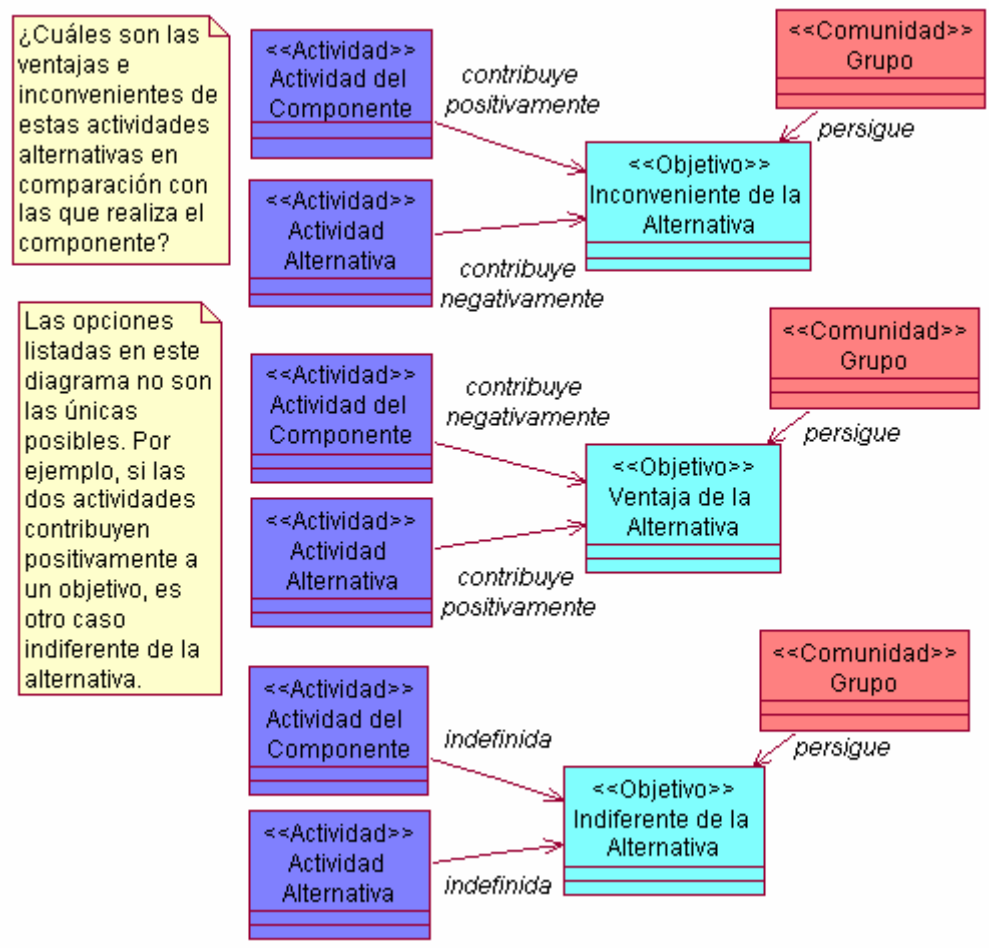


Fig. 41. Guía de Captura de Requisitos: ¿Cuáles son las ventajas e inconvenientes de estas actividades alternativas en comparación con las que realiza el componente?

4.5.2.2. Pregunta 2.4.3

Las *herramientas* juegan un papel habilitador esencial en la *actividad*: las *actividades* realizan su proceso de transformación del *objeto* mediante *herramientas* [Vygotsky 1978]. Por tanto, saber si una *actividad* puede o no llevarse a cabo depende en buena medida de la disponibilidad de sus *herramientas*.

La *pregunta* "¿Cuáles son las evidencias que señalan que estos recursos no están disponibles?" trata de determinar si existen datos que permitan establecer la disponibilidad de las *herramientas* antes de intentar llevar a cabo la *actividad*. Disponer de esta información permite mejorar la toma de decisiones acerca de cuál es la *actividad* que debe ejecutarse en un momento dado. Las respuestas son de la forma

"El *Recurso* no está disponible cuando aparece la *Evidencia de No Disponibilidad*". La representación de la *pregunta* puede verse en la Fig. 42.

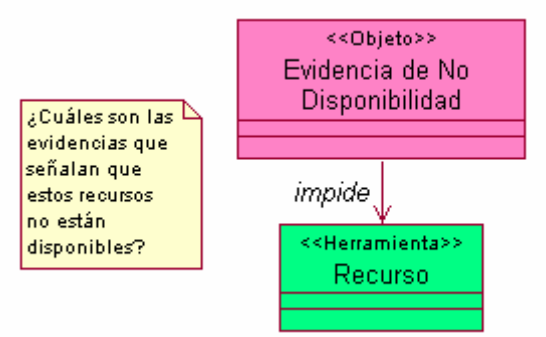


Fig. 42. Guía de Captura de Requisitos: ¿Cuáles son las evidencias que señalan que estos componentes no están disponibles?

La Fig. 42 refleja que la presencia del dato *Evidencia de No Disponibilidad* impide a la *herramienta Recurso* jugar su *rol* en la correspondiente *actividad*.

4.5.2.3. Pregunta 2.4.4

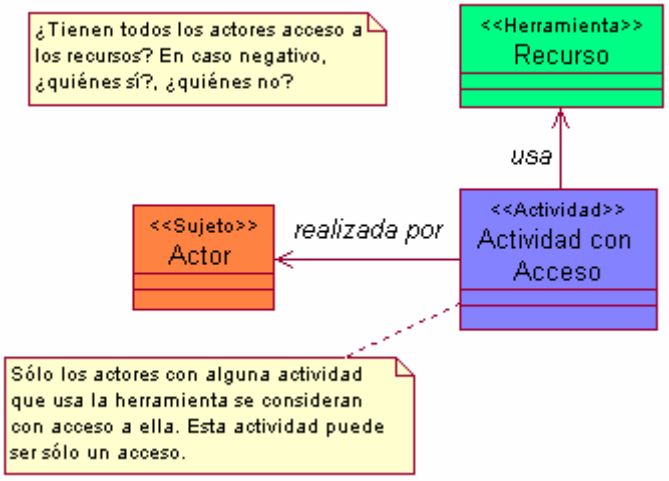


Fig. 43. Guía de Captura de Requisitos: ¿Tienen todos los actores acceso a los recursos? En caso negativo, ¿quiénes sí?, ¿quiénes no?

El funcionamiento de las organizaciones se gobierna mediante cuerpos de *reglas* que establecen normas implícitas y explícitas de comportamiento para sus miembros [Barab *et al.* 2002]. El uso de recursos es uno de los elementos que suelen estar sujetos a estas normativas. Puesto que no todos los actores tienen los mismos privilegios ni juegan los mismos roles, su acceso a los recursos no siempre es el mismo: cambia la disponibilidad y las acciones que pueden realizar con esos recursos.

La *pregunta* "¿Tienen todos los actores acceso a los recursos? En caso negativo, ¿quiénes sí?, ¿quiénes no?" pretende capturar parte de esta información. Establece una "imagen" estática de la disponibilidad de los recursos para los actores. Sus respuestas son por ejemplo "El *Actor* tiene acceso al *Recurso*" o "El *Actor* tiene acceso al *Recurso* cuando realiza la *Actividad con Acceso*". Su representación puede verse en la. Fig. 43.

En la Fig. 43 se aprecia que un *sujeto Actor* tiene acceso a una *herramienta* cuando es capaz de realizar *actividades* en las que se *usa* dicha *herramienta*. En la captura de requisitos quizás no se puedan conocer los *objetivos* concretos de la *actividad*, ya que ésta puede representar un mero acceso.

4.5.2.4. Pregunta 2.8.4

La estructura adoptada por las organizaciones para su funcionamiento depende de múltiples factores, tales como los *objetivos*, los *sujetos* participantes o el medio de intercambio empleado [Marx 1973]. Las estrategias aplicables para llevar a cabo una *actividad* dependen fuertemente de la estructura social en la cual se haya inmersa. Por ejemplo, en el caso de una jerarquía sólo es necesario convencer al líder de la necesidad de una tarea, mientras que en una asamblea es preciso argumentar hasta convencer a la mayoría de sus miembros.

Dada la importancia de establecer la estructura organizativa, se introducen *preguntas* como la presente: "¿Las decisiones relativas al funcionamiento del grupo se toman previa asamblea? ¿En comité? ¿Por jerarquía?". En este caso, se buscan respuestas que permitan localizar la organización del grupo dentro de una taxonomía de organizaciones y posibles representaciones de éstas. Para este caso la respuesta podría ser "El *Grupo* se organiza según una jerarquía" La Fig. 44 representa una versión reducida de esta *pregunta* donde sólo se considera la estructura jerárquica.

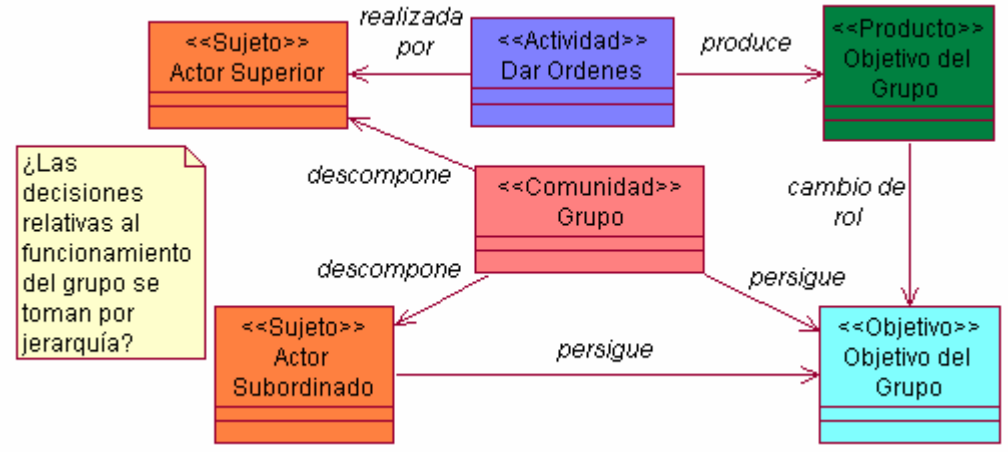


Fig. 44. Guía de Captura de Requisitos: ¿Las decisiones relativas al funcionamiento del grupo se toman por jerarquía?

La traducción de la estructura jerárquica a la Fig. 44 está inspirada en la revisión acerca de SMAs en [Sykara 1998]. Parte de este artículo se dedica a identificar algunas de las estructuras sociales que adoptan los SMAs. Entre ellas se encuentra la jerarquía definida como la organización donde "La autoridad para la toma de decisiones y el control está concentrada en un único resolutor de problemas […]. Los agentes superiores ejercen control sobre los recursos y la toma de decisiones.". Esta interpretación de qué es una jerarquía queda reflejada en la Fig. 44. Los *Actores Superiores* pueden dar órdenes a los *Actores Subordinados* que estos han de cumplir. La relación etiquetada como *cambio de rol* se emplea para indicar que el *producto* de la *actividad Dar Ordenes* se convierte en un *objetivo* para el *Actor Subordinado*.

4.5.3. Aprendizaje, cognición y articulación

El *área* de *Aprendizaje/cognición/articulación* en la GCR para SMAs queda resumida con la Tabla 14, que incluye un sumario textual de sus *aspectos* y *preguntas*. Esta *área* trata acerca de cómo los componentes proporcionan el conocimiento para realizar las *actividades* y cómo soportan la adquisición de dicho conocimiento.

VERSIÓN DE DISEÑO **Área de Aprendizaje/Cognición/Articulación**	
Aspectos	**Preguntas**
1. Observación y reflexión a través de la externaliza-ción	1. ¿Está contemplado y/o soportado el ciclo de vida completo, desde el establecimiento de los objetivos hasta la creación del producto? 2. ¿Cómo debe estar el conocimiento distribuido externamente accesible en el componente? ¿Cuál sería la información necesaria para representarlo? ✓ (ver sección 4.5.3.1) 3. ¿Debe proporcionar el componente información acerca de su estado actual? ¿Qué datos debe proporcionar para cada actividad? ✓ (ver sección 4.5.3.2)
2. Posibilidades de simulación de las acciones antes de su implementa-ción real	1. ¿Debe permitir el componente realizar simulaciones de las actividades que ejecuta? 2. ¿Estas simulaciones deben permitir el trabajo con datos reales (por supuesto, estos no serán alterados)? 3. En caso de simular una actividad, ¿han de ser los actores conscientes de que no se trata de una situación real? 4. ¿Se necesita que el componente proporcione información adicional en el caso de la simulación de actividades? En caso de respuesta afirmativa, ¿qué clase de información?
3. Partes de las acciones que se han de internalizar	1. ¿Existen acciones del componente que puedan ser integradas en versiones reducidas cuando el actor adquiere una cierta destreza? 2. ¿Sería conveniente que el componente aprendiera sobre la forma de comportarse de los actores a fin de automatizar parcialmente ciertas tareas?

4. Tiempo y esfuerzo necesario para aprender cómo usar la tecnología	1. Los usuarios del sistema, ¿son expertos en el manejo de sistemas similares? 2. Los usuarios del sistema, ¿son expertos en el dominio del problema? 3. ¿El sistema ha de contar con tutoriales para su aprendizaje? 4. ¿Ayuda el sistema a reducir la curva de aprendizaje? Es decir, ¿ayuda a evitar aprendizaje innecesario? 5. ¿Dada su experiencia y formación, encuentran los usuarios que la forma de realizar las acciones en el sistema es intuitiva?
5. Soporte para la articulación de problemas y la petición de ayuda en caso de fallo	1. ¿Proporciona el componente representaciones de los problemas en caso de fallo que puedan ayudar a encontrar la solución o formular una petición de ayuda? 2. ¿Qué clase de ayuda debería suministrar el componente en caso de problemas? 3. En caso de fallo en la actividad, ¿qué información resultaría útil para diagnosticar el fallo? 4. Dados los posibles fallos en los objetivos de la actividad, ¿qué información resultaría útil para diagnosticar cada tipo de fallo?
6. Estrategias y procedimien-tos para proporcionar ayuda a otros usuarios o individuos	1. ¿Debe proporcionar el sistema actividades colaborativas para ayuda entre personas? 2. ¿En qué situaciones se debería permitir la colaboración entre usuarios para proporcionarse ayuda? 3. ¿Qué actividades se permitirían para proporcionar esta ayuda?
7. Coordinación de las actividades individuales y de grupo a través de la externaliza-ción	1. ¿Hay representaciones externas de las actividades de los actores que puedan ser usadas como pistas por otros para coordinar sus actividades dentro del marco del grupo u organización? 2. ¿Qué información deben poder compartir los actores? 3. ¿Existen actividades en que varios actores puedan participar simultáneamente? ¿Cuáles? 4. ¿Existen actividades específicas de coordinación entre actores/componentes? 5. ¿El componente ha de proporcionar mecanismos para ceder el control de una actividad a otros actores? ✓ (ver sección 4.5.3.3) 6. ¿Debe existir la posibilidad de introspección/examen en las actividades en curso? Es decir, conocer aspectos como su punto de ejecución, los datos que están siendo usados o los recursos comprometidos. 7. ¿Se debe soportar la monitorización de tareas por otros actores distintos del que la realiza?
8. Uso de representacio	1. ¿Existen actividades realizadas simultáneamente por varios actores? ¿Cuáles?

nes compartidas para soporte del trabajo colaborativo	2. En el caso de actividades en las que participan varios actores, ¿qué información necesitan compartir para coordinarse y llevar a cabo la actividad?

Tabla 14. *Aspectos* y *preguntas* de la Guía de Captura de Requisitos para SMAs en su versión de diseño dentro del *área* de *Aprendizaje/cognición/articulación*.

Los siguientes apartados de esta sección explican con mayor detalle las *preguntas* marcadas con "✓" introducidas en la Tabla 14 para el *área* de *Aprendizaje/cognición/articulación*.

4.5.3.1. Pregunta 3.1.2

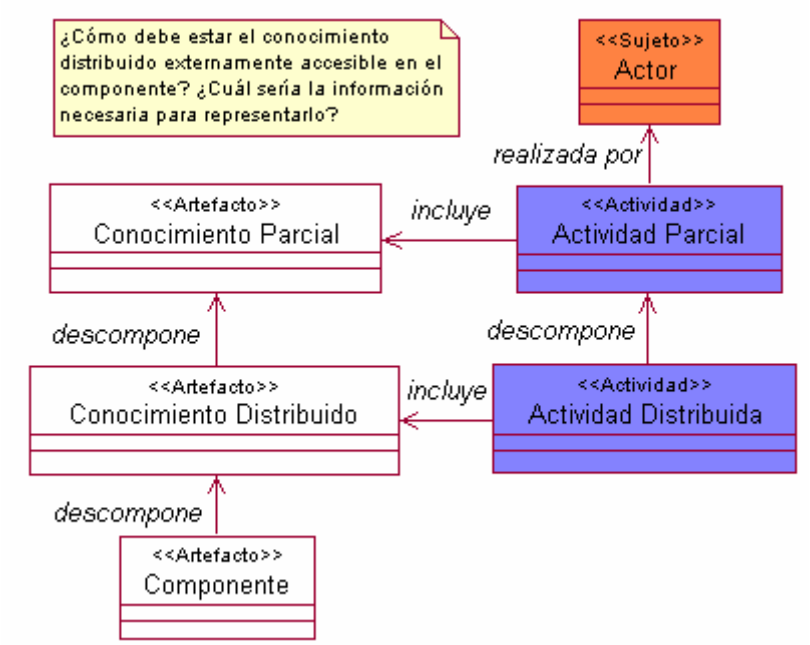


Fig. 45. Guía de Captura de Requisitos: ¿Cómo debe estar el conocimiento distribuido externamente accesible en el componente? ¿Cuál sería la información necesaria para representarlo?

La distribución temporal y espacial de las actividades tiene su reflejo en la información que éstas manejan y en cómo se hace disponible [Hutchins & Klausen 1992]. Los datos han de estar accesibles a los distintos *sujetos* en las diversas localizaciones y momentos del ciclo de vida de la *actividad*. Sin embargo, esto no significa que deba existir una disponibilidad total del conocimiento. Al contrario, cada actor ha de manejar sólo aquel conocimiento que le es relevante en cada momento. Parte de este conocimiento es capturado mediante la *pregunta* "¿Cómo debe estar el

conocimiento distribuido externamente accesible en el componente? ¿Cuál sería la información necesaria para representarlo?" cuya representación aparece en la Fig. 45. Las respuestas serían de la forma "Cuando el *Actor* realiza la *Actividad Parcial* sólo debe tener disponible el *Conocimiento Parcial* que forma parte del *Conocimiento Distribuido*".

La Fig. 45 refleja que el conocimiento necesario para realizar la *Actividad Distribuida* es el conjunto de información representado por el *artefacto Conocimiento Distribuido*. El *sujeto Actor* no necesita tener accesible toda la información de *Conocimiento Distribuido*, sino sólo la que aparece en *Conocimiento Parcial*, que es la necesaria para llevar a cabo su *Actividad Parcial*.

4.5.3.2. Pregunta 3.1.3

El funcionamiento de un componente puede ser objeto de introspección más allá de los resultados de sus actividades. Este es por ejemplo, el caso de la información necesaria para diagnosticar fallos. Por ello, al abordar el diseño es preciso considerar cómo informa el componente acerca de su estado en un momento dado [Bertelsen 1996, Bødker *et al.* 1998]. La *pregunta* "¿Debe proporcionar el componente información acerca de su estado actual? ¿Qué datos debe proporcionar para cada actividad?" trata de capturar qué clase de información puede ser necesaria para las labores de seguimiento del componente. Las respuestas a esta pregunta deben ser similares a "La *Actividad del Componente* debe suministrar como datos acerca de su funcionamiento la *Información sobre la Actividad del Componente*" Su representación gráfica queda recogida en la Fig. 46.

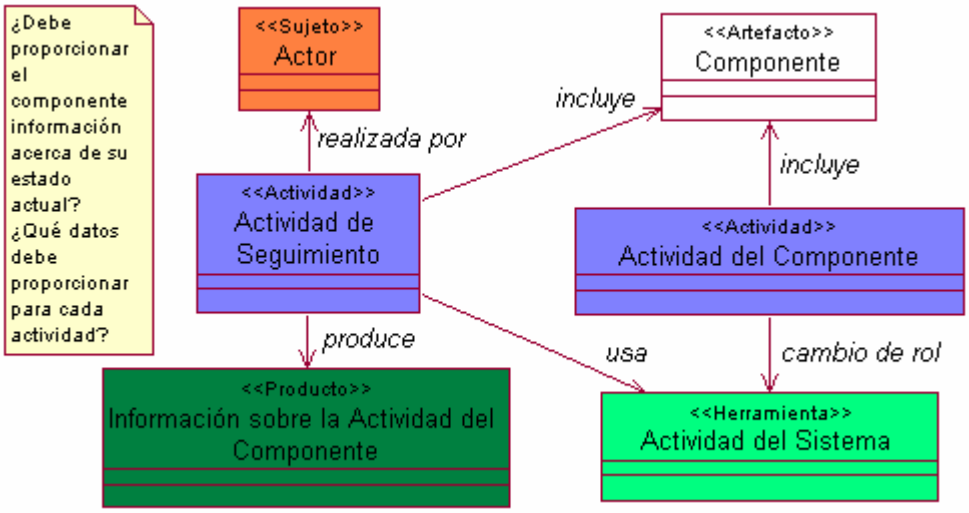


Fig. 46. Guía de Captura de Requisitos: ¿Debe proporcionar el componente información acerca de su estado actual? ¿Qué datos debe proporcionar para cada actividad?

En la Fig. 46, el *sujeto Actor* es capaz de realizar *actividades* de supervisión del componente. Estas *actividades* utilizan al propio componente para obtener información acerca de su estado en función de las *actividades* que está realizando.

Esta información de estado es representada por el *producto Información sobre la Actividad del Componente*.

4.5.3.3. Pregunta 3.7.5

El diseño de sistemas destinados a ser usados de forma colaborativa entre múltiples actores ha de contemplar aspectos tales como la compartición de los datos, el arbitrio, la resolución de conflictos o el control de recursos y actividades [Fjeld *et al.* 2002]. Así, las *preguntas* 3.1.2 o 3.7.2 incidían en cuestiones de datos mientras que ésta trata sobre el control de las actividades compartidas. La *pregunta* "¿El componente ha de proporcionar mecanismos para ceder el control de una actividad a otros actores?" refleja si el rol de director y ejecutor de una actividad puede o no cambiar durante la ejecución de la misma. Las respuestas serían "El control de la *Actividad Compartida* puede pasar durante su ejecución del *Actor 1* al *Actor 2*". La Fig. 47 refleja esta cuestión.

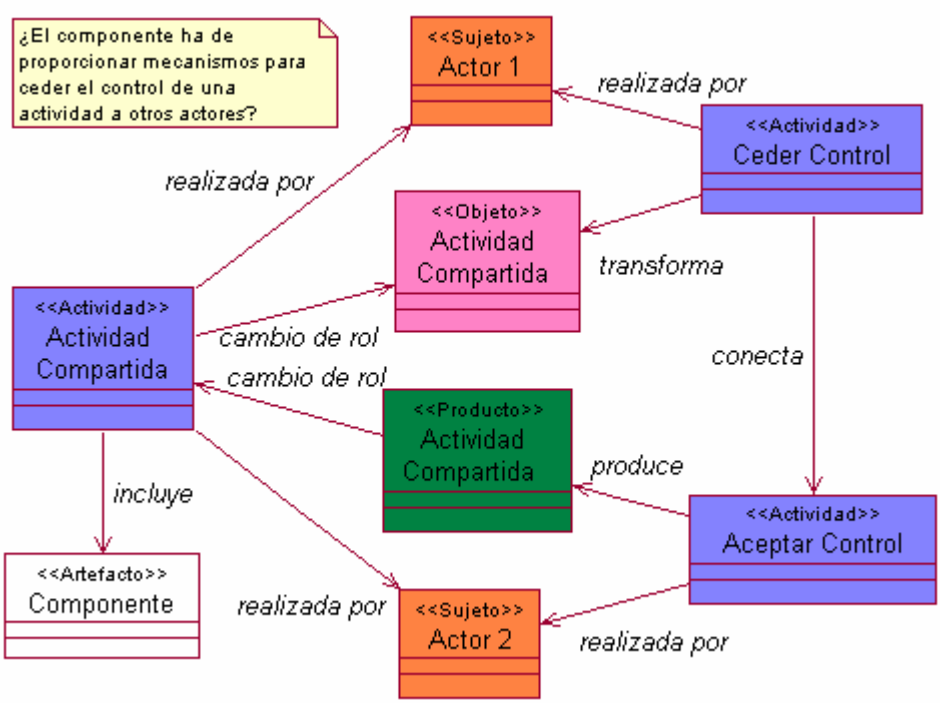


Fig. 47. Guía de Captura de Requisitos: ¿El componente ha de proporcionar mecanismos para ceder el control de una actividad a otros actores?

La Fig. 47 refleja la cesión del control de la ejecución de la *Actividad Compartida* del *Actor 1* al *Actor 2*. El control cambia mediante las *actividades Ceder Control* y *Aceptar Control*, que usan *actividades* como *objetos* cuyo control ha de cambiar. Inicialmente el *Actor 1* está realizando la *Actividad Compartida*. La *actividad Ceder Control* toma como *objeto* la *Actividad Compartida* y la transforma en un *producto*

que a su vez toma *Aceptar Control* como *objeto*. Este paso de la *Actividad Compartida* entre las *actividades Ceder Control* y *Aceptar Control* está abreviado mediante la relación *conecta*. Es en esta relación *conecta* donde tiene lugar la cesión del control del *Actor 1* al *Actor 2*. Posteriormente, la *actividad Aceptar Control* genera el *producto Actividad Compartida*, que representa la *actividad* controlada por el *Actor 2*.

4.5.4. Desarrollo

Los componentes de la GCR para SMAs en el *área* de *Desarrollo* están recogidos en la Tabla 15. Esta tabla incluye los *aspectos* y *preguntas* del *área* y muestra los índices necesarios para la posterior localización de las *preguntas*. Como se señaló previamente en la sección 4.2 a propósito de la GA, esta *área* trata sobre los cambios que la introducción del nuevo componente produce en la forma futura de las actividades.

VERSIÓN DE DISEÑO **Área de Desarrollo**	
Aspectos	**Preguntas**
1. Cambios anticipados en las acciones después de que se implemente el componente	1. ¿El uso del nuevo componente generará nuevas necesidades o expectativas para los actores? ¿Cuáles diría que serán esos nuevos objetivos? 2. ¿Cree que las actividades presentes se realizarán de otra manera a medida que los actores adquieran mayor dominio del componente? ¿Cómo? ✓ (ver sección 4.5.4.1) 3. ¿Qué nuevas actividades piensa que surgirán con el uso del componente? 4. ¿Qué actividades cree que desaparecerán con el uso del componente? ✓ (ver sección 4.5.4.2)
2. Uso de herramientas en diferentes etapas del ciclo de vida de las acciones	1. Las herramientas usadas en las diferentes etapas de la actividad, ¿corresponden al mismo momento evolutivo? Es decir, ¿qué herramientas considera más avanzadas? 2. ¿Qué herramientas cree que habría que sustituir por otras? ¿Por cuáles? 3. ¿Qué beneficios reportaría esta sustitución?
3. Transforma-ción de las presentes actividades en las futuras con el soporte del sistema	1. ¿Cómo piensa que debe ayudar el sistema a la transición desde las actividades actuales a las previstas en el futuro? 2. ¿Serán necesarios cursos comparando las "viejas" y "nuevas" formas de realizar las actividades?
4. Historia de la implementa-	1. ¿Requiere el componente una gran inversión en tiempo y esfuerzo para aprender cómo usarlo?

	ción de las tecnologías objetivo para el soporte de las acciones	2. ¿Cuáles son las consecuencias de la implementación de la tecnología sobre las acciones objetivo? ¿Aparecieron realmente los beneficios esperados? 3. ¿El componente muestra beneficios crecientes o decrecientes con su uso? 4. ¿Las actitudes de los actores hacia el componente se están volviendo más o menos positivas? 5. ¿Existen efectos colaterales positivos o negativos fruto del uso del componente?
5.	Cambios anticipados en los requisitos del componente	1. ¿Tenían los usuarios suficiente experiencia con el componente en el momento de la evaluación? 2. ¿Qué objetivos de la organización cree que cambiarán cuando esté en uso el componente? 3. ¿Qué objetivo de los usuarios cree que cambiarán cuando esté en uso el componente?

Tabla 15. *Aspectos* y *preguntas* de la Guía de Captura de Requisitos para SMAs en su versión de diseño dentro del *área* de *Desarrollo*.

Los siguientes apartados de esta sección explican con mayor detalle las *preguntas* marcadas con "✓" introducidas en la Tabla 15 para el *área* de *Desarrollo*.

4.5.4.1. Pregunta 4.1.2

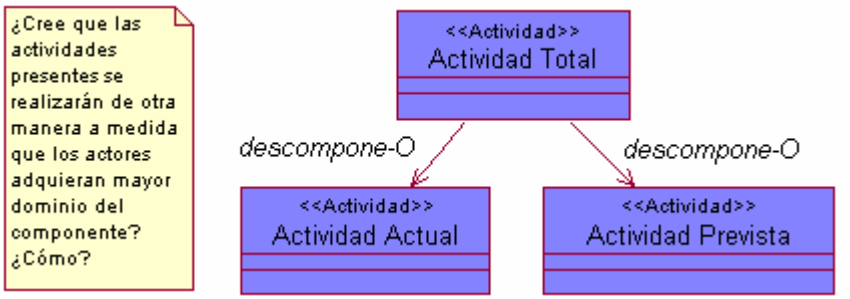


Fig. 48. Guía de Captura de Requisitos: ¿Cree que las actividades presentes se realizarán de otra manera a medida que los usuarios adquieran mayor dominio del componente? ¿Cómo?

Como ya se indicó en la introducción a la TA (ver sección 3.2), uno de los aspectos fundamentales del estudio de la *actividad* es su análisis histórico, su desarrollo en el tiempo [Leontiev 1978]. La introducción de nuevas *herramientas* y *objetos* es uno de los factores capaces de originar la evolución de una *actividad*. El desarrollo de SMAs no es una excepción. El diseño de sistemas es una labor donde el producto no queda concluido en su desarrollo sino sólo en su utilización, realizándose ciclos de desarrollo y uso que se realimentan mutuamente [Tuikka 2002]. Esta *pregunta* trata de anticipar algunos de estos cambios investigando las posibilidades que abre el componente a los actores. La representación de la *pregunta* "¿Cree que las actividades presentes se realizarán de otra manera a medida que los actores adquieran mayor

dominio del componente? ¿Cómo?" se ve en la Fig. 48. La respuesta esperada es una descripción de una nueva actividad, "La *Actividad Actual* será posiblemente sustituida en el futuro por la *Actividad Prevista*".

La Fig. 48 sólo permite establecer la relación entre la forma actual de las *actividades* y la prevista en el futuro. El análisis de aspectos tales como la forma en que se realizarán estas *actividades* y cuáles serán sus *objetivos* se hace a través de otras *preguntas* de la GCR.

4.5.4.2. Pregunta 4.1.4

Siguiendo la línea del análisis histórico de la *actividad* a la que corresponde también la *pregunta* 4.1.2, cabe preguntarse cuáles de las *actividades* actuales desaparecerán en el futuro. El sentido de este estudio es planificar la evolución de las características del componente, tales como las tareas que necesitarán ser migradas a nuevas formas, la forma en que se soportarán otras características cuando estas actividades desaparezcan o las características del componente que se volverán innecesarias. El punto de partida de este estudio es la *pregunta* "¿Qué actividades cree que desaparecerán con el uso del componente?" representada en la Fig. 49. La respuesta sería como la siguiente "Con el uso del *Componente* se dejará de realizar la actual *Actividad del Componente*".

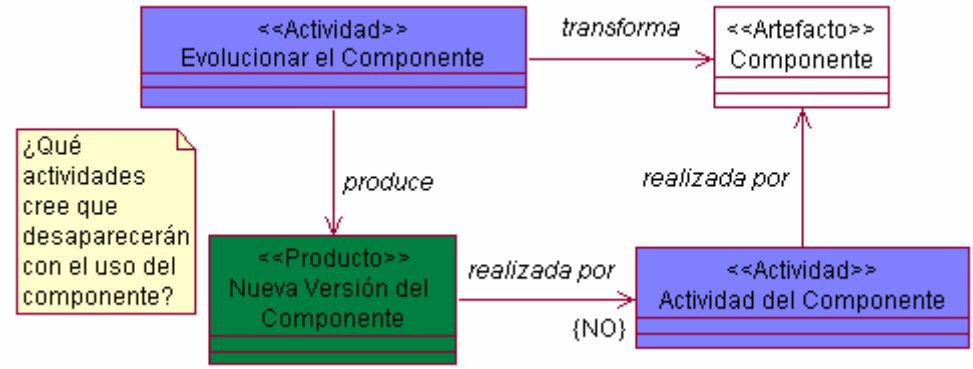


Fig. 49. Guía de Captura de requisitos: ¿Qué actividades cree que desaparecerán con el uso del componente?

En la Fig. 49 se representa la transición desde aquellas versiones del componente que soportan la *Actividad del Componente* a las nuevas versiones que no la incluyen.

4.6. Ejemplo de uso de la GCR

Para mostrar el uso de la GCR para SMAs se ha seleccionado un caso de estudio con equipos de agentes en el entorno *Robocode* [IBM 2002]. *Robocode* es un entorno para el aprendizaje de programación en Java. Simula batallas de tanques mediante robots que actúan con primitivas predefinidas (tales como avanzar, girar a la izquierda y abrir fuego) y perciben la situación con sensores de posición, rádares y la detección

de algunos eventos en el entorno. Los desarrolladores tienen que programar el comportamiento de los tanques para destruir a sus enemigos y sobrevivir a la batalla. El ganador de la batalla es el equipo con los últimos tanques supervivientes. El caso de estudio propuesto considera la colaboración entre tanques en este marco. La metodología de agentes objetivo a la cual se traducen los requisitos es INGENIAS [Pavón & Gómez-Sanz 2003]. La especificación completa de este problema puede encontrarse en http://ingenias.sourceforge.net.

Como ya se ha indicado, el caso de estudio contempla ejércitos de tanques colaborativos, i.e. *Ejércitos*, compuestos por escuadrones, i.e. *Escuadrones*. En cada escuadrón los agentes pueden jugar uno de dos posibles roles:

- *Soldados*. Son los miembros básicos del escuadrón. Tratan de sobrevivir a la batalla mientras destruyen a sus enemigos y cumplen las órdenes de sus superiores.
- *Capitanes*. Son los líderes de los escuadrones. Muestran el comportamiento básico de los soldados y además comandan sus tropas. Un capitán planea una estrategia para sus soldados, se la comunica, sigue el curso de la batalla y hace modificaciones en la planificación si es necesario.

Un *escuadrón* sólo puede tener un *capitán*, pero puede incluir varios *soldados*. La situación descrita queda resumida en la Fig. 50 con INGENIAS.

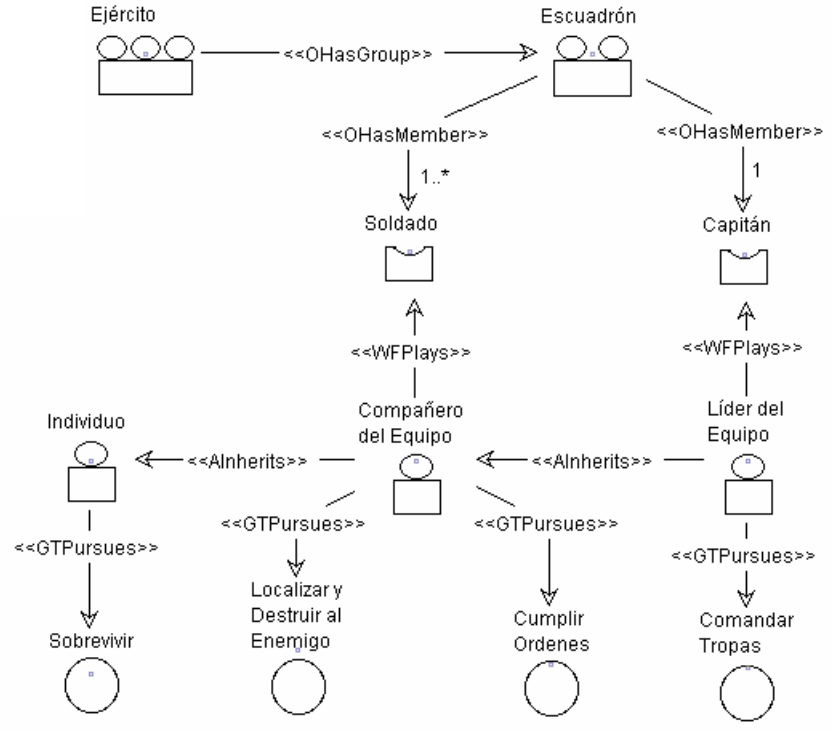


Fig. 50. Especificación de un ejército de *Robocode*.

Los elementos involucrados en la Fig. 50 son: *Ejército*, que es una organización; *Escuadrón*, que es un grupo de tanques; *Individuo*, *Compañero del Equipo* y *Líder del Equipo*, que son agentes; *Soldado* y *Capitán*, que son roles; los círculos representan objetivos. Un *Líder del Equipo* tiene el objetivo de *Comandar Tropas*. Cuando ejecuta las actividades asociadas a este objetivo, genera y comunica ordenes a sus tropas. Un *Compañero del Equipo* de estas tropas trata de satisfacer el objetivo de *Cumplir Ordenes*. Por tanto, trata de obedecer las ordenes de su *Líder del Equipo*. Un objetivo común a todos los agentes, heredado del *Individuo*, es *Sobrevivir*. Este objetivo impulsa a los agentes a preservar su vida cualquiera que sea la situación.

Una posible estrategia para el escuadrón es tener exploradores que reconozcan el terreno para determinar la posición de las tropas enemigas. Con esta información el líder puede planear el ataque de una forma muy precisa.

En este punto de estudio del sistema, el problema es cómo se ha de plasmar esta estrategia en los requisitos. Para resolverlo se utiliza la GCR siguiendo el proceso de la Fig. 35. Se comienza seleccionando las *preguntas* de la GCR a responder siguiendo la estructura jerárquica de *áreas*, *aspectos* y *preguntas*. Aunque existen varios puntos posibles de partida, se opta por comenzar en el *área* de *Medios/fines* para saber cuáles pueden ser las tareas/*actividades* con las que se plantea satisfacer los *objetivos*. Dentro de esta *área* encontramos el *aspecto* 8, *Operacionalización de los objetivos*, que incluye cuestiones para introducir las *actividades* que satisfacen, contribuyen, perjudican o impiden alcanzar los *objetivos*. La *pregunta* 1.8.1 introduce la nueva *actividad Localizar Enemigos*. Después, en la misma *área*, la *pregunta* 1.1.6 del *aspecto* 1 sobre *Actores del Nuevo Sistema*, permite asociar la *actividad* al *sujeto Soldado*. Por último, las *preguntas* del *aspecto* 1.11 sobre *Criterios para la satisfacción o fallo de los objetivos*, llevan a introducir el *producto* de la nueva *actividad*. El resultado de plantear estas *preguntas* puede verse en la Fig. 51.

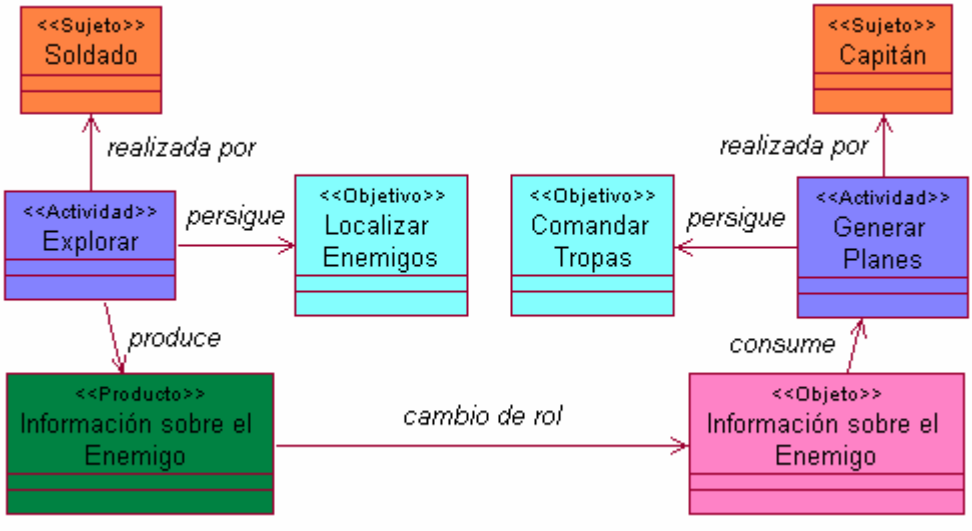


Fig. 51. *Actividades*, *objetivos* y *objetos* involucrados en la estrategia de explorar.

Los roles de INGENIAS son representados como *sujetos* de la TA. El usuario ha introducido varios elementos nuevos en los requisitos. El *objetivo Localizar Enemigos* surge asociado a la *actividad* de *Explorar* que realiza la recogida de información sobre el enemigo. Esta información es representada por el *artefacto Información sobre el Enemigo* que es usado por el *Capitán* para generar los planes.

Tras introducir estos nuevos elementos en los requisitos se hace necesario determinar cómo se ven afectados los elementos previamente existentes, la mejor forma de plantear su operacionalización o el refinamiento de los elementos.

Uno de los primeros puntos de interés es conocer si existen conflictos entre los nuevos *objetivos* y los ya existentes. Para ello se plantea a los usuarios la *pregunta* 1.13.6 "¿Cumplir los objetivos del componente dificulta la satisfacción de otros objetivos de los actores?". La representación con UML-TA de esta *pregunta* es muy similar a la mostrada para la *pregunta* 1.2.3 "¿Existen inconvenientes para la organización o partes de ella en la construcción del componente?" (ver sección 4.5.1.1). La respuesta proporcionada por el usuario sobre la plantilla se muestra en la Fig. 52.

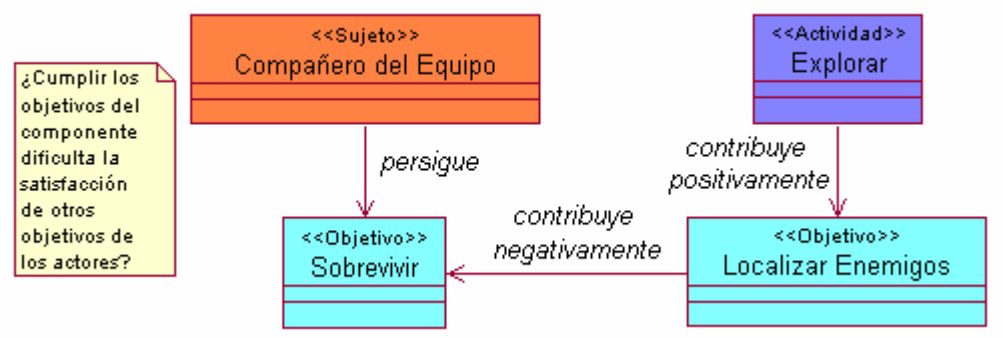


Fig. 52. *Pregunta* 1.13.6 respondida para la *actividad Explorar* en Robocode.

En la Fig. 52 el usuario realiza sustituciones sobre la plantilla de la *pregunta* y emplea algunas de las equivalencias en el lenguaje de la TA introducidas en la sección 3.4. El *Compañero del Equipo* persigue el *objetivo Sobrevivir* que hereda del *Individuo*. El *artefacto Componente* del marco de la *pregunta* es instanciado a la *actividad Explorar* y se añade también el *objetivo* que persigue. Usando las equivalencias vistas en la sección 3.4, cuando una *actividad persigue* un *objetivo* se puede considerar que dicha *actividad* al menos *contribuye positivamente* a dicho *objetivo*.

La contradicción entre objetivos que surge de esta respuesta se deduce también gracias a las equivalencias de la sección 3.4. Un *sujeto* que juega un rol pasa a perseguir todos los *objetivos* de dicho rol y realizar todas sus *actividades*. En este caso el *Compañero del Equipo* juega el rol de *Soldado* por lo que realiza la *actividad* de *Explorar* asociada a este rol. Además, cuando un *sujeto* realiza una *actividad* es porque persigue el *objetivo* de ésta. Por tanto, el *Compañero del Equipo* persigue el objetivo *Localizar Enemigos*. De este modo, la respuesta proporcionada a la *pregunta* nos muestra una contradicción entre los *objetivos* del *Compañero del Equipo* que es representada en la Fig. 53.

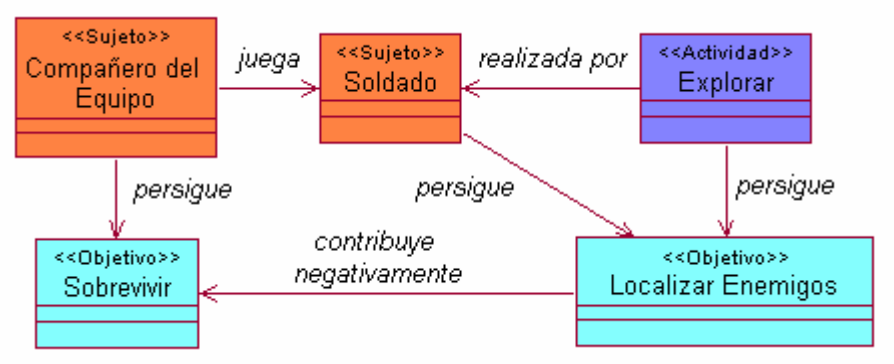


Fig. 53. Contradicción entre los objetivos del *Compañero del Equipo* en la *actividad Explorar*.

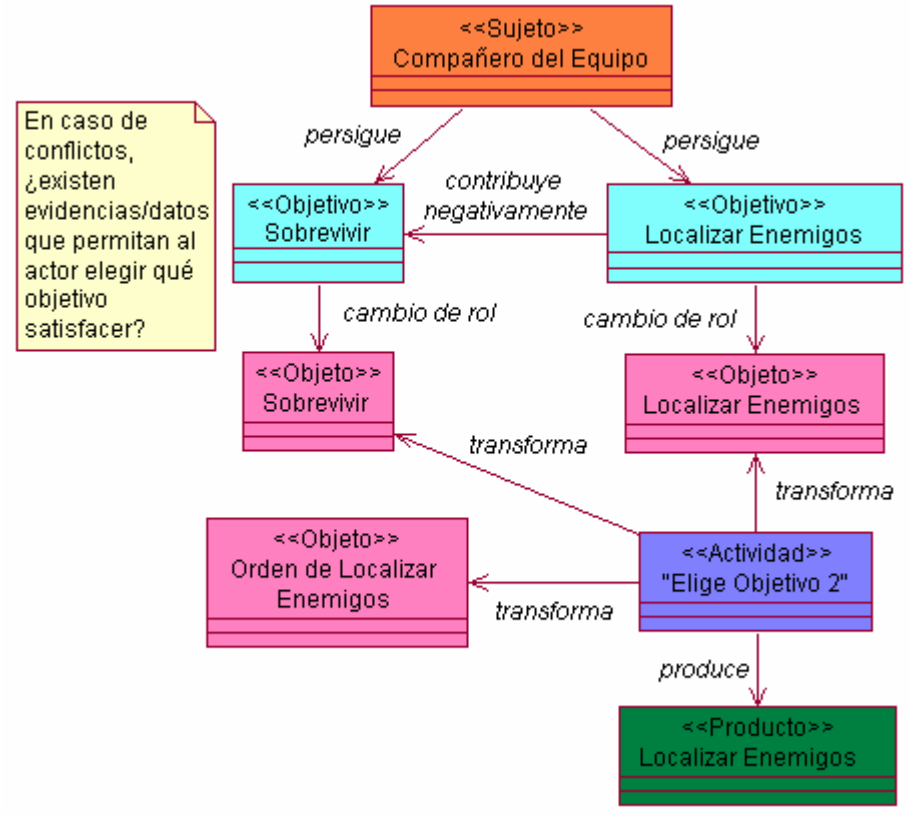


Fig. 54. *Pregunta* 1.16.3 respondida para la *actividad Explorar* en Robocode.

La detección de esta clase de conflictos se realiza mediante las contradicciones que se definen en el Capítulo 5. Por ello, la revisión en profundidad de la definición de las contradicciones y su detección se pospone al capítulo indicado.

Una vez detectado el conflicto se ha de plantear su resolución. Una vez más se sigue el proceso de la Fig. 35. Primero se localiza una *pregunta* potencialmente relevante para la información que se desea obtener y a continuación se trata de contestarla.

La GCR ofrece dentro del *área* de *Medios/fines* el *aspecto* 16 *Resolución de conflictos entre objetivos*. En este *aspecto*, la *pregunta* 1.16.2 "En caso de conflictos, ¿puede establecer prioridades entre los objetivos?" ofrece la solución más directa de la contradicción. El usuario mostró que en este caso el *sujeto Compañero del Equipo* no debía realizar siempre la *actividad* de *Explorar*, por lo que el *objetivo Localizar Enemigos* no era prioritario a *Sobrevivir* y el conflicto persistía.

Otra *pregunta* dentro del mismo *aspecto* anterior es la 1.16.3 "En caso de conflictos, ¿existen evidencias/datos que permitan al actor elegir qué objetivo satisfacer?". La respuesta muestra que la prioridad entre los *objetivos* depende de la interacción del *Compañero del Equipo* con el *Líder del Equipo*. Como se puede ver en la Fig. 54, *Localizar Enemigos* es un *objetivo* prioritario para el *sujeto Compañero del Equipo* cuando existe un *objetivo Orden de Localizar Enemigos*, que ha de ser generado por su superior en el escuadrón.

La respuesta acerca de la orden al *Compañero del Equipo* corresponde a una clase de *reglas* para las *actividades* del ejército relacionadas con la estructura jerárquica de los mismos. En el *área* de *Entorno* encontramos un *aspecto* relativo a *Reglas, normas y procedimientos que regulan la interacción y coordinación social en relación con las actividades*, que incluye la *pregunta* 2.8.4 "¿Las decisiones relativas al funcionamiento del grupo se toman previa asamblea? ¿En comité? ¿Por jerarquía?". Esta *pregunta* nos lleva a afirmar que el ejército de tanques tiene una estructura jerárquica y que el *objetivo Orden de Localizar Enemigos* ha de ser cumplido por venir de un superior. A este nuevo *objetivo Orden de Localizar Enemigos* sí se le podría aplicar la pregunta 1.16.2 y establecer que tiene prioridad sobre los demás *objetivos* del *sujeto Compañero del Equipo*, y en concreto sobre *Sobrevivir*.

4.7. Conclusiones

En este capítulo se ha introducido la Guía de Captura de Requisitos (GCR) como método basado en la TA para obtener los requisitos sociales de SMAs. La GCR ha surgido de la redefinición de la Guía de Actividades, una técnica analítica para HCI basada en la TA.

Las principales aportaciones de este capítulo para la captura de requisitos sobre las *propiedades sociales* de SMAs son:

- *GCR*. Es una técnica de ayuda a la captura de requisitos.
- *Método para adaptar la GCR a necesidades específicas*. El proceso de redefinición de la GCR da las pautas para incluir nuevos aspectos y preguntas que atiendan a necesidades específicas de proyectos o dominios. Además, la estructura jerárquica de la GCR, heredada de la GA, ayuda a localizar los puntos donde considerar la nueva información.
- *Método de uso de la GCR en procesos de la ISOA ayudados por la TA*. El método de utilización de la GCR se enmarca en un proceso más amplio basado en

la TA. Este capítulo ha mostrado en el ejemplo el uso de la GCR para obtener información. Las técnicas sobre uso de contradicciones con TA del capítulo siguiente son el complemento para realizar verificaciones tempranas sobre los requisitos del sistema.

A propósito de la creación de la GCR se debe decir que esta técnica no pretende capturar todos los requisitos de un SMA en todas las circunstancias con una única formulación. El proceso de modificación incluye el estudio de la información que se necesita capturar, lo que puede llevar a alterar los *aspectos* y *preguntas* considerados en la GCR. En todo caso, la guía será utilizada en combinación con otras herramientas de la Ingeniería de Requisitos con las que se complementará. Esta combinación se puede hacer sobre las especificaciones traducidas a UML-TA o sobre las especificaciones y procesos de la metodología de la ISOA.

El ejemplo de la sección anterior ha mostrado los principales beneficios del uso de la GCR en la ISOA:

- *Recordatorio de las características a considerar en el SMA*. La GCR apuntaba aspectos que se debían recoger para tener una visión completa al analizar las tareas u objetivos del sistema. Éste fue el caso de las preguntas sobre posibles contradicciones entre los objetivos.
- *Sugerencia en las decisiones de análisis*. La GCR ha proporcionado indicaciones sobre alternativas en el estudio en aquellos puntos donde podían existir dificultades para determinar cómo seguir. Por ejemplo, ésta fue la situación al establecer la organización jerárquica del ejército.
- *Facilidad de uso*. La estructura de la GCR con áreas, aspectos y preguntas facilita su uso. La jerarquía permite dirigirse fácilmente a aquellas partes de la GCR relevantes para la información que se está considerando y hallar las preguntas a realizar.
- *Guía al representar los requisitos en lenguajes de diseño*. Las plantillas asociadas a las preguntas de la GCR mostraban la forma de introducir la información en UML-TA. De este modo ayudaban a fijar la información en lenguaje natural de los clientes en otro lenguaje más apropiado para el desarrollo.

Una ventaja adicional de la GCR, que no ha podido ser apreciada en el ejemplo, es que está concebida para ser usada como ayuda a la discusión entre clientes y desarrolladores. Está orientación se plasma en una forma textual sencilla con preguntas, pero también en la estructura jerárquica que facilita su uso.

La propuesta con la GCR también tiene algunas limitaciones:

- *Fragmentación de la información*. Los patrones UML-TA de las preguntas dan lugar a un gran número de diagramas con pocos elementos en general. En ocasiones esto dificulta contemplar globalmente algunas informaciones. Este problema puede ser mitigado mediante el uso de propiedades descriptivas adicionales que ayuden a presentar la estructura del SMA a un nivel más alto.
- *Traducción de la forma textual a UML-TA*. El proceso de redefinición de la GCR incluye una de las tareas más difíciles de los procesos de este capítulo, la traducción entre las dos representaciones de las preguntas. Es el punto donde los conceptos de la TA han de ser identificados sobre la forma textual de la pregunta, que es ambigua en esencia, y representados con UML-TA. Es decir, en buena

medida se trata de un paso sujeto a la interpretación del que lo realiza. Este problema no es particular de la GCR, ya que se plantea en toda la Ingeniería del Software cuando se trata de trasladar el lenguaje natural a un lenguaje de diseño.

Para concluir indicar que, por supuesto, son posibles otras alternativas a los requisitos capturados con la GCR. De todas formas, hay que señalar que el papel de la GCR no es decidir cómo ha de ser el SMA, sino sólo proporcionar sugerencias a los desarrolladores y usuarios sobre cómo puede ser.

Capítulo 5. Manejo de contradicciones

Desde la perspectiva de la TA, el proceso de desarrollo de un SMA consiste en la permanente evolución de sus especificaciones para solventar las contradicciones que surgen en ellas. Este capítulo comienza profundizando en la visión de las contradicciones como directoras del desarrollo, realizando un repaso de las herramientas conceptuales específicas de la TA para las contradicciones. Tener los conceptos para manejar las contradicciones no basta para aplicarlas en la ISOA, también hacen falta procesos para usarlas en desarrollos de SMAs y conocimiento acerca de contradicciones concretas. En este capítulo también se describen métodos de definición de contradicciones a partir de fuentes textuales y de detección y solución de contradicciones concretas en especificaciones. El último elemento necesario para completar la propuesta es disponer de repositorios de contradicciones reutilizables para cualquier desarrollo. Este capítulo presenta una recopilación de algunas contradicciones obtenidas de la investigación en TA. La aplicación conjunta de conceptos, procesos y conocimiento es mostrada mediante un caso de estudio. Para terminar se discuten las ventajas y limitaciones de la propuesta sobre contradicciones.

5.1. Introducción

Describir un SMA es una tarea difícil que requiere un considerable esfuerzo. El gran número de elementos y aspectos a describir en sistemas complejos conlleva la creación de una gran cantidad de diagramas interrelacionados (como en INGENIAS [Pavón & Gómez-Sanz 2003] o en MAS-CommonKADS [Iglesias *et al.* 1998]) o de unos pocos diagramas de gran tamaño (como en Tropos [Castro *et al.* 2001] o GAIA [Wooldridge *et al.* 2000]).

A medida que el número de diagramas y elementos que aparecen en la especificación del SMA crece, también se incrementa la posibilidad de que aparezcan *contradicciones*. El término *contradicción* se emplea aquí para significar una inconsistencia, un punto oscuro en la especificación o un malentendido en los modelos [Pressman 2000].

Este capítulo plantea un método de comprobación de especificaciones orientado sobre todo a tratar las *contradicciones* relacionadas con las *propiedades sociales* [Constantine 1995] de un SMA. Se asume que, independientemente de la metodología elegida y de lo cuidadoso que sea el desarrollador, aparecerán contradicciones durante la construcción del sistema. El objetivo de la comprobación será facilitar la detección de estas contradicciones y usarlas como guía en el proceso de desarrollo.

Como ya se vio en el estado del arte (ver Capítulo 2), para identificar estas contradicciones y resolverlas, las metodologías de la ISOA han optado por diferentes técnicas. Los principales problemas que se identificaron para estas propuestas eran la falta de guía para localizar las contradicciones y resolverlas en procesos de verificación, la frecuente inadecuación de los lenguajes para tratar con contradicciones en los aspectos intencionales y sociales, la visión fragmentada de las *propiedades sociales* y la ausencia de conocimiento específico sobre propiedades de interés.

Para tratar estos problemas se argumentó en el Capítulo 1 la adecuación de la TA y en el Capítulo 3 se presentó la infraestructura básica para usar la TA sobre la ISOA. En el presente capítulo se completa la propuesta con el método de tratamiento de contradicciones, inspirado por los ciclos de estudio de la TA en las ciencias sociales.

El método planteado comienza identificando el modelo que indica cómo debería ser un SMA. Este modelo está basado en el concepto de *contradicción* de la TA. Se trata de lograr un SMA donde no existan estas *contradicciones* o se haya determinado que las que aparecen son inevitables por tratarse de una propiedad inherente del sistema. Estas *contradicciones* sociales presentes en la TA se describen en términos de la representación para *propiedades sociales* de la sección 3.6. Concretamente, estas *contradicciones* sociales son un caso de *propiedades anti-patrón* (ver sección 3.6).

Los desarrolladores estudian las *propiedades anti-patrón* para tratar las contradicciones en sus modelos. Se buscan instancias de los patrones de detección de las propiedades dentro de un proceso de verificación de SMAs. Si aparece una instancia en las especificaciones, se ha localizado una posible contradicción. Las correspondencias entre el patrón de detección y las especificaciones dan una explicación estructural y gráfica de la contradicción en términos de elementos de las propias especificaciones del SMA. La correspondencia permite además rellenar los patrones de solución de las *propiedades sociales* a partir de los modelos. Estos patrones de solución instanciados sugieren posibles modificaciones de las especificaciones para eliminar la contradicción hallada.

Este tratamiento de las *contradicciones* de la TA constituye una técnica independiente de la metodología de SMAs que se emplee. Gracias al uso de las correspondencias y el proceso de traducción introducidos en la sección 3.5, las contradicciones a comprobar y las especificaciones consideradas no tienen que estar descritas en el mismo lenguaje. Se admite como lenguaje para las especificaciones de los SMAs cualquiera basado en el paradigma de agentes para el que se puedan establecer correspondencias con la TA.

El resto del capítulo se organiza en las siguientes secciones. El primer apartado justifica el uso de las contradicciones como elemento director del desarrollo de SMAs mediante la aplicación de los niveles de contradicción de la TA al desarrollo basado en agentes. A continuación se revisan los procesos necesarios para la definición de contradicciones de la TA a partir de sus fuentes textuales y el método de comprobación sobre especificaciones. La sección 5.4 describe un catálogo de contradicciones sociales extraídas de la TA. Después se muestra un ejemplo de aplicación de las contradicciones sobre una especificación de SMA. Finalmente, se discute la aplicación de las contradicciones a la ISOA en vista de los resultados obtenidos.

5.2. Contradicciones como elementos directores del desarrollo

Las contradicciones son inherentes al desarrollo de un proceso software, con independencia de la metodología empleada o de la habilidad y disciplina de los desarrolladores. Las contradicciones pueden aparecer de diversas formas que requieren tratamientos distintos. De acuerdo con la TA, existen varios niveles en los cuales pueden aparecer las *contradicciones* de un sistema (ver sección 3.2.3). Estos niveles también aparecen en el desarrollo y funcionamiento de SMAs:

- *Contradicciones primarias*. Se trata de contradicciones internas. Es la clase de situación que puede aparecer en un SMA donde hay objetivos conflictivos o los agentes persiguen objetivos incompatibles. El ejemplo prototípico de *contradicción primaria* en la TA es la que se da entre el beneficio de usar un recurso y el coste que tiene ese uso. Así, la actividad de un agente está condicionada por el balance entre sus objetivos y el coste de las tareas que se requieren para satisfacerlos. Los agentes son entidades intencionales cuyo propósito es satisfacer sus objetivos y frecuentemente esta satisfacción se establece en términos de provisión de servicios. Por otra parte, la ejecución de tareas requiere el uso de recursos y la colaboración de otros agentes, lo cual puede implicar un precio. Los agentes deben decidir si aceptan o rechazan los servicios según su coste y la importancia de los objetivos que satisfacen.
- *Contradicciones secundarias*. También son conocidas como las contradicciones externas. Son contradicciones que tienen su origen en el entorno de la *actividad*. Aparecen cuando en el *sistema de actividad* existen conflictos entre las capacidades esperadas en los *artefactos* para realizar las tareas relacionadas con su *rol*, y sus capacidades reales. Por ejemplo, cuando un agente no controla el recurso que se esperaba, el entorno no genera ciertos eventos o diferentes SMAs ejecutándose en el mismo entorno se influyen negativamente entre sí.
- *Contradicciones terciarias*. Estas contradicciones están relacionadas con la evolución de sistemas dinámicos, cuyo comportamiento puede ser parcialmente reprogramado, e.g. mediante la introducción de nuevos tipos de agentes en el SMA o cambiando las reglas organizativas o políticas. Los conflictos pueden aparecer con respecto al comportamiento no actualizado de algunos agentes del sistema.
- *Contradicciones cuaternarias*. Aparecen en la interacción entre una *actividad* y sus *actividades vecinas*. Un ejemplo de este tipo de contradicción es el cambio de protocolos para la comunicación entre los agentes, buscando extender o mejorar sus capacidades, mientras que se mantiene la compatibilidad con versiones previas.

En el momento actual de la investigación, sólo se están considerando *contradicciones primarias*, *secundarias* y algunos casos de *cuaternarias*. Muchas *contradicciones terciarias* y *cuaternarias* se refieren al problema de mejorar y

extender dinámicamente las capacidades de un SMA existente y forman parte del trabajo futuro.

Entre las contradicciones también hay que contemplar los errores en el análisis. Un sistema software es el resultado de la colaboración entre clientes y desarrolladores. Estos grupos de personas tienen normalmente un bagaje diferente y esto provoca malententendidos en su comunicación. Además, existe una marcada tendencia en buena parte de la Ingeniería del Software a describir los sistemas mediante múltiples vistas. Resulta muy difícil mantener todas las vistas actualizadas y consistentes.

Como se ha visto en el Capítulo 2, muchas de las metodologías orientadas a agentes contemplan los problemas producidos por las contradicciones en el desarrollo. Las técnicas adoptadas para tratar con ellas varían desde las guías de buenas prácticas a los métodos con una fuerte base formal. Sin embargo, todas ellas tienen en común el ver las contradicciones como meros problemas que han de ser:

- *Eliminados de las especificaciones*. Algunas metodologías como AAII/BDI [Kinny *et al.* 1996] y GAIA [Wooldridge *et al.* 2000] no permiten que los agentes puedan tener estados mentales contradictorios. Consideran que esta clase de situaciones son errores de análisis que han de ser solventados con formulaciones alternativas.
- *Simplemente detectados*. En muchas metodologías se trata de propiedades que pueden aparecer en el sistema. A veces pueden ser detectadas mediante los propios procesos de verificación de la metodología como una propiedad más (e.g. INGENIAS [Pavón & Gómez-Sanz 2003] o ADELFE [Bernon *et al.* 2002]). Los desarrolladores han de decidir en función de la contradicción detectada qué hacer con ella.

La TA propone una nueva perspectiva sobre las contradicciones: mirarlas como el motor del cambio en los sistemas, convertirlas en el principio que guía el desarrollo [Leontiev 1978]. Esto significa que las nuevas versiones del sistema emergen como soluciones a las contradicciones de las versiones previas. La propuesta de este capítulo sigue este principio. Las contradicciones son vistas como un caso particular de *propiedades anti-patrón* (ver sección 3.6). Estas propiedades están inmersas en un proceso de verificación y validación de *propiedades sociales* que incluye su detección pero también su posible solución. Se trata de identificar patrones de contradicción que reflejen las situaciones conflictivas en los modelos de SMAs. Estos patrones tienen asociadas configuraciones alternativas de los modelos para mejorar y hacer evolucionar las especificaciones.

5.3. Métodos asociados a las contradicciones

Las contradicciones son *propiedades anti-patrón* (ver sección 3.6) extraídas de las investigaciones sociológicas y psicológicas de la TA. Estos anti-patrones se describen con la representación para *propiedades sociales* introducida en la sección 3.6. Esencialmente esta representación incluye descripción textual, una serie de patrones de detección para localizar las propiedades en las especificaciones y patrones de solución que representan reconfiguraciones de las especificaciones para resolver la

contradicción. Cada patrón cuenta a su vez con una descripción textual y un diagrama UML-TA (ver sección 3.4). El diagrama UML-TA se lee como una red semántica [Russell & Norvig 1995] y actúa como un marco con ranuras que se rellenan con la información obtenida de los modelos en la detección de la propiedad.

Debido al uso de la infraestructura para la TA común a todo este trabajo, los métodos de definición y uso de contradicciones de la TA guardan grandes similitudes con otros ya presentados en esta tesis. Así, la identificación de contradicciones y su traducción a *propiedades sociales* es similar al proceso para definir las *preguntas* de la GCR (ver sección 4.3.2). Del mismo modo, el uso de contradicciones para detectar problemas en las especificaciones responde al método genérico de comprobación de propiedades introducido en la sección 3.7.

5.3.1. Método de definición

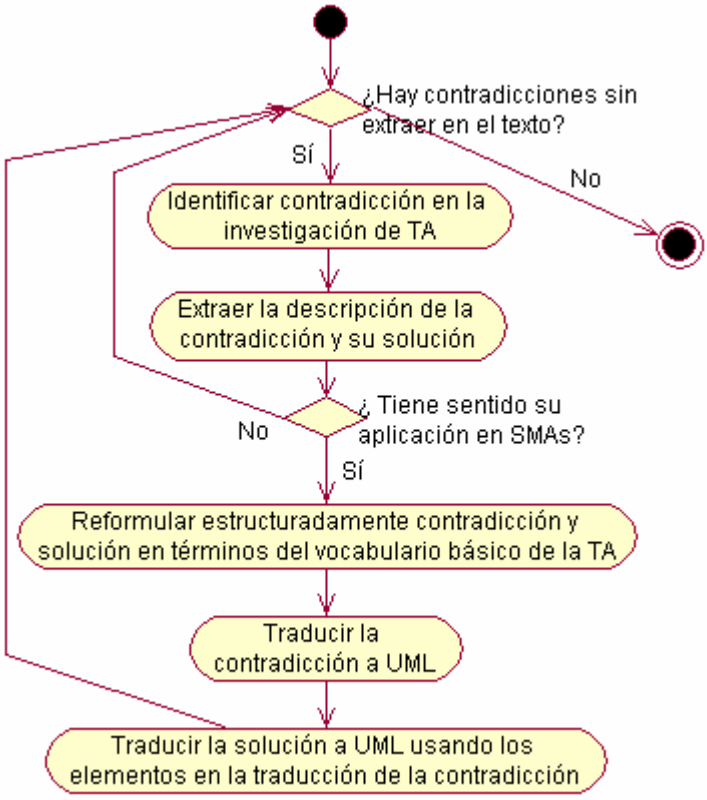


Fig. 55. Proceso de definición de *contradicciones* de la TA.

Resumidamente, el proceso de definición de contradicciones consta de cuatro fases. El primer paso es identificar en la literatura de TA una *contradicción*. Después se ha de determinar si ésta tiene sentido como información que deba considerarse bajo

el paradigma de agentes. En el caso de que la respuesta al estudio anterior sea afirmativa, la *contradicción* y su solución han ser clarificadas y estructuradas, a fin de que queden descritas con los términos esenciales de la TA (para una definición estos términos ver sección 3.2). Las formas textuales son traducidas después a su forma UML-TA atendiendo a consideraciones similares a las de la sección 4.3.2 a propósito de las *preguntas* de la GCR. Este proceso está descrito como un diagrama de actividad de UML en la Fig. 55.

Como se ha indicado, el proceso de definición de contradicciones comienza identificándolas sobre el texto de los estudios de TA. Afortunadamente, las *contradicciones* son un concepto central de la investigación en TA por lo que suelen aparecer claramente identificadas en sus trabajos. El problema de esta fase viene de la naturaleza discursiva de los análisis. Al igual que ocurre con otros estudios en las disciplinas sociales (léanse para algunos ejemplos [Bateson 1972, Werner & Shoepfle 1987, Bisgaard *et al.* 1989, Hutchins & Klausen 1992, Hall *et al.* 1995]), es difícil delimitar sobre los textos los elementos de interés (por ejemplo los componentes de un *sistema de actividad*) o eliminar sus ambigüedades.

Una vez que se han extraído la contradicción y su solución en su forma textual original, hay que evaluar si son utilizables en SMAs. Hay que comprender en qué consiste la contradicción y cuál puede ser su relación con el paradigma de agentes. Es decir, se trata de determinar si la contradicción describe un conocimiento que, extrapolado a la base común del paradigma de agentes, representa una situación conflictiva en un SMA. Del mismo modo, hay que establecer si la solución resolvería la contradicción al ser aplicada sobre unas especificaciones de SMA.

Cuando se ha identificado una *contradicción* aplicable a SMAs, el siguiente paso es representarla con la estructura para *propiedades sociales* (ver sección 3.6). Esta estructura consta de una descripción textual genérica de la contradicción, patrones de detección (que corresponderán a la contradicción propiamente dicha) y patrones de solución (que recogerán la solución a la contradicción).

La descripción textual de la propiedad es una reescritura de la formulación de la *contradicción* en la literatura de TA. La nueva descripción ha de ser sintética (frente a la discursiva original) e indicar explícitamente los conceptos básicos de la TA. Este paso tiene un cierto equivalente en la identificación de los conceptos propios de la TA sobre la forma textual de las *preguntas* de la GCR.

A continuación se han de describir los patrones de detección de la contradicción. La situación conflictiva se representa mediante UML-TA (ver sección 3.4). Los conceptos y relaciones a incluir son indicados por la descripción textual de la contradicción previamente elaborada. Un aspecto importante a considerar de este patrón es su papel en la detección de la propiedad sobre especificaciones de SMAs. El patrón será instanciado con información de las especificaciones para localizar y explicar la contradicción. Es decir, el patrón se comporta como un marco cuyas ranuras se rellenan con información de las especificaciones. Por tanto, no sólo se han de identificar los conceptos sino también los valores en ellos que serán modificables, tales como nombres o valores de propiedades específicas.

El proceso de definición de las *contradicciones* de la TA no termina con los patrones de detección como ocurría con las *preguntas* de la GCR. Las contradicciones sí tienen asociados patrones de solución. En las contradicciones hay que considerar cómo se puede expresar su solución en términos de su patrón de detección. Por

supuesto, se puede considerar añadir nuevas entidades y relaciones, pero la base ha de ser una reconfiguración de las entidades que originaron el problema y que ya formaban parte de las especificaciones. Una vez más, hay que decidir también qué ranuras del patrón de solución serán modificables.

Hay que hacer dos puntualizaciones al proceso que se acaba de presentar. Una se refiere a la evaluación de las *contradicciones* de la TA como información para SMAs y la otra al esfuerzo que requiere el proceso.

Acerca de la viabilidad de las *contradicciones*, en este capítulo se ha optado por considerar sólo aquellas *contradicciones* de la TA aplicables sobre la base común del paradigma de agentes, es decir, basadas en contemplar un SMA como una sociedad de actores intencionales. Considerando metodologías concretas de la ISOA es posible incorporar otras contradicciones, que ahora son descartadas porque no pueden ser representadas en algunas de las metodologías revisadas. Éste es el caso de contradicciones que tienen que ver con aspectos perceptivos-mentales o con la creación de funciones mentales de orden superior como las que aparecen en [Engeström & Middleton 1998]. Estas contradicciones podrían considerarse por ejemplo en KAOS (ver sección 2.3.2) o DESIRE (ver sección 2.3.5) pero no en Tropos (ver sección 2.3.6) o INGENIAS (ver sección 2.3.8).

En cuanto al esfuerzo requerido indicar que al igual que ocurría con la GCR, el estudio para definir contradicciones implica conocimientos avanzados acerca de la TA y los SMAs. No obstante, este esfuerzo tiene la ventaja de ser reutilizable. Las contradicciones constituyen librerías de conocimiento experto procedente de las ciencias sociales. Este conocimiento puede ser aplicado a cualquier desarrollo de SMAs siempre que existan traducciones disponibles para su lenguaje. Este uso como repositorios de conocimiento se muestra en el Capítulo 6, dedicado a las herramientas automatizadas de ayuda al desarrollo, y se ejemplifica en el Capítulo 7, dedicado al caso de estudio.

5.3.2. Método de uso

Las contradicciones se usan en el marco de un proceso de localización y tratamiento de *propiedades sociales* en los modelos de SMAs (ver sección 3.7). El proceso propuesto se basa en la representación de las *propiedades sociales* mediante patrones de detección y solución. El método encuentra en las especificaciones instancias de las *propiedades sociales* mediante sus patrones de detección y modifica los modelos de acuerdo con los patrones de solución. La representación de las propiedades como patrones estructurales de detección permite reducir el proceso de descubrimiento de propiedades a buscar correspondencias entre el patrón y los modelos.

Como se indicó al presentarlo, este proceso puede ser contemplado desde distintos puntos de vista según las *propiedades sociales* con las que se usa: captura de requisitos, verificación y validación. Las *contradicciones* de la TA son un caso particular de las denominadas *propiedades anti-patrón*. Las *propiedades sociales anti-patrón* se relacionan con la verificación de SMAs. Se trata de propiedades cuya aparición se desea evitar en las especificaciones. Como herramienta de verificación, las *propiedades sociales* no muestran el modo de construir el SMA, al estilo de los

patrones de diseño [Gamma *et al.* 1995], sino que su objetivo es detectar configuraciones conflictivas en los modelos y sugerir sus posibles soluciones.

El proceso de la sección 3.7 no es completamente automático ya que requiere la intervención de los usuarios. En primer lugar, estos han de seleccionar las contradicciones a comprobar y establecer para ellas los valores necesarios de las variables en las ranuras de los patrones de detección. Una vez localizadas las contradicciones en las especificaciones, se requiere el juicio de los usuarios para determinar qué contradicciones tienen un significado real en su problema. Las contradicciones son patrones genéricos de conflicto, por lo que establecer si constituyen un problema real depende de su significado en el contexto en el que se hayan inmersas. Finalmente, si se detecta una contradicción de interés, el usuario ha de evaluar si la solución propuesta le parece adecuada y si es así, fijar en el patrón de solución los valores de las ranuras que sean necesarios.

Para terminar, decir que el uso de este método, al igual que el de las otras técnicas basadas en TA, se tiene que complementar con procesos convencionales de Ingeniería del Software. Se trata únicamente de una herramienta de ayuda al desarrollo de SMAs que permite identificar configuraciones potencialmente conflictivas en los modelos y proponer soluciones.

5.4. Contradicciones en la TA para SMAs

La investigación en TA [Vygotsky 1978, Leontiev 1978, Engeström 1987, Kuutti 1996, Bednyi & Meister 1997] describe situaciones conflictivas en las *actividades* humanas, las *contradicciones*. Los investigadores comienzan estudiando la situación de una sociedad. Partiendo de la situación social identificada, se sigue la respuesta que da la sociedad para tratar de modificarla y se estudia la situación resultante. En base a esta información de análisis se establecen contextos sociales, algunos de los cuales son *contradicciones*. Estos contextos son descripciones de la configuración de la sociedad en un momento dado. El resultado final de los estudios de la TA es una explicación de las razones del comportamiento de las organizaciones humanas y su evolución, en base a los contextos y las reacciones a ellos.

A lo largo de las siguientes secciones se presentan algunas de las *contradicciones* extraídas de la literatura sobre TA como *propiedades anti-patrón*, usando las estructuras para *propiedades sociales* de la sección 3.6. Cada una de estas *contradicciones* está introducida con una reseña de su contexto en la investigación sociológica y psicológica de la TA. Después se dan sus patrones de detección para la ISOA con el lenguaje UML-TA y se explica su significado en unas especificaciones de SMA. Finalmente se presentan los patrones de solución asociados a dicha *contradicción*. La porción del patrón de solución que cambia respecto al patrón de detección se enmarca con trazos discontinuos. Aquí se debe señalar que aunque en esta presentación sólo se describe por cada contradicción un patrón de detección y otro de solución, es posible contar con varios que respondan a distintas interpretaciones de la fuente textual (ver sección 3.6).

Antes de comenzar a describir los patrones, conviene recordar brevemente algunas características de UML-TA y sus estructuras:

- *Valores fijos y variables*. Los patrones actúan como marcos con ranuras editables. Estas ranuras pueden contener en ocasiones valores fijos que se representarán entre ‘ “ ’ y ‘ ” ’.
- *Elementos equivalentes*. Existen notaciones de UML-TA que pueden ser equivalentes en ciertos contextos. El *rol artefacto* puede ser reemplazado por cualquier otro *rol* válido en un *sistema de actividad*. Otro caso es la relación *persigue* entre una *actividad* y un *objetivo* que puede ser reemplazada por una relación *contribuye positivamente*. Una lista completa de estas equivalencias puede verse en la sección 3.4.2.
- *Deducción de relaciones*. Algunas de las relaciones que figuran en los patrones no tienen que ser directas necesariamente, sino que pueden ser deducidas. En la sección 3.4 se introdujeron las relaciones *juega* y de contribución. La relación *juega* entre *sujetos* hace que un *sujeto* persiga los *objetivos* y participe en las mismas *actividades* que otro. En cuanto a las relaciones de contribución, cuando un *producto* contribuye a un *objetivo* se considera que todos los *artefactos* del *sistema de actividad* que genera el *producto* contribuyen al *objetivo* de la misma forma, salvo que se indique explícitamente lo contrario.

Una última indicación es que las explicaciones y patrones de las contradicciones, aunque formulados para su uso con SMAs, están descritos con las abstracciones propias de la TA. Hay que recordar que su aplicación a una metodología orientada a agentes concreta requeriría el uso de las correspondencias de traducción.

Tras estas indicaciones se procede a presentar el repositorio de *contradicciones* de la TA como *propiedades anti-patrón*. Las contradicciones consideradas son: *Significado Dual*, *Valor de Intercambio*, *Objetivo Social*, *Doble Vínculo*, *Reflexión sobre la Actividad*, *Interrupción de Uso*, *Paradoja de Planificación*, *Estado de Necesidad*, *Conflicto Productor-Usuario* e *Información Contradictoria*.

5.4.1. Significado Dual

La *contradicción* de *Significado Dual* está directamente relacionada con las contradicciones internas de la *actividad* humana. Estas contradicciones emergen de la existencia dual de la *actividad* como producción individual independiente y al mismo tiempo subordinada a la producción total de la sociedad. Este choque surge de la *división del trabajo* [Marx 1909].

La *división del trabajo* emerge naturalmente en las *comunidades* prehistóricas según diferencias fisiológicas por edad y sexo. Los distintos grupos de individuos tienen diferentes capacidades que les hacen más adecuados para ciertos trabajos. Esta división se ve acentuada con el incremento de la población, la expansión de la *comunidad* y la subordinación de parte de sus miembros a otros. Por otra parte, las diferentes *comunidades* encuentran diferentes medios de producción. Cuando estas *comunidades* entran en contacto se ven llamadas al intercambio de *productos*. El intercambio no crea las diferencias entre las distintas esferas de producción de una misma *comunidad* pero sí pone en relación *comunidades* que eran diferentes, convirtiéndolas en elementos más o menos interdependientes del colectivo de

producción de una sociedad expandida. Esta relación acentúa las diferencias previas entre las esferas de producción de una *comunidad*.

Con la *división del trabajo* se diferencian los motivos de las *actividades* tal y como las realizan los individuos del significado que adquieren en la sociedad. Así, la *actividad* de un artesano está dirigida a la fabricación de un *producto*. Sin embargo, esa misma *actividad*, contemplada en el contexto global de la sociedad, es un medio de obtener un *objeto* que una vez vendido permitirá al artesano adquirir otros bienes. También en la sociedad, el cliente del artesano ve la *actividad* de fabricación como aquella que generará el *producto* que él necesita. Como consecuencia, el mismo *objeto* es el responsable de satisfacer de forma inmediata la necesidad del cliente, proporcionar ingresos al artesano por su venta y a la vez el resultado de un proceso de elaboración.

Por tanto, la *división del trabajo* pone en contacto y a la vez diferencia las visiones que los *sujetos* tienen de su entorno y de los elementos en él. En este crisol de perspectivas surgen los múltiples significados de los *objetos* para los *sujetos* que son la base de esta *contradicción*.

La *contradicción* de *Significado Dual* surge porque un *objeto*, inmerso en una red de *actividades* interconectadas, puede satisfacer distintas necesidades para los *sujetos* involucrados. Estas necesidades pueden responder a puntos de vista contradictorios en la *comunidad*, e.g. beneficio frente a calidad. Cuando las necesidades a las que responde un *objeto* son contradictorias entre sí existe una situación de *Significado Dual* para el *objeto*.

5.4.1.1. Representación UML-TA del Significado Dual

La situación de *Significado Dual* surge cuando un objeto responde a necesidades contradictorias en una red de *actividades*. Las "necesidades" de *sujetos* o *comunidades* se expresan en el vocabulario de la TA como *objetivos*. La *contradicción* indica la existencia de un "objeto" que contribuye de forma opuesta a esos *objetivos*. En la TA cualquier elemento del modelado puede contribuir a un *objetivo*, sea cual sea el *rol* que desempeña el elemento en el *sistema de actividad*. Por ello la traducción más adecuada para el "objeto" de la introducción es el *rol* de *artefacto*. El hecho de que los *objetivos* sean "necesidades contradictorias" respecto al *artefacto* corresponde a relaciones de contribución de carácter opuesto con él. Ejemplos de estas relaciones de contribución contradictorias son *contribuye positivamente* y *contribuye negativamente* o *garantiza* e *indefinida*. Además, para que exista la *contradicción* los *objetivos* han de ser perseguidos por un mismo *sujeto* que puede ser individual o social, e.g. en la introducción la comunidad o el artesano. La formulación resultante de la *contradicción* de *Significado Dual* se ve en la Fig. 56.

Una situación de *Significado Dual* (ver Fig. 56) surge cuando un *artefacto* afecta a distintas necesidades dentro de una *actividad*, puesto que puede desempeñar varios papeles a distintos niveles. Estos usos diferentes quedan representados por sus contribuciones a *objetivos*. Si estos usos no son consistentes desde el punto de vista de un *sujeto*, se produce una contradicción interna al *sistema de actividad*.

La *contradicción* de *Significado Dual* no implica siempre un error de análisis. La situación puede llamar la atención sobre la necesidad de discutir ciertos aspectos del SMA y su entorno con los clientes o puede detallar posibles puntos de expansión del análisis.

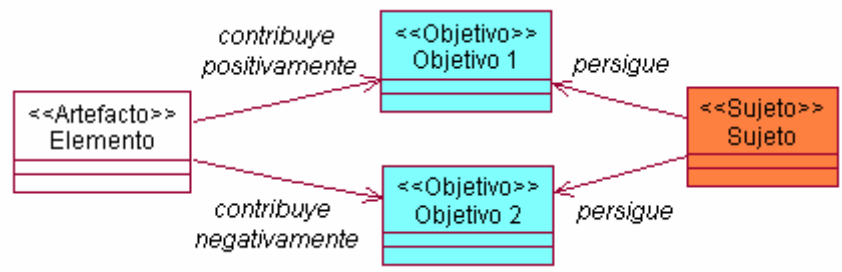


Fig. 56. Contradicción de *Significado Dual*.

5.4.1.2. Soluciones del Significado Dual

Una posible solución a una *contradicción* de *Significado Dual* es refinar las contribuciones del *artefacto* a los *objetivos*. Si el *artefacto* ha de ser preservado en el sistema es porque resulta *esencial* para alcanzar alguno de los *objetivos*. Si este *objetivo* no existe, el *artefacto* debería ser eliminado del sistema. Al final, el modelo podría quedar como el que se puede ver en la Fig. 57.

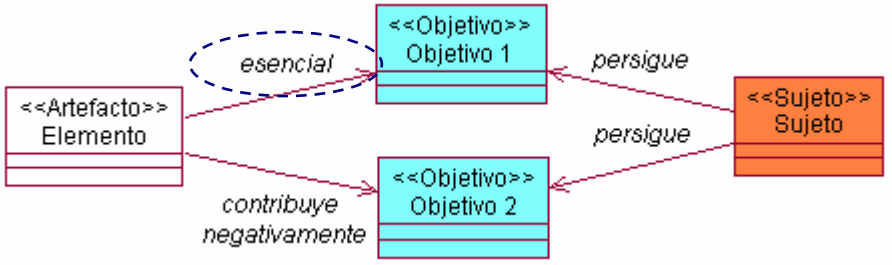


Fig. 57. Una posible solución de la contradicción de *Significado Dual*.

Otras soluciones posibles serían establecer prioridades entre los *objetivos* mediante la relación *supera*, o bien determinar si hay evidencias antes las cuales se ha de usar necesariamente el *artefacto* para satisfacer un *objetivo*. De esta forma el *artefacto* cumpliría con su *rol* sólo si contribuía positivamente al *objetivo* más importante en ese momento.

5.4.2. Valor de Intercambio

Como se vio anteriormente, la contradicción inherente de la *actividad* es su existencia dual como independiente y subordinada a la *actividad* social completa. Esta contradicción adopta diferentes formas históricas según la organización económica reinante. En las modernas sociedades capitalistas se plasma en el doble valor de los *objetos*, de uso y de intercambio [Engeström 1987]. El valor de uso del *objeto* es el que viene determinado por las necesidades del *sujeto* que satisface. El valor de intercambio queda establecido por la cantidad media de trabajo social que es

necesaria para su producción. Los *objetos* que poseen valor de intercambio son *mercancías* [Ilyenkov 1982]. Los *artefactos* se convierten en *mercancías* porque son el producto de la labor de individuos o grupos de individuos privados que trabajan independientemente de la sociedad. Es decir, los productores de las *mercancías* no tienen contacto social con otros productores hasta el intercambio de sus productos, por lo que el carácter de la labor de cada productor sólo se muestra en el *momento de intercambio*[3] de la *actividad* [Marx 1909].

La contradicción esencial de la *actividad* se plasma entonces en la mutua dependencia y a la vez exclusión del valor de uso y el de intercambio en cada *mercancía*. Esta contradicción domina cada esquina del triángulo del *sistema de actividad* (ver Fig. 4). La existencia de los *artefactos* de la *actividad* como *mercancía* hace que el *sujeto* perciba la labor productiva como ajena a la satisfacción directa de sus necesidades. El individuo sólo obtiene los medios para satisfacer dichas necesidades entregando el resultado de su *actividad* a otros miembros de la *comunidad* [Leontiev 1978].

La *contradicción* del *Valor de Intercambio* aparece en una situación en la que un *sujeto* ha de crear un *producto* para la *comunidad*. Sin embargo, el *objetivo* final del *sujeto* involucrado en la *actividad* productora sólo puede ser satisfecho por un *producto* creado por su *comunidad*. La *contradicción* surge de la ausencia de este *producto* que ha de satisfacer la necesidad del *sujeto*, por lo que el *sujeto* no tiene un motivo para realizar la *actividad*.

5.4.2.1. Representación UML-TA del Valor de Intercambio

La *contradicción* del *Valor de Intercambio* involucra a un *sujeto* que crea un *producto* para su *comunidad*. La traducción a UML-TA de estos elementos implica por tanto la aparición de un *sujeto* individual, de la *actividad* realizada por éste y del *producto* de dicha *actividad*. Por otra parte se indica que este *producto* satisface una necesidad de la *comunidad*. Esta satisfacción podría indicarse mediante una relación de contribución de carácter positivo entre el *producto* y un *objetivo* perseguido por la *comunidad*. Sin embargo, la *contradicción* hace referencia también a que la *comunidad* debería crear un *producto* para el *sujeto*. Por ello se opta por representar esta relación mediante una *actividad*. La *comunidad*, representada como un *sujeto* colectivo, realiza una *actividad* para satisface uno de sus *objetivos* y utiliza el *producto*. Como se desconoce la utilidad específica del *producto* en la *actividad* de la *comunidad*, se elige representarlo de la forma más genérica posible, mediante el *rol artefacto*. El último elemento a modelar son los *objetivos* del *sujeto* individual involucrados en la *contradicción*. Existen dos *objetivos* diferentes, aquel que satisface directamente su *actividad* y otro que es el que debiera ser satisfecho por la *comunidad*. Con estos elementos la *contradicción* del *Valor de Intercambio* puede modelarse como se muestra en la Fig. 58.

[3] Los *momentos de la actividad* fueron descritos en la sección 3.2.1.

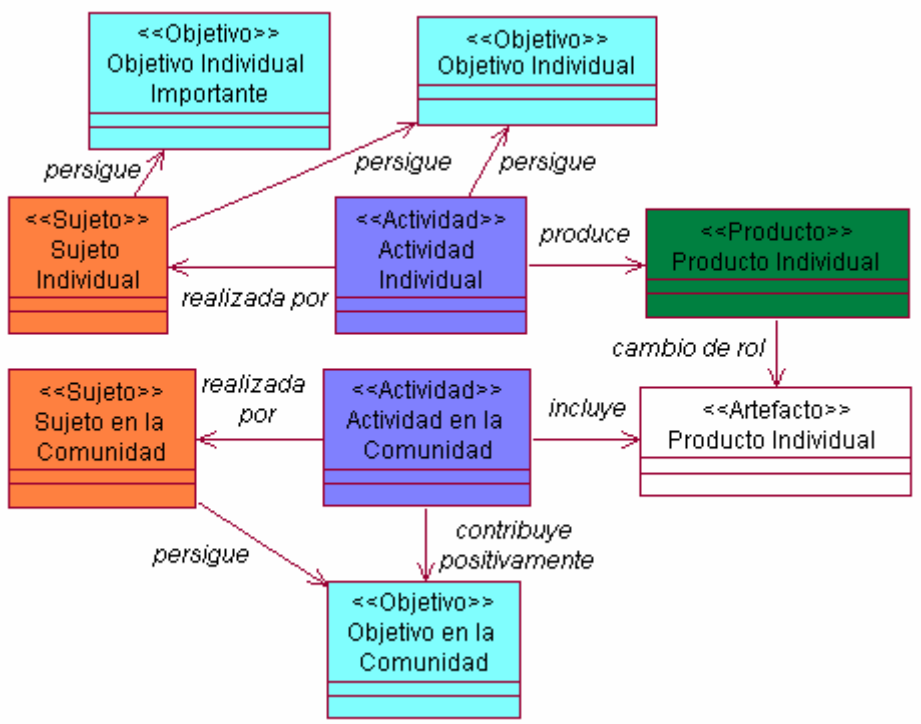


Fig. 58. Contradicción del *Valor de Intercambio*.

La *contradicción* del *Valor de Intercambio* radica en que un *sujeto*, i.e. el *Sujeto Individual*, ha de crear un *producto* para el consumo de otros miembros de la *comunidad*. Sin embargo, este *sujeto* no ve satisfechos *objetivos* suyos de suficiente relevancia como para que superen los inconvenientes de la tarea, como pueden ser los costes asociados. Incluso podría darse el caso de que no hubiera *objetivos* del *sujeto* satisfechos directamente por la creación del *producto*. Debido a ello, aunque el *Sujeto Individual* tiene la capacidad para crear el *producto* y ponerlo a disposición de la *comunidad*, representada por *Sujeto en la Comunidad*, no siente la inclinación a hacerlo. La Fig. 58 representa el hecho de que el *sujeto* no ve satisfecha por la creación del *producto* un *objetivo* suyo de suficiente relevancia, por lo que nunca llega a ejecutar las tareas asociadas a dicha creación. Ello impide a los miembros de la *comunidad* que dependen de dicho *producto* satisfacer sus necesidades.

La diferente relevancia del *Objetivo Individual Importante* y el *Objetivo Individual* podría marcarse con una relación *supera*.

5.4.2.2. Soluciones del Valor de Intercambio

Una forma de solucionar la *contradicción* del *Valor de Intercambio* es lograr que el *Sujeto Individual* obtenga un beneficio superior de ejecutar la *Actividad Individual*. Se trata de hacer explícito el *momento de intercambio* de la *actividad* global de la *comunidad*. Los *sujetos* representados por el *Sujeto en la Comunidad*, aquellos que se

benefician del *producto*, deben recompensar al *sujeto* que lo produce. Para ello pueden proporcionar al *Sujeto Individual productos Recompensa* que le satisfagan algún objetivo adicional, el *Objetivo Individual Importante* en este caso. Para crear este nuevo *producto* se introduce la *Actividad de Retribución* que es realizada por el *Sujeto en la Comunidad*. El lazo de *actividades* constituido por la *Actividad Individual*, *Actividad en la Comunidad* y la nueva *Actividad de Retribución* es la representación explícita del *momento de intercambio* del *sistema de actividad* global El modelo que incluye esta representación del *momento de intercambio* quedaría como el de la Fig. 59.

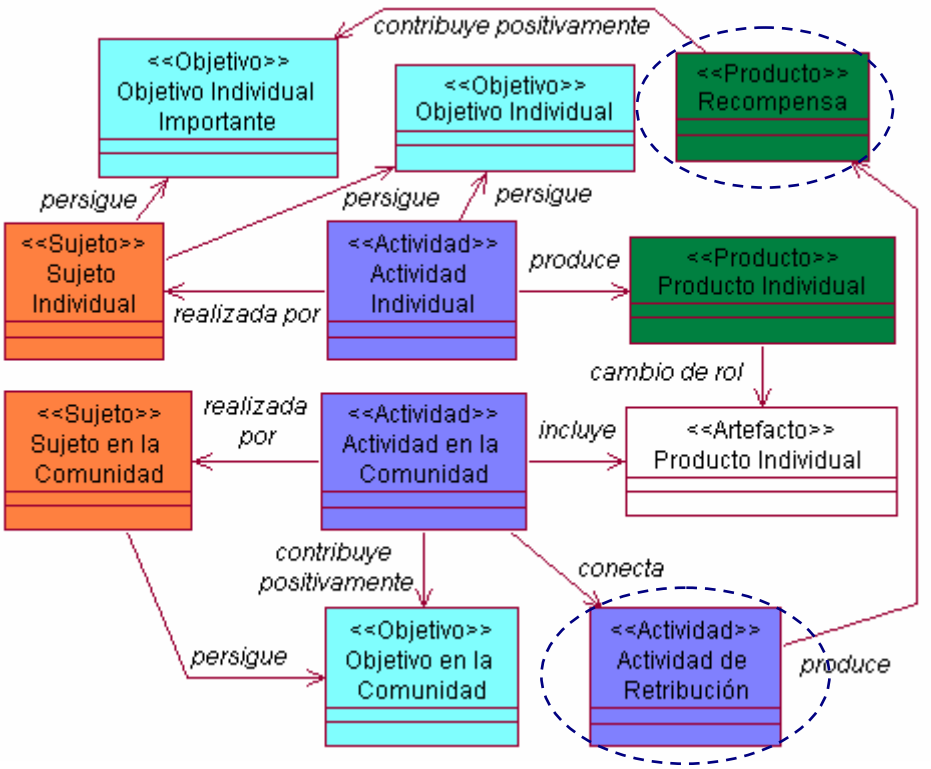


Fig. 59. Una posible solución a la contradicción del *Valor de Intercambio*.

5.4.3. Objetivo Social

Toda la actividad humana es mediada por la sociedad del *sujeto* [Leontiev 1981]. El hombre sólo se relaciona con su entorno a través de *herramientas* y es su sociedad la que le proporciona dichas *herramientas*, en las cuales cristaliza su experiencia en la *actividad*. Pero la mediación de la sociedad en la *actividad* del individuo no sólo se produce a través de las *herramientas*: el carácter social de la *actividad* se hace evidente cuando se considera la *división del trabajo*.

La *división del trabajo* surge con la transición del hombre individual a la sociedad. Según la TA, con la *división del trabajo* el *sujeto* pasa de realizar *actividades* a ejecutar *acciones*. Las *acciones* del *sujeto* cumplen *metas* conscientes que no satisfacen directamente sus necesidades, pero que contribuyen a alcanzar los *motivos* de la *actividad* de la *comunidad*. Las *acciones* generan resultados parciales que aislados no bastan para satisfacer las necesidades de los participantes. Sólo la relación del *sujeto* con el resto de los miembros de la *comunidad* le conecta con el verdadero *motivo* de la *actividad*. El *momento de intercambio* (ver Fig. 4) es el que da a cada *sujeto* parte del *producto* de la *actividad* colectiva, y es este *producto* el que realmente satisface sus necesidades.

Del razonamiento anterior se deduce que la *actividad* colectiva necesita contemplar el *momento de intercambio* para hacer llegar sus *productos* a los individuos. En caso contrario, la *actividad* pierde su sentido para el *sujeto*. Por esta razón se puede decir que la *contradicción* del *Objetivo Social* y del *Valor de Intercambio* comparten una cierta similitud. Aunque desde diferentes puntos de vista, ambas se producen por una ausencia o errónea implementación del *momento de intercambio* de la *actividad* colectiva.

La *contradicción* del *Objetivo Social* aparece en situaciones en las que hay un *objetivo* perseguido por los miembros de la *comunidad*. Este *objetivo* sólo puede ser satisfecho mediante la consecución de *objetivos* parciales. A su vez, estos *objetivos* parciales están asignados a diferentes miembros de la *comunidad*. Parte de los *sujetos* sólo pueden satisfacer el *objetivo* global si los demás miembros de la *comunidad* les proporcionan explícitamente los resultados de sus *objetivos* parciales.

5.4.3.1. Representación UML-TA del Objetivo Social

La *contradicción* del *Objetivo Social* se centra en el trabajo de una *comunidad* de *sujetos* a través de los *objetivos* que satisfacen como grupo pero persiguen individualmente. La *contradicción* involucra un *objetivo* global que se descompone en *objetivos* parciales. Cada *sujeto* de la *comunidad* persigue el *objetivo* social y uno de los *objetivos* parciales. Gracias a la relación de descomposición, el *objetivo* global se considera satisfecho cuando se satisfacen algunos *objetivos* parciales. La satisfacción de los *objetivos* parciales se representa mediante la ejecución de *actividades*. Cada *actividad* sólo se relaciona con el *sujeto* que la realiza y el *objetivo* que persigue, ya que no existe *momento de intercambio* en el *sistema de actividad*. Sin embargo, algunos de los *sujetos* necesitan *objetos* provistos por otros miembros de la *comunidad* para satisfacer los *objetivos* parciales que les permiten cumplir el *objetivo* global. La *contradicción* radica en la falta de conexión entre las *actividades*, lo que impide el intercambio de estos *objetos*. Es decir, ninguna *actividad* tiene una asociación con un *sujeto* que no sea el suyo o con otras *actividades*, por lo que su *producto* no puede ser usado nada más que por su propio *sujeto*. La representación resultante del *Objetivo Social* puede verse en la Fig. 60. La *comunidad* de los *sujetos* no está explícitamente representada en el patrón.

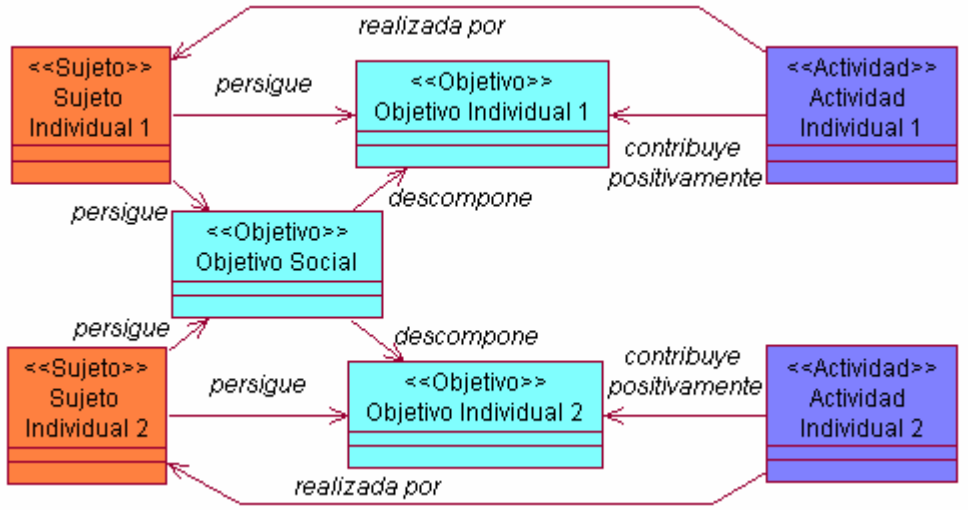


Fig. 60. Contradicción del *Objetivo Social*.

El *Objetivo Social* corresponde a las situaciones en las que los *sujetos* de una *comunidad* tienen un *objetivo* que se alcanza mediante la satisfacción de una serie de *objetivos* intermedios. Los distintos individuos son responsables sólo de la ejecución de tareas asociadas a *objetivos* parciales. La satisfacción de su *objetivo* global implica sin embargo al resultado final de la *actividad* total de la *comunidad*. Por tanto, para satisfacer dicho *objetivo* global, los *sujetos* han de intercambiar información acerca de la ejecución de la *actividad* con el resto de los miembros de la *comunidad*. Si este intercambio no se produce los *sujetos* no pueden saber que han alcanzado sus *objetivos* finales. Esta visión del *Objetivo Social* se presenta en la Fig. 60.

5.4.3.2. Soluciones del Objetivo Social

La *contradicción* del *Objetivo Social* surge por la inexistencia del *momento de intercambio* en la *actividad*. Este momento ha de hacer partícipes a todos los miembros de la *comunidad* del resultado de la *actividad* global. Puesto que cada *sujeto* sólo produce un resultado parcial, una posible forma de incluir este momento es mediante tareas que comunican a cada *sujeto* los resultados parciales que les puedan interesar de otras *actividades* parciales. Estas tareas son las *Actividades de Intercambio*. La comunicación de resultados a los *sujetos* de la *comunidad* es representada haciendo que los *sujetos* se conviertan en *objetos* de las *Actividades de Intercambio*. El modelo con las nuevas tareas sería como el de la Fig. 61. Obviamente, las *Actividades de Intercambio* necesitarían *objetos* adicionales para realizar su función que no están representados en el diagrama. Algunos de estos *objetos* necesarios podrían ser la información suministrada o las herramientas para realizar la comunicación.

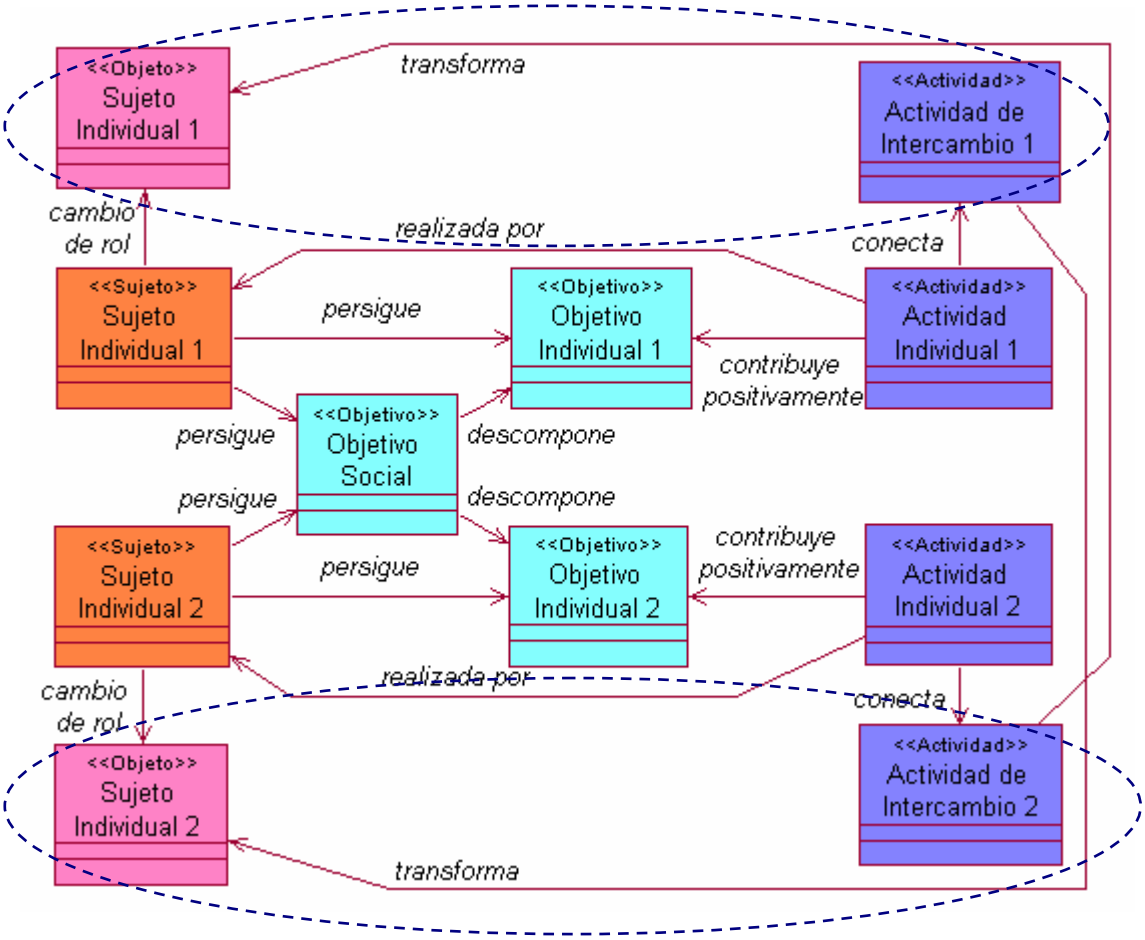


Fig. 61. Una posible solución a la contradicción del *Objetivo Social*.

5.4.4. Doble Vínculo

El *Doble Vínculo* fue descrito por primera vez en 1956 en el trabajo de Bateson, en relación con el estudio de la esquizofrenia. Un *Doble Vínculo* [Bateson 1972] es "*una situación en la que no importa lo que el individuo haga porque no puede ganar*". En una situación de *Doble Vínculo*, el *sujeto* está involucrado en una relación intensa en la cual recibe varios mensajes u ordenes que se niegan entre sí. Sin embargo el individuo no es capaz de hacer un razonamiento metacomunicativo sobre dichos mensajes para resolver la contradicción. Citando un ejemplo de Bateson "*Si me dices que este palo es real, te golpearé con él. Si me dices que este palo no es real, te golpearé con él. Si no dices nada, te golpearé con él.*". El *Doble Vínculo* no es una acción sobre alguien; por el contrario, reside en una interacción a lo largo del tiempo

en la cual relaciones importantes para el *sujeto* están sujetas crónicamente a invalidación por una interacción paradójica.

En Psicología, el producto más conocido de continuos *Dobles Vínculos* es la esquizofrenia. Se trata de una reestructuración profunda de la consciencia del *sujeto*, causada por su permanente incapacidad de resolver los contextos contradictorios que se le presentan.

Siguiendo una vez más el ejemplo de Bateson, si el *sujeto* fuese capaz de comentar metacomunicativamente estos contextos "*se levantaría y arrebataría el palo al otro individuo*". En otras palabras, el *sujeto* trascendería las restricciones del contexto al situarse en un marco más amplio donde éstas se tornan relativas y modificables.

La *contradicción* del *Doble Vínculo* aparece cuando el *sujeto* se encuentra en una situación la que todas las tareas que pude realizar tienen algún efecto negativo sobre sus propios *objetivos*. En estas condiciones no es capaz de elegir ninguna de esas tareas porque siempre se estaría perjudicando.

5.4.4.1. Representación UML-TA del Doble Vínculo

La *contradicción* del *Doble Vínculo* aparece en torno a un *sujeto*, sus tareas y los *objetivos* que persigue. Las tareas del *sujeto* se modelan como *actividades* realizadas por éste. Para modelar la cuantificación universal referida a todas las *actividades* del *sujeto* se emplea el adorno de relaciones *todos*. Este adorno implica que el patrón tiene que mantenerse para todas las *actividades* que tengan la relación señalada con el *sujeto*. La cuantificación existencial referida al *objetivo* perjudicado por la *actividad* está implícita en el patrón. Es decir, los patrones representan configuraciones que ocurren al menos una vez, por lo que basta con expresar la relación de *contribuye negativamente* entre algún *objetivo* y las *actividades*. El diagrama para el *Doble Vínculo* con estos elementos aparece en la Fig. 62.

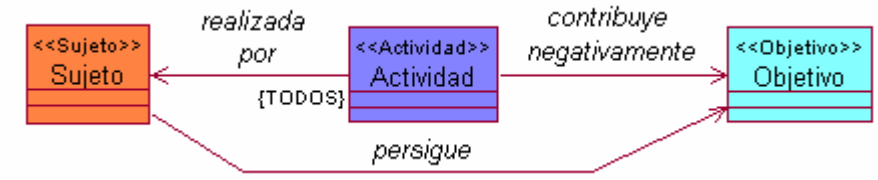


Fig. 62. Contradicción de *Doble Vínculo*.

Un *Doble Vínculo* representa una situación en la que no importa lo que haga el *sujeto* porque cada una de sus *actividades* tiene efectos negativos sobre al menos uno de sus *objetivos*. Es decir, el *sujeto* es capaz de llevar a cabo varias *actividades* pero todas tienen asociado algún *objetivo* que es afectado negativamente por esa acción.

5.4.4.2. Soluciones del Doble Vínculo

Una posible solución a la *contradicción* de *Doble Vínculo* es alterar la forma en la que se llevan a cabo las *actividades* conflictivas, es decir, trascender el contexto preconfigurado para su realización. Una *actividad* puede realizarse a través de diferentes secuencias de *actividades*. Aunque posiblemente algunas de las *actividades* en la secuencia preservarán el efecto negativo sobre los *objetivos* de la *actividad*

original, es posible introducir otras con efectos positivos. El modelo podría ser cambiado para incluir esta información de la manera que se ve en la Fig. 63.

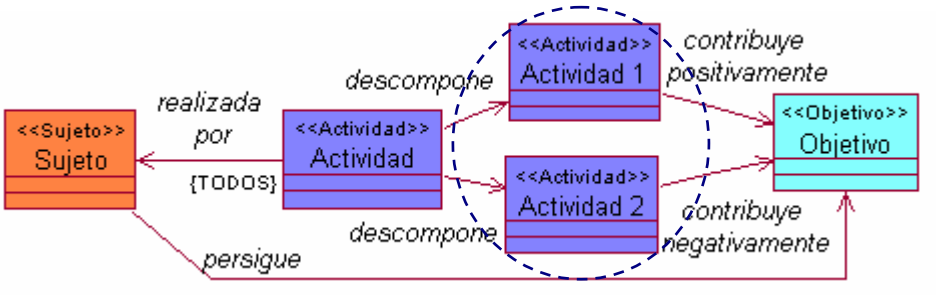


Fig. 63. Una posible solución a la contradicción de *Doble Vínculo*.

La *Actividad* original es descompuesta en las nuevas *Actividad 1* y *Actividad 2*. Si bien la *Actividad 2* mantiene el efecto negativo original, la *Actividad 1* introduce un efecto positivo sobre el *Objetivo*.

5.4.5. Reflexión sobre la Actividad

Cuando se introdujo el concepto de *actividad* se indicó que ésta se dirigía hacia un *objeto* que representaba su verdadero motivo. Así, el individuo realiza la *actividad* para satisfacer unas necesidades mediante la transformación del *objeto*. Sin embargo, las *actividades* no son sólo tareas ancladas a la realidad. Éstas y sus motivaciones constituyen también elementos para la reflexión mental del *sujeto*. Sólo cuando las *actividades* se confrontan con el *objeto* capaz de satisfacer la necesidad que conllevan, quedan concretadas en el mundo real.

La reflexión sobre las tareas involucradas en la *actividad* y su ejecución implica una completa jerarquía de motivaciones y tareas asociadas [Leontiev 1978]. Esta capacidad de reflexión es la que permite a los *sujetos* anticipar su ejecución y resultados en un plano meramente psíquico. Los *sujetos* deben aprovechar estas capacidades para conocer sus necesidades conscientes e inconscientes *a priori* y así controlar en alguna medida el resultado de la *actividad* [Leontiev 1981]. Si el *sujeto* no lleva a cabo esta reflexión está obviando un aspecto importante de la realización de sus *actividades* previa a su ejecución real.

La *contradicción* de la *Reflexión sobre la Actividad* incide en el hecho de que el *sujeto* cuenta con una "imagen mental" de la *actividad* a realizar y sus requisitos. Ésta le permite reflexionar sobre cómo ha de satisfacer sus *objetivos* anticipadamente. En otras palabras, el *sujeto* es consciente, dentro de sus límites cognitivos y perceptivos, del entorno concreto en el que se haya y de los medios de que dispone para alcanzar sus *objetivos*. En función de esta información ha de organizar su respuesta para elegir el mejor curso de acción con el que alcanzar sus *objetivos*.

5.4.5.1. Representación UML-TA de la Reflexión sobre la Actividad

La *contradicción* de la *Reflexión sobre la Actividad* se centra en el análisis de aquellos aspectos de la misma que pueden ser conocidos por el *sujeto* previamente a su ejecución. Estos elementos externos al *sujeto* que participan en la *actividad* pueden ser *herramientas* u *objetos*. Por tanto, el elemento sobre el que se produce la reflexión se representa mediante un *artefacto*, que puede ser tanto una *herramienta* como un *objeto*. El diagrama resultante incluye al *sujeto* que realiza la *actividad*, la propia *actividad* y un *artefacto* que usa ésta. La *contradicción* se refleja en la no existencia de una *actividad* previa que compruebe la adecuación del *artefacto* para la *actividad* que se va a realizar sobre él. Esta *actividad* previa representaría la reflexión sobre la *actividad* que se va a realizar. El diagrama resultante para la *Reflexión sobre la Actividad* se ve en la Fig. 64.

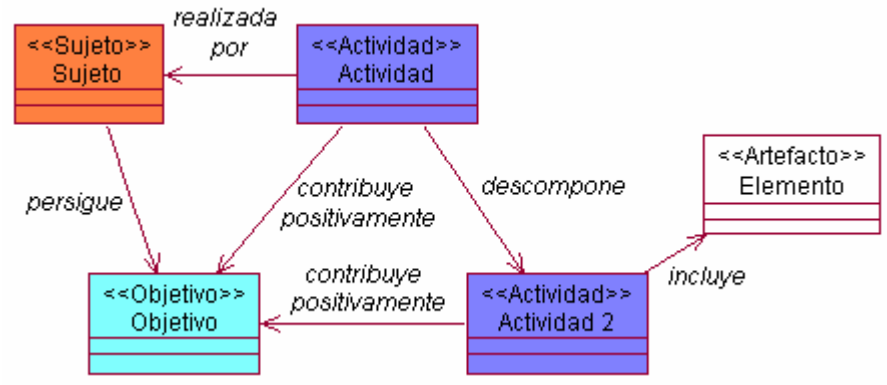


Fig. 64. Contradicción de *Reflexión sobre la Actividad*.

La *contradicción* en la *Reflexión sobre la Actividad* surge cuando el *sujeto* obvia el análisis mental de la *actividad* previo a la realización de (ver Fig. 64). De esta forma está perdiendo posibilidades de mejorar su rendimiento en el cumplimiento de su agenda.

5.4.5.2. Soluciones de la Reflexión sobre la Actividad

La solución a la *contradicción* de la *Reflexión sobre la Actividad* pasa por hacer al *sujeto* consciente de sus posibilidades de análisis sobre la *actividad* que va a comenzar. Evidentemente, el *sujeto* no puede anticipar todas las contingencias de la ejecución de la *actividad* pero debe considerar aquellas que pueda conocer. Así, si una serie de *artefactos* han de estar presentes para llevar a cabo la *actividad* y sus condiciones pueden ser constatada por el *sujeto* a priori, entonces el *sujeto* ha de realizar ese estudio (ver Fig. 65). Este análisis previo de los *artefactos* de la *actividad* es representado con la *Actividad 1*, que es una *actividad* de reflexión. Si los *artefactos* examinados resultan adecuados se puede considerar la ejecución del resto de la *actividad*. En caso contrario no tiene sentido para el *sujeto* comenzar una *actividad* que no puede completar.

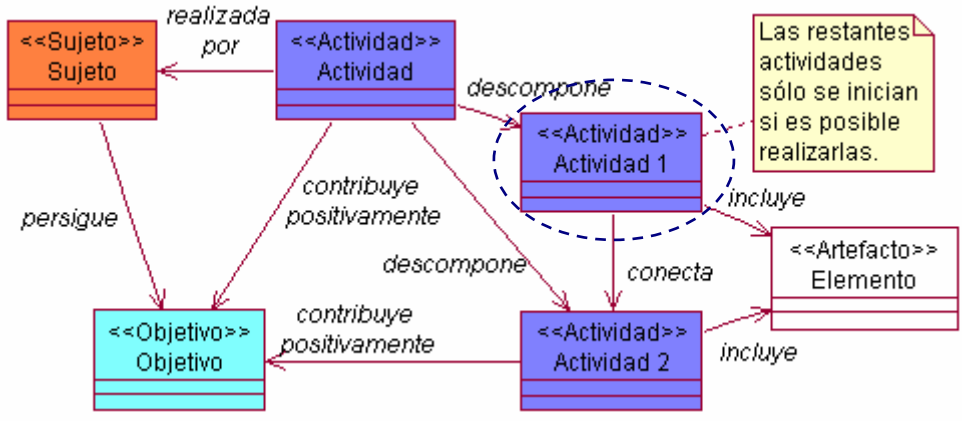


Fig. 65. Una posible solución a la contradicción de *Reflexión sobre la Actividad*.

5.4.6. Interrupción de Uso

El término *Interrupción de Uso* aparece en la literatura de TA frecuentemente relacionado con el desarrollo de sistemas de información. La expresión fue acuñada en [Winograd & Flores 1986] para hacer referencia a aquellas situaciones en las que el flujo normal de la *actividad* se ve interrumpido por algún motivo, por ejemplo el mal funcionamiento de una *herramienta*. Estas interrupciones son importantes en el estudio de la *actividad* porque suponen que el foco de atención de un actor cambia, generando un punto de apertura para el aprendizaje [Bødker & Grønbæk 1996].

El aprendizaje originado por situaciones de interrupción de uso no tiene que ser necesariamente casual, también puede surgir de acciones deliberadas por parte de los *sujetos*. La *actividad* es sensorial y práctica y en ella los *sujetos* entran en contacto con los *objetos* del mundo que les rodean, los estudian y actúan sobre ellos reconociendo sus propiedades objetivas [Marx & Engels 1844]. Incorporar conscientemente el aprendizaje a raíz de una interrupción de uso supone dotar al *sujeto* de las tareas para analizar *a posteriori* los fallos en la ejecución de sus *actividades*. Los elementos utilizables en este análisis son los resultados de dichas *actividades* y los artefactos que se han usado en ellas. Por tanto, las interrupciones de uso en la *actividad* pueden originar reflexiones sobre sus *artefactos* a fin de resolver sus contradicciones internas. La ausencia de esta reflexión constituye la contradicción, ya que el *sujeto* desperdicia una oportunidad de aprender y mejorar su desempeño de la *actividad*.

Tanto la *contradicción* de la *Reflexión sobre la Actividad* como la de *Interrupción de Uso* se basan en la ausencia de un ciclo de reflexión sobre la *actividad*. En la *Reflexión sobre la Actividad* se examinan antes de la ejecución los *artefactos* de la *actividad*. En la *Interrupción de Uso* se examinan tras la ejecución los *artefactos* de la *actividad* para aprender de su fallo.

La *contradicción* de la *Interrupción de Uso* es una situación en la que el *sujeto* no realiza un ciclo de reflexión acerca de las causas que han originado un fallo en la

satisfacción de sus *objetivos*. En ausencia de esta reflexión, el *sujeto* no puede realizar el aprendizaje que le permitiría posiblemente diagnosticar y corregir el fallo en ocasiones posteriores.

5.4.6.1. Representación UML-TA de la Interrupción de Uso

La *contradicción* de la *Interrupción de Uso* aborda el estudio de los resultados de una *actividad* que ha fallado en alcanzar sus *objetivos*. Los elementos iniciales en esta descripción del contexto son la *actividad* y el *objetivo*, unidos por una relación *falla* que indica la no satisfacción. Los demás elementos que pueden formar parte del análisis de la *actividad* son sus *herramientas* y *objetos*. Para representarlos genéricamente se emplea el *rol artefacto*, al igual que se hizo en la *contradicción* de la *Reflexión sobre la Actividad*. Finalmente, el diagrama incluye al *sujeto* que realiza la *actividad* y la reflexión sobre ella. La *contradicción* emerge de la ausencia de la *actividad* de estudio del fallo. La representación de la *Interrupción de Uso* queda como se ve en la Fig. 66.

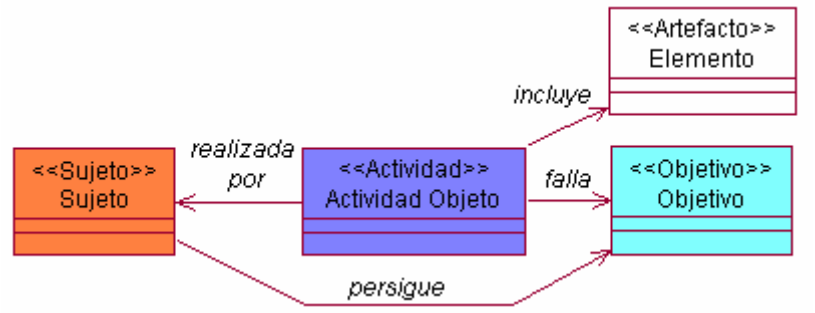


Fig. 66. Contradicción de *Interrupción de Uso*.

La *Interrupción de Uso* surge cuando se altera el desarrollo normal de la *actividad*, aquél que la llevaría a satisfacer las necesidades de su *sujeto*. La no satisfacción de los *objetivos* de la *actividad* debe producir un ciclo de reflexión en el *sujeto*, que incluye el diagnóstico del problema y su corrección. La *contradicción* surge cuando el *sujeto* carece de la información sobre la ejecución de la *actividad* que necesita para dicho estudio. Es decir, el *sujeto* sólo sabe que sus acciones no han alcanzado el resultado deseado pero no dispone de información adicional (e.g. cuál ha sido el desarrollo de la *actividad* o el comportamiento de sus *artefactos*) o no puede reflexionar sobre dicho fallo. Las situaciones de esta clase generan un estado en el que el *sujeto* es incapaz de resolver las contradicciones generadas por sus *actividades*, pudiendo degenerar en *Dobles Vínculos* (ver contradicción 5.4.4 en este capítulo) si se hacen crónicas.

5.4.6.2. Soluciones de la Interrupción de Uso

La solución de la *Interrupción de Uso* consiste en que el *sujeto* adquiera la capacidad para reflexionar sobre el resultado de sus *actividades*, o al menos, sobre aquellas cuyo flujo normal puede ser alterado. Esta capacidad de reflexión puede ser representada como una *actividad* adicional, i.e. la *Actividad de Reflexión*, en la que el *sujeto* puede analizar los *artefactos* y *objetivos* de la *actividad* que falló para tomar las

medidas que estime oportunas. La solución podría quedar como la representada en la Fig. 67.

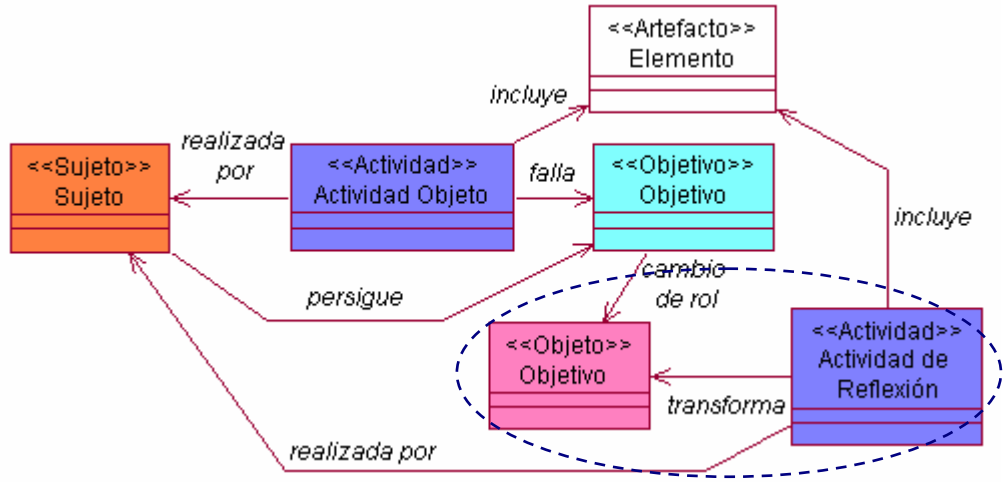


Fig. 67. Una posible solución a la contradicción de *Interrupción de Uso*.

La *Actividad de Reflexión* es sólo una indicación del lugar del flujo de trabajo donde debe comenzar el proceso de diagnóstico y aprendizaje sobre el fallo de la *Actividad Objeto*. Por supuesto, resolver completamente el ciclo de reflexión necesitará un mayor refinamiento que dependerá del contexto total del sistema.

5.4.7. Paradoja de Planificación

La *Paradoja de Planificación* [Suchman 1987, Bardram 1997] emerge de la contradicción entre la visión de los planes como anticipación ideal de las acciones a realizar y su ejecución en el contexto real.

Frecuentemente los planes son vistos como formas de canalizar información entre usuarios y especificar automáticamente las acciones a tomar con esa información. Esta visión de los planes se describe típicamente mediante flujos de trabajo que ayudan a definir, ejecutar, coordinar y monitorizar el trabajo en una organización. La representación de estos flujos de trabajo se suele hacer mediante la descomposición jerárquica o secuencial de la *actividad* en tareas, donde las salidas producidas por unas tareas son las entradas de otras.

La visión de los planes como flujos de trabajo no puede sin embargo prever los fallos y excepciones en su aplicación real. Es decir, los planes son anticipaciones racionales e ideales de la *actividad* a realizar, pero se llevan a acabo mediante *operaciones*[4] situadas en un contexto determinado que pueden ser consideradas como improvisaciones *ad hoc*.

[4] Las *operaciones* fueron introducidas en la sección 3.2.2 como parte de la descomposición jerárquica de una *actividad*, pero no son contempladas en el lenguaje UML-TA.

De esta doble naturaleza, como elemento ideal de realización contextualizada, surge la *Paradoja de Planificación*. Por una parte, debido a las contingencias que aparecen en el contexto concreto del trabajo, el plan tiene una naturaleza improvisada y adaptada a las circunstancias. Por otra parte, el plan sirve como un artefacto del razonamiento para anticipar los resultados de la *actividad* y permitir su posterior reconstrucción y la reflexión sobre él.

La *Paradoja de Planificación* aparece cuando los planes sólo contemplan una de sus dimensiones, la de descripción ideal de un flujo de trabajo. Aunque no se pueden prever todas las vicisitudes de la ejecución real del plan, si puede hacerse evolucionar éste para contemplar los contratiempos que se sabe que pueden llegar a surgir. Estos posibles contratiempos han de ser incorporados al plan, a fin de poder corregirlos durante la ejecución.

Un caso particular de la *Paradoja de Planificación* corresponde a planes cuya ejecución puede ser abortada por errores [Bardram 1997]. En esta situación, el plan sólo se ha ejecutado parcialmente pero ya puede haber alterado el estado de algunos de los *artefactos* que usa. Para contemplar este fallo y adoptar un curso alternativo de acción, una solución genérica es restaurar el estado previo al fallo si ello es posible. Existirá una contradicción si el plan no contempla ninguna medida frente al fallo.

5.4.7.1. Representación UML-TA de la Paradoja de Planificación

La *contradicción* de la *Paradoja de Planificación* se centra en el hecho de que los planes han de contemplar los contratiempos posibles en su flujo normal de ejecución. Uno de estos casos [Bardram 1997] corresponde a la ejecución abortada de planes y la posterior restauración de un estado previo para poder llevar a cabo un curso alternativo de acción en el plan.

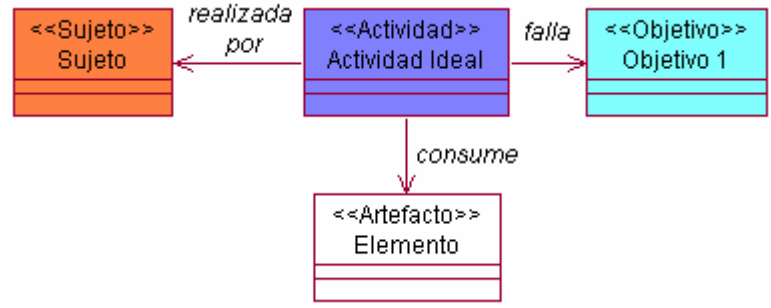


Fig. 68. *Paradoja de Planificación*.

La *contradicción* comienza con una *actividad* abortada. La no finalización de la *actividad* es representada en este caso mediante el fallo en la satisfacción de su *objetivo*. Otra posibilidad hubiera sido mostrar la *actividad* y su *objetivo* descompuestos en dos elementos donde sólo el primero ha tenido éxito. El otro aspecto clave de la *contradicción* es que la *actividad* altera el estado de algunos de los elementos que usa. Estos elementos pueden ser *objetos* o *herramientas* por lo que se representan mediante el *rol artefacto*. La alteración del estado podría reflejarse simplemente con una relación *incluye* desde la *actividad* al *artefacto*. Sin embargo, a fin de hacer el cambio más evidente (especialmente en la solución de la

contradicción) se ha optado por usar la relación *consume*. La relación *consume* describe el uso de una *herramienta* o la transformación de un *objeto* y la posterior destrucción de ese *artefacto*. La *contradicción* se refleja entonces por la ausencia de *actividades* capaces de restaurar el *artefacto* eliminado. La representación de la *contradicción* de la *Paradoja de Planificación* puede verse en la Fig. 68.

Una *Paradoja de Planificación* corresponde a un escenario en el que los planes son usados simultáneamente como modelo ideal de la *actividad* y de su ejecución real y, sin embargo, uno de esos aspectos no es correctamente contemplado. Este sería el caso cuando una *actividad* no considera la situación de un fallo parcial. Si la *actividad* ha de satisfacer varios *objetivos* que descomponen otro global, un fallo parcial en la satisfacción de dichos *objetivos* ha de ser contemplado en el plan. Una forma de considerarlo es ejecutar cursos alternativos de acción en el plan. Sin embargo, estos cursos pueden requerir un cierto contexto que ha sido alterado por la ejecución fallida. Este contexto incluye algunos *artefactos* del *sistema de actividad*, concretamente las *herramientas* usadas y los *objetos* transformados. Por tanto, para poder ejecutar la alternativa, el *sujeto* debería disponer de mecanismos para restaurar el estado previo. En el caso de que no cuente con ellos aparece la paradoja.

5.4.7.2. Soluciones de la Paradoja de Planificación

En la forma en la que ha sido descrita, la paradoja aparece porque no se contempla adecuadamente la posibilidad de que la ejecución ideal de la *actividad* falle. La solución más simple a la formulación anterior de la *Paradoja de Planificación* es proporcionar al *sujeto* los mecanismos necesarios para restaurar el estado en el caso de que lo juzgue necesario. Al regenerar el estado anterior con la *Actividad de Restauración*, el *sujeto* es capaz de hacer frente a las vicisitudes originadas por la ejecución real de la *actividad*. El modelo modificado quedaría como se ve en la Fig. 69.

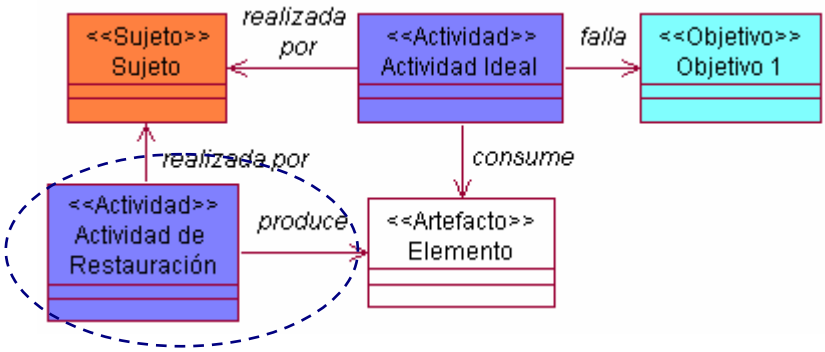


Fig. 69. Una posible solución a la *Paradoja de Planificación*.

La solución a la *Paradoja de Planificación* de la Fig. 69 mediante una *Actividad de Restauración* no es evidentemente una solución universal. Frecuentemente el estado anterior a la ejecución fallida no puede ser restaurado. En tales casos, los cursos alternativos de acción han de hacer uso de recursos diferentes o bien la *Actividad Ideal* ha de contemplar la creación de recursos que permitan recuperar el estado previo.

5.4.8. Estado de Necesidad

Las *actividades* surgen para satisfacer las necesidades de los *sujetos*. Sin embargo, dichas necesidades no son el motor directo de las *actividades*, este papel lo desempeñan los *motivos*. Los *motivos* surgen a partir de las necesidades cuando éstas encuentran un *objeto* cuya transformación puede satisfacerlas y a partir de ese encuentro brota la *actividad* [Leontiev 1978].

La relación entre la *actividad* y sus *motivos* no es estática en el tiempo. Dada la naturaleza histórica del *desarrollo* de la actividad, sus componentes están sujetos a evolución. Generalmente se produce un ciclo de generación de nuevas necesidades expandidas a partir de la *actividad* existente. Este ciclo puede verse interrumpido cuando los medios de la *actividad* no se corresponden ya con las necesidades previas o bien las necesidades no pueden ser satisfechas con los medios disponibles. Se produce entonces un estado de indeterminación del *sujeto* en el cual sus *objetivos*, tal y como estaban concebidos, pierden su *objeto*. Este estado de indeterminación recibe el nombre de *Estado de Necesidad*.

El *Estado de Necesidad* del *sujeto* [Bratus & Lishin 1983] se caracteriza por la perdida de los *motivos* de sus *actividades*, entendidos estos como asociaciones necesidad-*objeto*. Un aspecto esencial del *Estado de Necesidad* es que el *sujeto* se enfrenta a múltiples alternativas sin poder determinar la dirección correcta de sus nuevos esfuerzos.

El *Estado de Necesidad* es una situación temporal en esencia. Aunque la *actividad* original ya no se corresponde con su *motivo*, el *sujeto* encuentra en algún momento un nuevo *objeto* para su necesidad. Este nuevo *objeto* coloca el *Estado de Necesidad* en una situación en la que ha vuelto a encontrar un *motivo*. Este nuevo *motivo* es el que impulsa el desarrollo de la nueva *actividad*.

La *contradicción* del *Estado de Necesidad* puede definirse como una situación en la que el contexto necesario para que una *actividad* satisfaga sus *objetivos* es alterado. Este contexto es representado por el *objeto* de dicha *actividad*. En estas condiciones la *actividad* no puede garantizar ya que su *producto* cubre las necesidades del *sujeto*.

5.4.8.1. Representación UML-TA del Estado de Necesidad

La *contradicción* del *Estado de Necesidad* refleja una situación en la que el contexto de una *actividad* es alterado externamente de una forma no prevista. Esta alteración hace que el proceso de transformación de la *actividad* cambie. Esta alteración se plasma en el resultado de la *actividad*, i.e. su *producto*.

La *contradicción* parte por tanto de una *actividad*, i.e. la *Actividad Antigua*, que genera un *producto* que satisface un *objetivo* de su *sujeto*. Sobre esta *actividad* surgen modificaciones externas cuyo resultado son cambios en el *producto*. La forma de representar estas modificaciones es mediante otra *actividad*, i.e. la *Actividad Externa*, que modifica el *producto* de la primera. Para representar la transformación, el *producto* de la primera *actividad* deviene en *objeto* de la externa mediante una relación de *cambio de rol*. La *contradicción* radica en que el *producto* alterado ya no garantiza la satisfacción del *objetivo*. La *contradicción* del *Estado de Necesidad* con estos elementos se muestra en la Fig. 70.

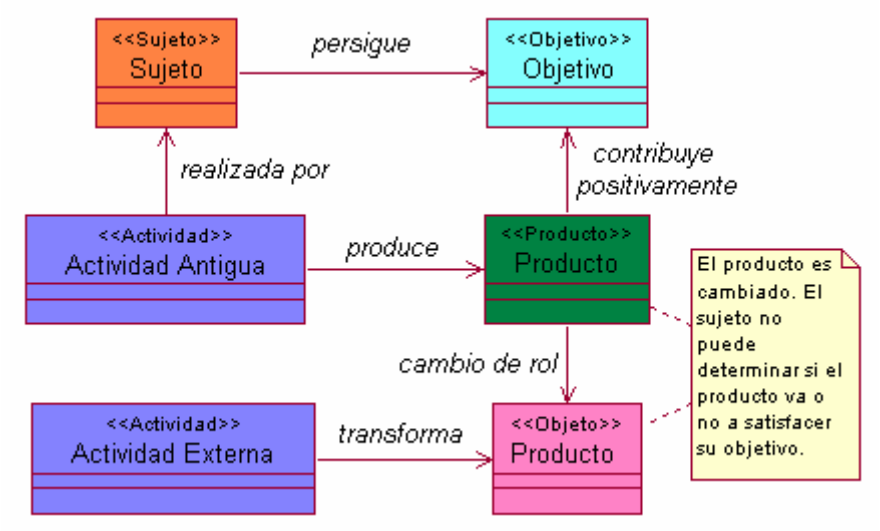


Fig. 70. Contradicción del *Estado de Necesidad*.

El *Estado de Necesidad* es la situación en la que el *sujeto* debe satisfacer un *objetivo*, pero la *Actividad Antigua* con la que cuenta para hacerlo no le garantiza el resultado. No se trata de una situación temporal para esa *actividad*, por ejemplo debida a una configuración concreta del entorno, sino crónica. El *sujeto* se ve por tanto en la imposibilidad de prever razonablemente cuándo podrá satisfacer sus necesidades.

5.4.8.2. Soluciones del Estado de Necesidad

La solución del *Estado de Necesidad* pasa por la elaboración de alternativas para satisfacer los *objetivos* del *sujeto*. Las *actividades* disponibles no garantizan ya una probabilidad de cumplimiento adecuada para los *objetivos*. Por ello es necesario suplir al *sujeto* con nuevas *actividades* que le permitan satisfacer esas necesidades o al menos incrementar la posibilidad de su cumplimiento. La propuesta quedaría representada por el diagrama de la Fig. 71. En este diagrama se introduce la *Actividad Nueva* para dar al *sujeto* una alternativa en la satisfacción de sus necesidades representadas por el *objetivo*.

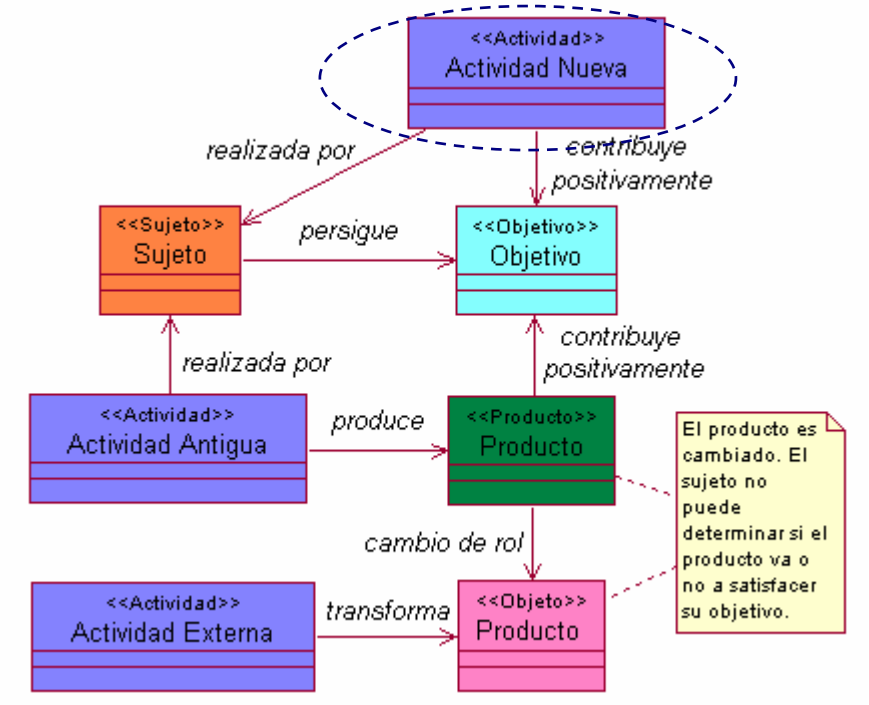


Fig. 71. Una posible solución del *Estado de Necesidad*.

5.4.9. Conflicto Productor-Usuario

Los conflictos entre productores y usuarios de *artefactos* en una red de *actividades* son un ejemplo de contradicciones cuaternarias [Engeström 1987]. Esta clase de contradicciones se presentan entre la *actividad central* y aquellas que producen, consumen o modifican los *artefactos* de ésta[5]. En esta contradicción, el conflicto surge entre la *actividad* que genera una *herramienta* y aquella que la utiliza. Un ejemplo ilustrativo de estas contradicciones es la etapa de transferencia de innovaciones tecnológicas de los desarrolladores a los usuarios finales [Hasu & Engeström 2000]. En la *actividad* de los productores, el *artefacto* juega el rol de *objeto* a transformar y de *producto* resultante. Por el contrario, en la de los usuarios el *artefacto* actúa como una *herramienta*. La mayor parte de los análisis sobre estas situaciones asumen la simplificación de que todos los actores involucrados comparten unos valores e intereses comunes, por lo que fallan en estudiar las interacciones reales en sus contextos específicos. La consecuencia de esta forma de actuar es que los estudios sobre las innovaciones son insensibles a la forma en que la tecnología puede

[5] Los conceptos de *actividad central* y *actividades vecinas* fueron presentados en la sección 3.2.3 de la introducción a la TA.

alterar la estructura organizativa y ocupacional del trabajo [Barley 1986]. Esta "insensibilidad" se plasma en el aislamiento de las *actividades* involucradas y evita que se perciba el carácter polifacético del *artefacto*. De esta forma, los actores carecen de la habilidad para razonar metacomunicativamente sobre dicho *artefacto*, es decir, discriminar sus facetas como *objeto-producto* y *herramienta*. Esta discriminación es no obstante imprescindible para la reflexión y expansión constructiva sobre las *actividades* y sus *artefactos*.

La *contradicción* del *Conflicto Productor-Usuario* aparece en una red de *actividades* entre la *actividad central* y una *actividad* productora de *herramientas*. Surge cuando dichas *actividades* no tienen en cuenta el papel que el *artefacto* que comparten juega en la *actividad vecina* para su correspondiente *sujeto*.

5.4.9.1. Representación UML-TA del Conflicto Productor-Usuario

La *contradicción* del *Conflicto Productor-Usuario* surge entre dos *actividades* vecinas, una de las cuales produce la *herramienta* que usa la otra. Este elemento es el *Artefacto* que se comparte en el diagrama. En la actividad productora el *Artefacto* juega el *rol* de *objeto*, aunque también podría modelarse como *producto*. En la actividad que lo usa, el *Artefacto* juega el *rol* de *herramienta*. Esta transición se representa mediante una relación de *cambio de rol*. El otro aspecto importante de la *contradicción* es la incapacidad que tienen los *sujetos* para discriminar las distintas facetas del *Artefacto*, es decir, el papel que juega en la *actividad vecina*. Estos papeles son representados por los *objetivos* de las *actividades*. La incomunicación, y con ella la *contradicción*, se plasma en la inconexión de las *actividades* salvo por el *Artefacto* compartido. Ninguna de las *actividades* es consciente de las necesidades existentes en la otra *actividad*. La representación resultante del *Conflicto Productor-Usuario* puede verse en la Fig. 72.

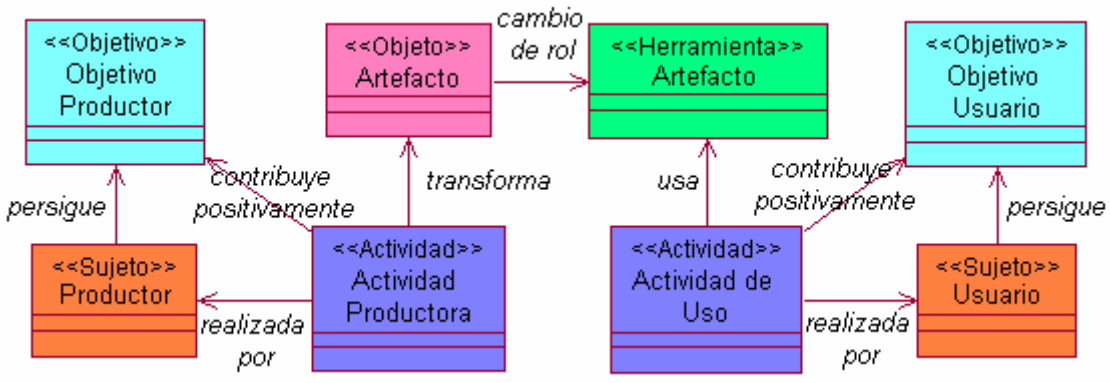


Fig. 72. *Conflicto Productor Usuario.*

Un *Conflicto Productor-Usuario* surge cuando un *artefacto* es compartido por varias *actividades*, siendo en unas el *objeto* de la *actividad* y en otras una *herramienta*. Según la TA, una de las características más relevantes de la *actividad* es su orientación al *objeto* [Vygotsky 1978, Leontiev 1978]: el *objeto* es el verdadero motivo de la *actividad*. Por ello, estas *actividades vecinas* dirigidas hacia *objetos*

diferentes, tienen *objetivos* diferentes que no consideran necesariamente los *objetivos* de las demás *actividades*. Esta ignorancia sobre los demás objetivos a los que sirve el *artefacto* origina la contradicción reflejada en la Fig. 72.

5.4.9.2. Soluciones del Conflicto Productor-Usuario

Una posible solución al *Conflicto Productor-Usuario* es hacer que las *actividades vecinas* tomen conciencia del papel que juega su artefacto en las demás *actividades*. Así la *actividad* productora del *Artefacto*, donde éste juega el papel de *objeto*, ha de incorporar intencionalidad relativa a crear un *artefacto* que satisfaga las necesidades como *herramienta* en la *actividad* consumidora. Por otra parte, la *actividad* consumidora puede realizar un uso del *Artefacto* que permita hacer evolucionar su construcción. Esta intencionalidad se representa mediante nuevos *objetivos*: la *Actividad Productora* satisface un *objetivo* para el *Usuario* y la *Actividad de Uso* satisface un *objetivo* para el *Productor*. Las *actividades* conscientes de su posición en la red de *actividades* podrían representarse como se ve en la Fig. 73.

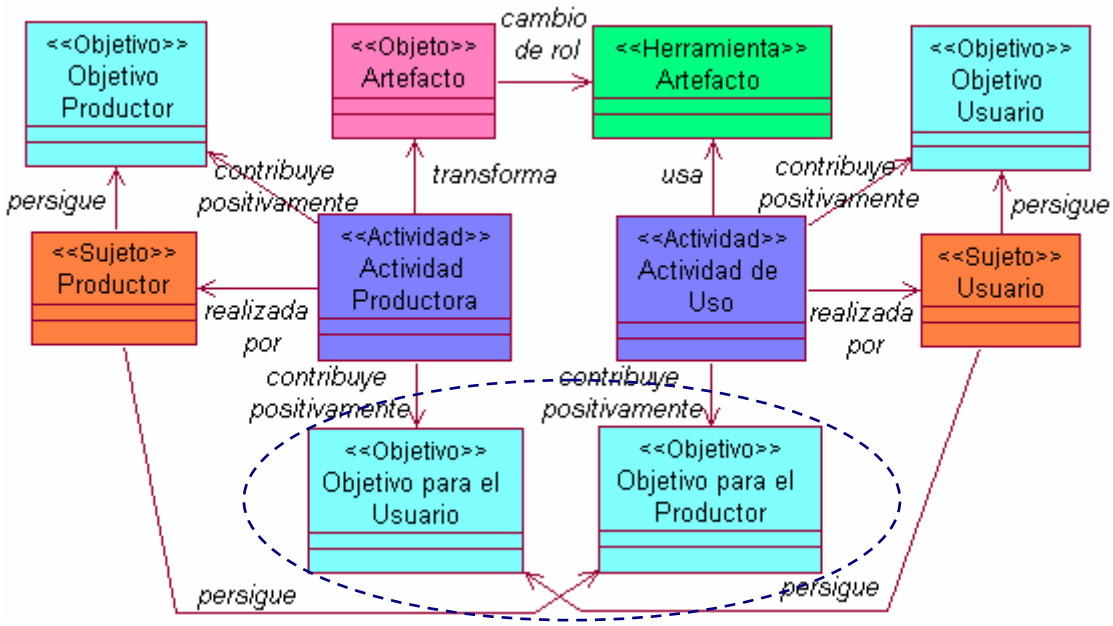


Fig. 73. Una posible solución del *Conflicto Productor-Usuario*.

5.4.10. Información Contradictoria

El aprendizaje es una *actividad* destinada a la transformación del propio *sujeto*, concretamente a su transformación psíquica [Davydov 1988]. Otras *actividades* son también capaces de producir aprendizaje pero no es éste su *objetivo*. Una característica esencial de la *actividad* del aprendizaje es que la adquisición de nuevo conocimiento y habilidades, requiere ciertos prerrequisitos en términos de

habilidades, motivación y estructuras cognitivas del *sujeto*. Sin embargo, éstas sólo están desarrolladas parcialmente cuando son necesarias. Si estuvieran completamente desarrolladas el aprendizaje no sería necesario; si no estuvieran desarrolladas en absoluto, el aprendizaje no sería posible.

En el aprendizaje colaborativo [Johnson & Johnson 1975], los *sujetos* tratan de construir un *artefacto* común que represente su conocimiento sobre una situación dada. Tanto el conocimiento individual de los *sujetos* como el que almacena el *artefacto* compartido pueden estar en dos posibles estados: en conflicto o coherente. A nivel individual, el *sujeto* construye nuevo conocimiento integrando nueva información en sus estructuras cognitivas. Cuando la nueva información contradice el conocimiento existente, se dice que ocurre un conflicto. El *sujeto* debe entonces reconciliar el conflicto, quizás modificando sus estructuras cognitivas. A nivel del grupo, cuando varios *sujetos* están en desacuerdo con la información en el espacio de trabajo compartido también existe un conflicto. En este caso los *sujetos* deben negociar para decidir cómo deben representarse las diferentes perspectivas en el *artefacto* compartido. Se han de identificar los puntos de conflicto y reconciliarlos mediante la discusión.

El problema de la *Información Contradictoria* [Miao *et al.* 2000] surge en el aprendizaje colaborativo cuando el *artefacto* compartido no proporciona el soporte adecuado. Esencialmente el *artefacto* ha de proporcionar soporte para representar el conocimiento tratado, para explorarlo y para negociar sobre él.

5.4.10.1. Representación UML-TA de la Información Contradictoria

La *contradicción* de la *Información Contradictoria* surge cuando el *artefacto* empleado como soporte del aprendizaje colaborativo no permite la exploración o transformación del conocimiento de la *comunidad.* Esta incapacidad adopta en ocasiones la forma de un acceso restringido al *artefacto* compartido. Esa es la versión que adopta este patrón de detección.

El patrón describe una *comunidad* de aprendizaje cuyos *sujetos* modifican el *artefacto* compartido. La modificación del *artefacto* se representa como *actividades* que trabajan sobre dicho *artefacto*. Puesto que el *artefacto* es transformado en el proceso se ha de modelar como un *objeto* de la TA, i.e. el *Objeto Compartido*. Para representar la incapacidad de modificar parte de la información, se representa el *objeto* descompuesto en varios *objetos*. La posibilidad de que la información representada por estos *objetos* sea inconsistente (ver [Miao *et al.* 2000]) se representa mediante una relación de contribución *indefinida*, aunque ésta no es esencial para la contradicción. Cada uno de estos *objetos* sólo es modificado por una *actividad*, por lo que permanece inaccesible a las restantes. Por último, la introducción de las *actividades* lleva a representar la *comunidad* a través de los *sujetos* que la componen y que llevan a cabo las *actividades*. El diagrama correspondiente a la *contradicción* de la *Información Contradictoria* puede verse en la Fig. 74.

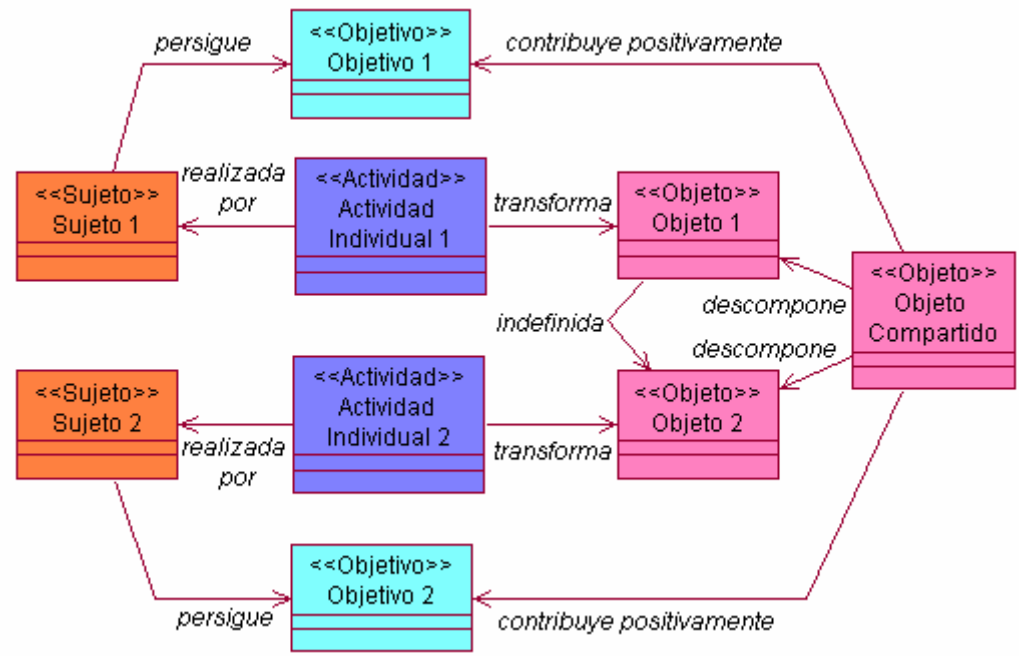


Fig. 74. *Información Contradictoria.*

El problema de la *Información Contradictoria* comprende cualquier fallo en el soporte al aprendizaje que proporciona el *Objeto Compartido* por la *comunidad*. Para este patrón nos centramos en la existencia de *actividades* que posibiliten la representación y negociación sobre dicho *objeto*. En este sentido, el problema de la *Información Contradictoria* aparece cuando los *sujetos* deben crear un *objeto* compartido y sin embargo no existen *actividades* comunes para analizar las diferentes perspectivas y construirlo colaborativamente. Es decir, los *sujetos* se reparten la construcción del *objeto* pero con una visión individual y no colectiva, de manera que el *objeto* es la mera adición de las aportaciones de los *sujetos*. Esta situación queda reflejada en la Fig. 74.

5.4.10.2. Soluciones de la Información Contradictoria

El problema de la *Información Contradictoria*, tal y como se ha modelado, es que no existen *actividades* de construcción colaborativa del *objeto*. Los *sujetos* sólo comparten el soporte donde expresan su conocimiento, pero no hay análisis y síntesis de visiones dispares sobre dicho conocimiento. Para lograr esta integración ha de haber discusión sobre el *artefacto* común y las distintas perspectivas. Dicha negociación se representa con la existencia de una nueva *actividad*, realizable por todos los *sujetos* de la *comunidad*, donde se contraste la información disponible y las perspectivas contradictorias. La nueva situación es la que se presenta en la Fig. 75. La *actividad* compartida por los miembros de la *comunidad* de aprendizaje es la *Construcción Colaborativa*.

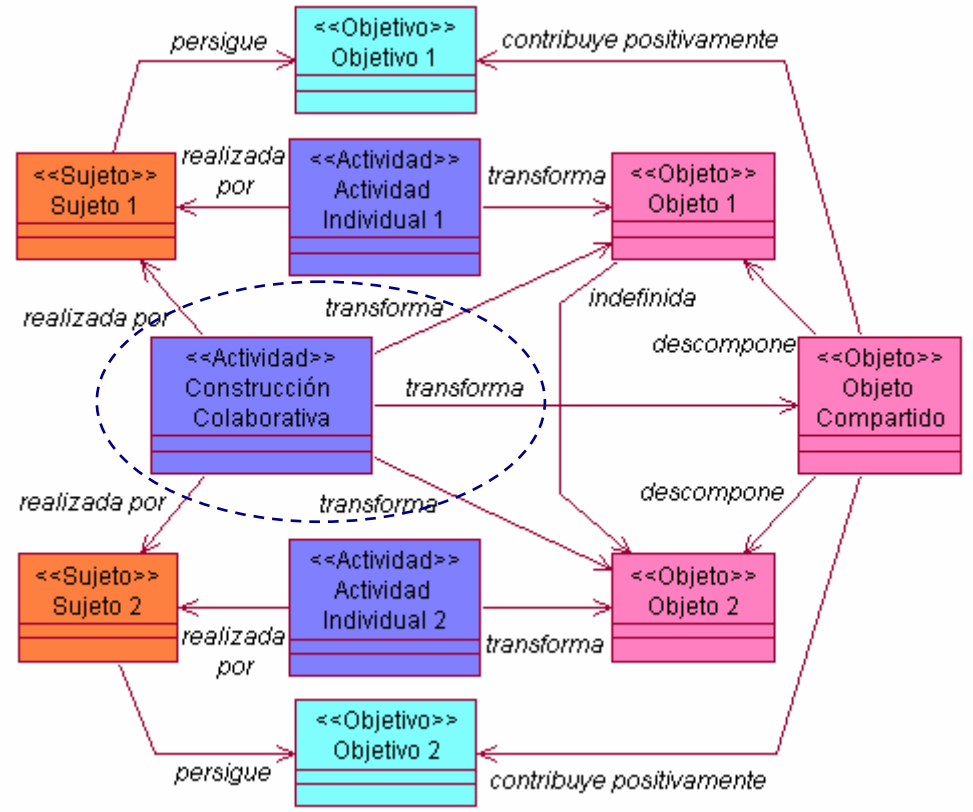


Fig. 75. Una posible solución a la *Información Contradictoria*.

El patrón de solución no resuelve el problema de cómo se realiza la negociación entre los usuarios o la forma de representar su conocimiento compartido. Tan sólo indica la necesidad de contar con la *actividad* de *Construcción Colaborativa* del *Objeto Compartido*. Resolver los aspectos abiertos es trabajo de refinamientos ulteriores.

5.5. Ejemplo de uso de contradicciones de la TA

El uso de las contradicciones para la mejora y corrección de especificaciones, se muestra con un ejemplo de sistema de recomendación basado en el filtrado colaborativo y descrito en [Gómez-Sanz *et al.* 2003]. La especificación resultante de este problema en INGENIAS puede encontrarse en *http://ingenias.sourceforge.net*.

Un sistema de filtrado colaborativo asume que si un usuario encuentra interesante una información, entonces otros usuarios con opiniones y preferencias similares pueden encontrar también de interés esa información. El filtrado colaborativo o social automatiza el proceso de compartición de la información entre los miembros de la

comunidad de interés. La similitud entre usuarios puede obtenerse por medio de técnicas estándares para calcular la correlación estadística.

El sistema que se presenta en [Gómez-Sanz *et al.* 2003] incluye dos tipos de agentes:

- *Agentes de la Comunidad*. Representan a un grupo de usuarios que comparten intereses comunes. Son los responsables de la administración de usuarios y de la difusión de la información.
- *Agentes Personales*. Representan a los individuos que pertenecen a la comunidad. Un agente personal puede estar suscrito a varias comunidades. Este tipo de agente es la interfaz entre los usuarios y la comunidad en los procesos de búsqueda y colaboración.

Un *Agente Personal* juega varios roles en el sistema. Dos de estos roles están intrínsecamente relacionados con el propio proceso de intercambio de información colaborativo. Son los roles que pueden verse en la Fig. 76: *Informador*, el que hace la sugerencia a la comunidad; *Evaluador*, el que valora la sugerencia de un miembro de la comunidad.

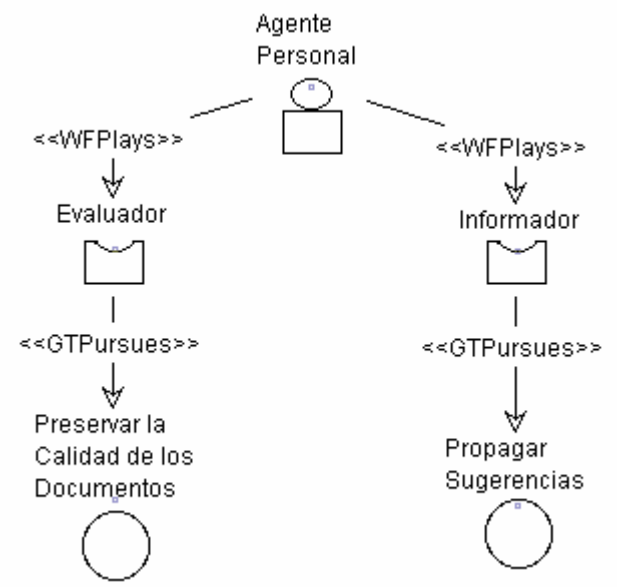


Fig. 76. *Agente Personal* con roles y objetivos.

Un *Agente Personal* tiene el objetivo de *Propagar Sugerencias*, que es perseguido por el rol *Informador*. Este objetivo es necesario para lograr que los agentes suministren información a la comunidad de usuarios. Además, como miembro de la comunidad, el agente está interesado en que la información que se maneja sea de calidad. Este interés queda representado por el rol *Evaluador* que persigue el objetivo *Preservar la Calidad de los Documentos*. Estos objetivos no pertenecen a la descomposición de un objetivo común del sistema. Su relación procede de los flujos de trabajo que proporcionan una sugerencia a la comunidad y la evalúan. Dichos

flujos de trabajo son mostrados en la Fig. 77 como una cadena productor-consumidor de tareas y entidades mentales de los agentes, que son lo que se produce y consume. La Fig. 77 agrupa varios diagramas de INGENIAS para su mejor visualización. Los iconos de INGENIAS se han sustituido por estereotipos UML.

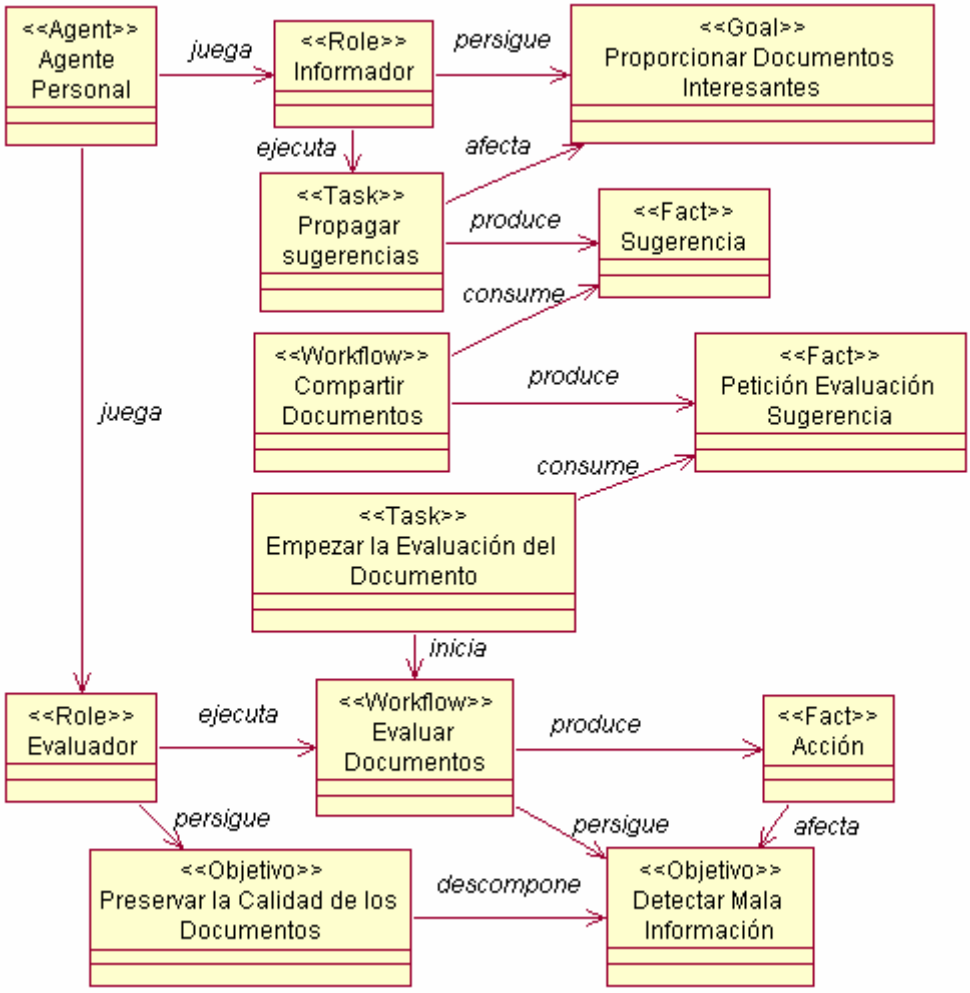


Fig. 77. Flujos de trabajo que relacionan los objetivos de los roles *Evaluador* e *Informador*.

El flujo de trabajo representado en la Fig. 77 comienza con el *Agente Personal* jugando el rol de *Informador*. El *Informador* realiza una *Sugerencia* a la comunidad. Dicha *Sugerencia* representa su contribución a la información de la comunidad. Al tratarse de un sistema de filtrado colaborativo, los miembros de la comunidad deben evaluar la *Sugerencia*, a fin de saber si representa información relevante de acuerdo con la temática del grupo. Para realizar la evaluación, el *Agente de la Comunidad* ejecuta el flujo de trabajo *Compartir Documentos* donde selecciona los evaluadores

de la *Sugerencia* y les proporciona información acerca de ella. Aquellos miembros de la comunidad que juegan el papel de *Evaluadores* ejecutan la tarea *Empezar la Evaluación del Documento* que inicia el flujo *Evaluar Documento*. Los *Agentes Personales* que ejecutan estas tareas están satisfaciendo su objetivo de *Preservar la Calidad de los Documentos* a través del objetivo *Detectar Mala Información*. El resultado del flujo de trabajo es el hecho *Acción*, el cual usa el *Agente de la Comunidad* como base para elaborar la evaluación final.

Esta aproximación a la evaluación fue el resultado de suponer que los usuarios podrían ajustar sus recomendaciones a los temas de la comunidad simplemente estudiando la información intercambiada en ella. Los resultados del prototipo que implementó esta política no fueron los esperados. Muchos usuarios nuevos no eran capaces de continuar como miembros de la comunidad a pesar de su interés. Sus recomendaciones eran siempre rechazadas pero ellos no conocían las razones. Por supuesto, el problema era que no tenían suficiente información acerca de los resultados del proceso de evaluación, pero el lugar de los modelos donde se debían realizar los cambios no estaba claro.

Para detectar cuál era exactamente el problema, se uso el proceso basado en contradicciones. Para ello el primer paso fue traducir las especificaciones representadas en los diagramas Fig. 76 y Fig. 77 a UML-TA. Para ello se usó el proceso de traducción y las correspondencias vistas en la sección 3.5. A modo de resumen la Tabla 16 recoge la traducción de los elementos.

Teoría de Actividad	INGENIAS
Sistema de Actividad	Propagar Sugerencias, Compartir Documentos, Empezar la Evaluación del Documento, Evaluar Documentos
Actividad	Propagar Sugerencias, Compartir Documentos, Empezar la Evaluación del Documento, Evaluar Documentos
Sujeto	Informador, Agente Personal, Evaluador
Objeto	Sugerencia, Petición Evaluación Sugerencia, Acción
Producto	Sugerencia, Petición Evaluación Sugerencia, Acción
Objetivo	Proporcionar Documentos Interesantes, Preservar la Calidad de los Documentos, Detectar Mala Información
Herramienta	Sugerencia, Petición Evaluación Sugerencia, Acción
Comunidad	—
Reglas	—
División del Trabajo	—

Tabla 16. Traducción a la Teoría de Actividad de los flujos de trabajo para el filtrado colaborativo especificados con INGENIAS.

El problema de la carencia de realimentación para los informadores puede ser descrito en términos de un *Conflicto Productor-Usuario*, anteriormente descrito en la sección 5.4.9. Un *Agente Personal* que juega el rol de *Informador* genera una *Sugerencia* para satisfacer su objetivo de *Proporcionar Documentos Interesantes*. Otros miembros de la comunidad que juegan el rol de *Evaluadores* analizan esta *Sugerencia*. Los *Evaluadores* hacen el análisis porque contribuye positivamente a su objetivo de *Preservar la Calidad de los Documentos*. Los resultados de este análisis,

i.e. la *Acción*, son procesados principalmente por el *Agente de la Comunidad*. Si los resultados no son comunicados a los *Informadores*, estos no pueden saber porque sus *Sugerencias* fueron rechazadas. Esta contradicción se muestra en la Fig. 78 y las correspondencias con el patrón en la Tabla 17. Como se aprecia en la Tabla 17 la identificación con el patrón de detección ha hecho uso de las equivalencias en UML-TA de la Tabla 5.

Conflicto Productor-Usuario	**Flujos de trabajo del filtrado colaborativo**
Objetivo Productor	Proporcionar Documentos Interesantes
Productor	Agente Personal + Informador
Actividad Productora	Propagar Sugerencias
Artefacto	Sugerencia
Actividad de Uso	Compartir Documentos + Petición Evaluación Sugerencia + Empezar la Evaluación del Documento + Evaluar Documentos = Actividad 1
Usuario	Agente Personal + Evaluador
Objetivo Usuario	Detectar Mala Información + Preservar la Calidad de los Documentos

Tabla 17. Correspondencias con el *Conflicto Productor-Usuario* para el filtrado colaborativo.

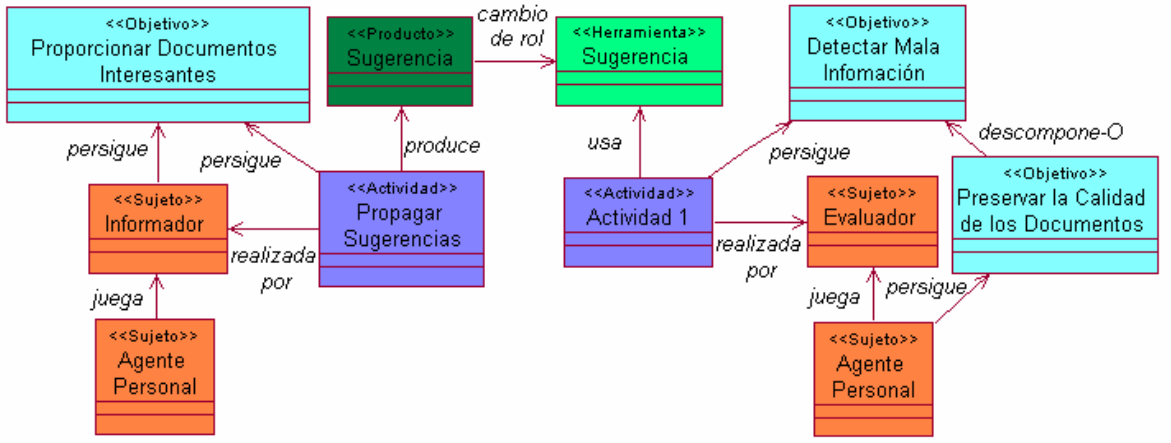


Fig. 78. *Conflicto Productor-Usuario* en la recomendación de información.

La Fig. 78 merece algunos comentarios referentes a las equivalencias entre conceptos de UML-TA. Nótese que agentes y roles de INGENIAS se traducen al mismo concepto *sujeto* de la TA y se relacionan con la asociación *juega*. Estas relaciones entre *sujetos* pueden ser pueden ser separadas o reunidas en un único *sujeto* en los diagramas para lograr una mejor comprensión de la situación, como es el caso para el *sujeto Agente Personal* y su rol *Evaluador*. Otra puntualización es que los objetivos de la *Actividad 1*, i.e. el *objetivo Detectar Mala Información*, y del *sujeto*

Agente Personal que juega el rol de *Evaluador*, i.e. el *objetivo Preservar la Calidad de los Documentos*, no son los mismos. El patrón es aplicable gracias a la relación *descompone-O* entre ambos objetivos. Esta relación implica que un modo de satisfacer el objetivo *Preservar la Calidad de los Documentos* es *Detectar Mala Información*. La *Actividad 1* es una *actividad* introducida en la detección de la contradicción. Esta *actividad* incluye los flujos de trabajo *Compartir Documentos* y *Evaluar Documentos* y la tarea *Empezar la Evaluación del Documento*.

El patrón de solución relacionado con el *Conflicto Productor-Usuario* (ver sección 5.4.9.2) sugiere que las *actividades* involucradas deberían también satisfacer *objetivos* para las *actividades vecinas*. Esto es, la *actividad* del productor *Informador*, i.e. *Propagar Sugerencias*, debería cumplir algún *objetivo* para el consumidor *Evaluador* y la *actividad* del consumidor, i.e. *Actividad 1*, satisfacer algún *objetivo* para el productor. La Fig. 79 muestra estos cambios en el modelo. *Propagar Sugerencias* persigue ahora un *objetivo* adicional *Detectar Usuarios Molestos* y la *Actividad 1* persigue el *objetivo* adicional *No Enviar Información No Deseada*. Esta solución permite el intercambio de información necesitado por los *Informadores* para ajustar su comportamiento al requerido en la comunidad.

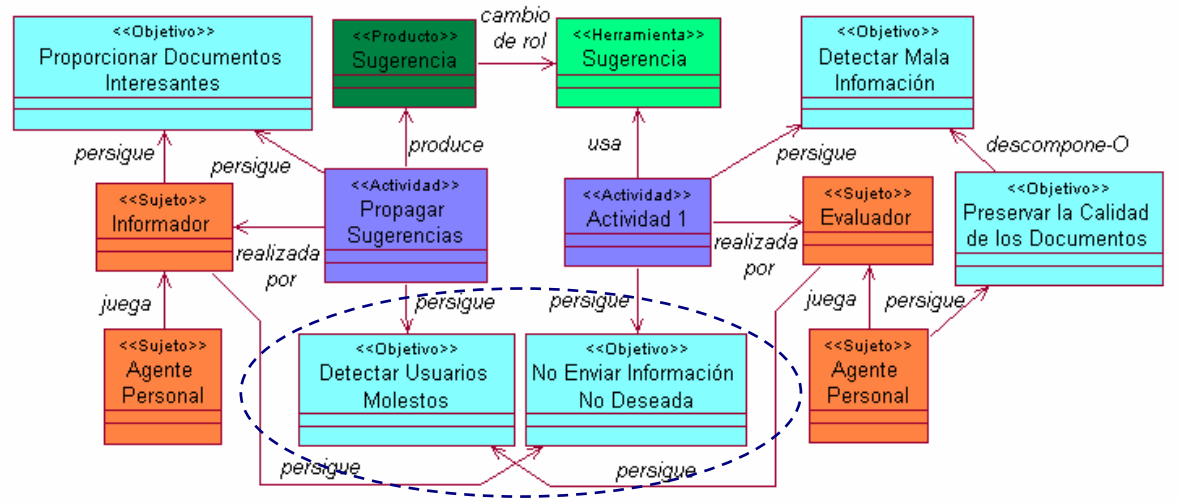


Fig. 79. Solución al *Conflicto Productor-Usuario* para recomendación de información.

El diagrama de la Fig. 79 no incluye elementos adicionales a los *objetivos* que podrían ser necesarios para implementar la solución, tales como nuevas *actividades*, *herramientas* u *objetos*. Por ejemplo, la *Actividad 1* puede crear algún tipo de informe acerca de las razones de una evaluación dada. No obstante, la no aparición de ciertos elementos no tiene que ser vista como una limitación de la propuesta. Los patrones de solución de las *propiedades sociales* dan ideas acerca de posibles mejoras en los modelos. No pretenden ser exhaustivos acerca de cómo realizar los cambios. Si los desarrolladores quieren incluir estas particularizaciones, pueden desarrollar nuevas descripciones UML para la contradicción que muestren los elementos involucrados en su visión particular de dicha contradicción.

5.6. Conclusiones

En este capítulo se han presentado las técnicas para trabajar con contradicciones sociales en las especificaciones de SMAs. Se ha partido del hecho de que las *contradicciones* siempre están presentes en los desarrollos de software. Siguiendo las ideas de la TA, estas contradicciones no son meros problemas que deben desaparecer, sino que pueden ser vistas como una fuerza impulsora de la evolución de los modelos. Cada nueva versión de la especificación surge como un intento de solución a las contradicciones existentes en las versiones previas.

Las principales aportaciones de este capítulo para el tratamiento de contradicciones acerca de las *propiedades sociales* en SMAs son:

- *Estructuras para representación*. Basadas en contemplar las contradicciones como *propiedades anti-patrón*. Esta descripción se basa en el lenguaje UML-TA introducido en la sección 3.5 y en la representación para las *propiedades sociales* de la sección 3.6. Esta representación de las contradicciones cubre los objetivos de permitir su uso por clientes y desarrolladores, servir como vehículo de discusión y poder ser usada como herramienta en el desarrollo.
- *Método de detección y solución*. La detección y corrección de contradicciones se ha realizado mediante el método genérico de comprobación de *propiedades sociales* ya introducido en la sección 3.7.
- *Método de definición*. Uno de los aspectos novedosos de este capítulo en relación con los previos ha sido el método de trabajo para obtener la definición de nuevas contradicciones. Esta información incluía la detección, explicación, evolución y solución de contradicciones genéricas. Para obtenerlas se han debido analizar los casos de estudio presentados en la literatura de TA sobre conflictos en sociedades humanas e intrapersonales. Estos casos han tenido que ser interpretados, abstraídos y adaptados para su uso orientado a la ISOA.
- *Repositorio de contradicciones*. Las contradicciones obtenidas de los estudios de la TA reflejan conocimiento experto sobre conflictos en sociedades humanas. Este conocimiento aplicado al desarrollo de SMAs trae la experiencia de décadas de investigación acerca de las *propiedades sociales* en sociedades humanas. En estas sociedades, dichas propiedades se muestran con una riqueza y complejidad muy superior a las que aparecen en el software actual.

La representación para *propiedades sociales* junto con las correspondencias entre vocabularios permiten usar las contradicciones para comprobar las especificaciones de los SMAs. El método de comprobación traduce las especificaciones con las correspondencias asociadas al vocabulario de una metodología de SMAs. Luego busca partes de los modelos traducidos que se correspondan con los patrones de detección de las contradicciones, es decir, que posean la misma estructura que estos patrones y valores compatibles en los campos de sus componentes. Si se encuentran, estas coincidencias son señaladas como posibles puntos contradictorios del modelo. La reorganización del patrón de detección y la adición de nuevos elementos permite

construir patrones de solución para la contradicción que apuntan a posibles mejoras en las especificaciones.

El caso de estudio acerca del tratamiento de contradicciones en la ISOA ha mostrado los principales beneficios del uso de esta propuesta:

- *Guía de contradicciones sociales a comprobar en SMAs*. Las librerías de contradicciones extraídas de la TA constituyen conocimiento experto listo para usarse en desarrollos de la ISOA. Las contradicciones propuestas trabajan directamente sobre la componente social e intencional de los SMAs. Puesto que éstas son el núcleo de la aportación original de muchas aproximaciones en el paradigma de agentes, y no están presentes en otros modelos de desarrollo de software, se ha proporcionado conocimiento para verificar nuevas propiedades de los SMAs.
- *Guía de resolución*. Las *propiedades anti-patrón* no sólo ofrecen la detección de situaciones conflictivas, sino que también describen posibles alternativas en la configuración de los modelos para eliminar las contradicciones.
- *Simplicidad del proceso de comprobación*. La representación con patrones estructurales UML-TA facilita la localización de contradicciones reduciéndola a un proceso de detección de patrones.
- *Flexibilidad*. El uso de patrones estructurales con variables en su descripción permite combinar patrones para describir situaciones complejas.
- *Generalidad del proceso*. En este capítulo se ha trabajado sólo con contradicciones extraídas de la TA. Sin embargo, no hay ninguna limitación a comprobar propiedades con distinto origen. Siempre que las propiedades sean traducidas al vocabulario de la TA, el método de comprobación sigue funcionando del mismo modo con otras clases de inconsistencias.
- *Guía en la evolución del desarrollo*. La identificación y resolución de contradicciones ofrece una indicación precisa de cómo se pueden dirigir las iteraciones del desarrollo. En la línea de la TA, las contradicciones en las especificaciones dirigen su evolución

En cuanto a las limitaciones del proceso propuesto son principalmente:

- *Interpretación de las contradicciones*. Las especificaciones de un SMA pueden presentar numerosas correspondencias con los patrones de detección de las contradicciones. Entre ellas aparecerán algunas que carezcan de significado en el problema que se trata, pero que consumirán recursos en su identificación y posterior descarte. Como ayuda para reducir el número de falsas identificaciones se ofrece la posibilidad de fijar parte de las variables de los patrones de detección. Además, las herramientas que implementen el método pueden disponer de filtros que permitan descartar parte de las identificaciones en función de los criterios establecidos. Ejemplos de estos filtros pueden encontrarse en el Capítulo 8 a propósito de la herramienta de ayuda para la ISOA con la TA.
- *Traducción de la forma textual a UML-TA*. Una vez más, la traducción a UML es una de las partes más difíciles del proceso de uso de las contradicciones: implica la interpretación y traslación de un análisis en lenguaje natural sobre una sociedad humana a UML-TA para SMAs.

Como conclusión, afirmar que el método propuesto no pretende reemplazar las guías de desarrollo de las metodologías ni los métodos existentes de verificación y validación en la ISOA. Se trata de una herramienta de ayuda al desarrollo para las metodologías de la ISOA. Tiene su propio nicho de utilidad en el tratamiento de las contradicciones sociales y es en ese aspecto es donde puede usarse como guía y mecanismo de comprobación.

Capítulo 6. Herramientas de desarrollo

Un aspecto muy importante de las técnicas de Ingeniería del Software es el de las herramientas de desarrollo con las que cuentan. La misión de estas herramientas es facilitar a los desarrolladores el seguimiento del proceso establecido por la técnica y liberarles de las tareas rutinarias a fin de que se centren en aquellas que requieren un mayor conocimiento. Este capítulo está dedicado a la presentación de la herramienta de ayuda al desarrollo en la ISOA basada en los resultados de esta tesis. Esta presentación comienza con una exposición de los criterios de diseño y los requisitos de la herramienta. A partir de estos objetivos se describe su arquitectura. Para dotar a la herramienta de la máxima flexibilidad, el conocimiento sobre lenguajes y propiedades concretas no está incluido en la implementación de la propia herramienta sino que se configura por medio de ficheros. La presentación finaliza con la descripción de la interacción del usuario con la herramienta. El capítulo concluye con una revisión de las características del sistema y las ventajas y limitaciones que su arquitectura reporta.

6.1. Introducción

En el estado actual de evolución de la Ingeniería del Software, se puede afirmar que una técnica que pretenda tener uso en desarrollos reales ha de contar con herramientas automatizadas que faciliten su aplicación. Estas herramientas cumplen varias misiones en el proceso de desarrollo:

- *Guía de aplicación*. Las herramientas cristalizan parte del conocimiento sobre el proceso de aplicación de la técnica. Es decir, la herramienta ayuda a los usuarios a conocer el orden de las tareas involucradas, cuáles son éstas y cómo realizarlas.
- *Verificación de los resultados*. Las herramientas permiten a los desarrolladores comprobar la correcta aplicación de las técnicas sobre los resultados.
- *Automatización de tareas repetitivas*. Parte de los procesos de las técnicas de Ingeniería del Software tienen un carácter repetitivo y requieren un conocimiento limitado. Estos procesos pueden ser automatizados liberando así a los usuarios para realizar aquellos procesos que sólo pueden ser hechos con la intervención humana.

Siguiendo esta línea de razonamiento, en el curso de la presente investigación se ha desarrollado una herramienta software para ayudar en la aplicación de las técnicas basadas en la TA al desarrollo de SMAs. Esta herramienta es el Asistente de Teoría de Actividad (ATA). El ATA ha sido implementado como un *plug-in* para las herramientas de modelado de SMAs. El objetivo era crear un asistente que asesore al

desarrollador con el conocimiento de la TA a medida que va construyendo las especificaciones.

El asistente se integra en los entornos de modelado existentes para la ISOA mediante una serie de APIs para la coordinación. Estas interfaces contemplan el acceso y modificación de las especificaciones en el entorno de modelado, la descripción de *propiedades sociales* en él o la notificación y recepción de eventos sobre el proceso de edición en el entorno.

En cuanto al tratamiento de las *propiedades sociales*, el ATA está concebido como una plataforma genérica de tratamiento de estas propiedades. El conocimiento relativo a los lenguajes de modelado, la traducción entre ellos y las *propiedades sociales* se proporciona de forma declarativa con XML. De este modo los usuarios del asistente pueden adaptarlo a nuevas metodologías o modificar las propiedades sin necesidad de alterar el código.

La arquitectura del asistente incluye un núcleo que refleja la infraestructura común para la TA en la ISOA (ver Capítulo 3). Partiendo de esta implementación básica se han contemplado las técnicas para la captura de requisitos con la GCR (ver 0) y la evolución de las especificaciones mediante el uso de contradicciones (ver Capítulo 5).

La infraestructura común para la TA en la ISOA comprende el lenguaje UML-TA, las correspondencias para su traducción a lenguajes basados en agentes, las estructuras para representar *propiedades sociales* y los métodos de traducción entre lenguajes y comprobación de propiedades. Estos elementos han sido implementados en paquetes genéricos basados en un lenguaje de metamodelado. Esta decisión está fundamentada en la necesidad de considerar las definiciones de lenguajes que no están especificados a partir de UML.

Sobre la base anterior, la implementación de la GCR y de las contradicciones supone principalmente la incorporación de librerías de *propiedades sociales*. No obstante han sido necesarios algunos elementos adicionales. Entre estos cabe citar las estructuras jerárquicas de manejo de *propiedades sociales* para reproducir la estructura de la GCR o los filtros para descartar las contradicciones que el usuario considere carentes de interés.

Por último, el ATA presenta una serie de módulos adicionales destinados a la evaluación de la propuesta de esta tesis. El ATA recoge estadísticas acerca del uso de las *propiedades sociales*, tales como el tamaño de los diagramas generados, la frecuencia de utilización de cada propiedad o la utilidad de los resultados para el desarrollo.

Como prueba de la viabilidad de la implementación del *plug-in*, la versión actual del asistente está integrada en la plataforma *INGENIAS Development Kit* (IDK) (*http://ingenias.sourceforge.net*). El IDK es un entorno para el desarrollo de SMAs que abarca el análisis, diseño e implementación de estos sistemas. La herramienta dispone de interfaces que permiten integrarle nuevos módulos con acceso a los recursos del IDK. Además se trata de una herramienta de código abierto y gratuita.

En las siguientes secciones de este capítulo se presentan los objetivos de diseño y los requisitos del ATA. A continuación se revisa la arquitectura implementada para satisfacer estas metas y los formatos de la información que hay que suministrarle para su funcionamiento. Después se describe cómo se lleva a cabo la interacción entre el asistente y el usuario según las distintas técnicas de la TA. Finalmente, el capítulo se

cierra con una discusión acerca de la herramienta y las ventajas y compromisos que implican las decisiones de diseño adoptadas.

6.2. Requisitos de la herramienta

El Asistente de la Teoría de Actividad (ATA) tiene como objetivo fundamental implementar las técnicas basadas en la TA para ayuda en el desarrollo de SMAs. Esta implementación incluye la infraestructura común para la TA (ver Capítulo 3), la GCR (ver 0) y las contradicciones (ver Capítulo 5). En definitiva, el asistente ha de contemplar:

- *Descripción de propiedades sociales*. Las técnicas de tratamiento de las *propiedades sociales* se fundamentan en las estructuras introducidas para su representación en la sección 3.6. Una parte esencial de esta representación son los patrones estructurales descritos con UML-TA para la detección y resolución de propiedades. Las especificaciones del SMA, una vez traducidas, también se representan como diagramas UML-TA.
- *Descripción de repositorios de propiedades sociales*. Las *propiedades sociales* son utilizadas en librerías para las diferentes técnicas. Para estas librerías resulta conveniente disponer de funciones tales como listados de propiedades o inclusión de nuevas propiedades. La forma concreta de realizar estos procesos dependerá de la técnica a la que se encuentren asociados. Por ejemplo, para la GCR resultan naturales los listados de *preguntas* por *áreas* y *aspectos*.
- *Traducción con correspondencias*. Las técnicas de esta tesis son un complemento para las metodologías de la ISOA que no reemplazan sus procesos. Las metodologías siguen trabajando con su propio lenguaje y actividades; sólo cuando se usan las técnicas de la TA es necesario contemplar otro lenguaje. Para que el proceso sea aplicable, la traducción de información entre la TA y los lenguajes basados en agentes ha de ser automática. En esta tesis la traducción se ha fundamentado en la descripción de correspondencias entre estructuras de lenguajes (ver sección 3.5).
- *Detección de patrones en especificaciones*. La comprobación de propiedades se basa en la identificación de sus patrones de detección sobre las especificaciones de SMAs (ver sección 3.7). Este proceso hace uso del conocimiento acerca de las primitivas de UML-TA con que se representan los patrones estructurales y las equivalencias en el lenguaje (ver sección 3.4).
- *Edición de patrones*. Los patrones no son estructuras estáticas. Se trata de marcos con ranuras modificables. La edición de los patrones puede requerir obtener información del resto de las especificaciones para determinar valores o bien realizar cambios en la representación gráfica de los patrones según los nuevos valores de las ranuras.
- *Mostrar patrones/propiedades a los usuarios*. La representación de las *propiedades sociales* incluye una parte textual y otra con la notación gráfica UML-TA. Por otra parte, las especificaciones de SMAs sobre las que se trabaja pueden también tener información textual o diagramas. La implementación ha de permitir la visualización de estas descripciones.

- *Obtener las especificaciones del SMA*. Los procesos basados en la TA trabajan con la información generada por el proceso de desarrollo del SMA. Se necesitan mecanismos para obtener esta información y hacerla accesible al asistente.
- *Modificar especificaciones*. La captura de nueva información con la GCR o la resolución de contradicciones requiere hacer cambios en los modelos del SMA. Estos cambios están limitados por el conocimiento disponible sobre el lenguaje y la herramienta de origen de las especificaciones. Es decir, la definición de las *propiedades sociales* permite saber que hay que cambiar elementos pero no todas las restricciones que el lenguaje o la herramienta de origen pueden aplicar a ese cambio.

Además de estos requisitos, relacionados directamente con la implementación de las técnicas de la TA, el asistente ha de ayudar a evaluar las aportaciones de esta propuesta. La evaluación viene dada por las estadísticas de uso de los diferentes componentes de la GCR y de las contradicciones de la TA. Estas estadísticas incluyen para la GCR:

- *Preguntas usadas en cada aspecto*. Aquellas *preguntas* que se han presentado alguna vez para capturar información.
- *Número de veces que se ha planteado cada pregunta*. Proporciona una estimación de cuáles son las *preguntas* que los usuarios han considerado más prometedoras a partir de su formulación y han intentado responder.
- *Número de veces que se ha resuelto cada pregunta*. Recoge cuántas veces se planteó cada *pregunta* y los usuarios pudieron rellenar los datos asociados.
- *Número de veces que la pregunta capturó información relevante*. Se trata de una estimación de las respuestas a *preguntas* que proporcionaron información considerada útil para el desarrollo. Incluye la información usada en construir el SMA y comprender su entorno.

Para las contradicciones de la TA, las estadísticas incluyen:

- *Contradicciones usadas*. Aquellas contradicciones cuya detección se ha planteado alguna vez.
- *Número de veces que se ha tratado de detectar la contradicción*. Indica cuáles son las contradicciones que los usuarios han considerado interesante buscar en sus especificaciones.
- *Número de detecciones de cada contradicción*. Recoge cuántas veces se intentó detectar cada contradicción y se halló en las especificaciones.
- *Número de veces que la detección fue relevante*. Las contradicciones pueden generar falsas identificaciones, es decir, la identificación de estructuras que se corresponden con el patrón de detección pero que no corresponden a contradicciones de interés en las especificaciones. Este valor indica el número de veces que se detectó cada contradicción y se trató de un problema real en las especificaciones.
- *Número de veces que se usó la solución*. Recoge cuántas veces se identificó una contradicción relevante y se usó el patrón de solución asociado para resolverla.

Finalmente, y en común para la GCR y las contradicciones, también resulta de interés conocer el tamaño de las especificaciones con las que se ha trabajado en cuanto a diagramas y elementos. De esta forma se puede obtener una idea aproximada de la complejidad del problema abordado con estas técnicas.

Disponer de estas estadísticas añade los siguientes requisitos al ATA:

- *Detección de eventos en la interfaz*. La mayor parte de las estadísticas citadas sólo requieren determinar cuando el usuario solicita acciones o realiza elecciones. Éste es el caso de saber cuántas veces se ha lanzado el proceso de detección de contradicciones o las *preguntas* que se ha intentado resolver. No obstante, en algunos casos, como determinar cuando una *pregunta* de la GCR capturó información relevante, puede ser necesario interrogar explícitamente al usuario del ATA.
- *Guardar estadísticas*. Las estadísticas anteriores han de ser recogidas a lo largo de un desarrollo completo. Las estadísticas de sesiones individuales de trabajo con la herramienta no son significativas porque están sesgadas por el momento del desarrollo.
- *Introspección de propiedades sociales*. Parte de las estadísticas involucran conocer el tamaño de las especificaciones obtenidas o el número de *propiedades sociales* en un repositorio.

Con esta sección se plantea la funcionalidad que ha de cubrir la implementación del ATA. Existen también consideraciones a tener en cuenta acerca de cómo debe ser dicha implementación. Estos principios de diseño se recogen en la siguiente sección.

6.3. Principios de diseño

El diseño del ATA está presidido por el principio fundamental de facilitar la extensibilidad de la herramienta con nuevo conocimiento y funcionalidades. Esta extensibilidad es necesaria fundamentalmente en cinco aspectos:

- *Cambios en el lenguaje*. La actual formulación de UML-TA viene dada por el núcleo conceptual común de la investigación en TA (ver Fig. 4) y la experiencia del uso de la TA en desarrollos de SMAs. Por tanto, es previsible que con la evolución de la TA y su aplicación a la ISOA puedan cambiar estos conceptos. La implementación del ATA ha de facilitar la incorporación de estos cambios en el lenguaje.
- *Incorporación de propiedades sociales*. En este trabajo se han presentado dos librerías de *propiedades sociales*: la GCR para la captura de requisitos y las contradicciones de la TA. Estos repositorios pueden ser alterados según la nueva experiencia en desarrollos. Por otra parte, es posible que surjan también nuevos repositorios, por ejemplo para dominios de trabajo concretos. Se ha de plantear un mecanismo para modificar el contenido de las librerías que además no debería confiar en la programación para cambiar las *propiedades sociales*. Evitando que la modificación requiera programación se fomenta la colaboración de los clientes de los proyectos en crear nuevo conocimiento.

- *Tratamiento de nuevas metodologías*. Las técnicas de la TA utilizan correspondencias para trabajar sobre las especificaciones de SMAs. Esta flexibilidad permite su aplicación sobre nuevas metodologías.
- *Conexión con nuevas herramientas de modelado*. La utilización efectiva de las técnicas de la TA sobre metodologías de la ISOA requiere aprovechar las capacidades de automatización que se ofrezcan. La conexión con herramientas de modelado trabaja en este sentido para minimizar la posible intervención del usuario. Por ejemplo, puede evitar tener que guardar las especificaciones en formatos específicos o interrumpir el flujo normal de trabajo sobre las especificaciones.
- *Incorporación de nueva funcionalidad sobre las propiedades sociales*. Las funciones esenciales sobre propiedades requeridas por el ATA han sido presentadas en la sección anterior. El diseño debería contemplar sin embargo la posibilidad de nuevas funcionalidades. Por ejemplo, facilitar al usuario el examen de las contradicciones halladas en unas especificaciones puede requerir la implementación de filtros para descartar algunas automáticamente.

Además de los requisitos y de los principios de diseño, es importante considerar la clase de interacciones que deben tener lugar entre el usuario y el sistema. Este aspecto es considerado en la siguiente sección.

6.4. Interacción con el usuario

Todos los procesos presentados en los capítulos anteriores están concebidos para un trabajo interactivo con el usuario. Se trata de métodos que pueden ser automatizados en un alto grado, pero que requieren la participación del usuario para seleccionar opciones y juzgar la validez de las elecciones hechas automáticamente. Esta intervención del usuario hace conveniente prestar atención a las interacciones entre el usuario y el asistente antes de detallar la arquitectura del ATA.

En concreto, la interacción con el usuario es necesaria para:

- Decidir qué *preguntas* de la GCR atañen al problema que se está tratando.
- Decidir qué contradicciones se han de comprobar sobre las especificaciones.
- Determinar los valores de las variables en los patrones de detección de las *propiedades sociales*, tanto para las *preguntas* como para las contradicciones.
- Fijar los filtros aplicables a las identificaciones de propiedades en las especificaciones.
- Establecer cuándo una contradicción identificada en los modelos representa un problema relevante en el dominio real.
- En las contradicciones, decidir la mejor manera de modificar los modelos y fijar las variables del patrón de solución.
- En general, fijar las métricas sobre el funcionamiento del asistente que se usan para evaluar la propuesta de esta tesis.

A continuación se detallan las interacciones entre el usuario y el ATA. En estas interacciones se muestran los puntos donde se han de solicitar los datos anteriores al

usuario. Sólo las métricas relativas al funcionamiento del sistema, que son sobre todo relevantes para los investigadores, quedan fuera de esta presentación.

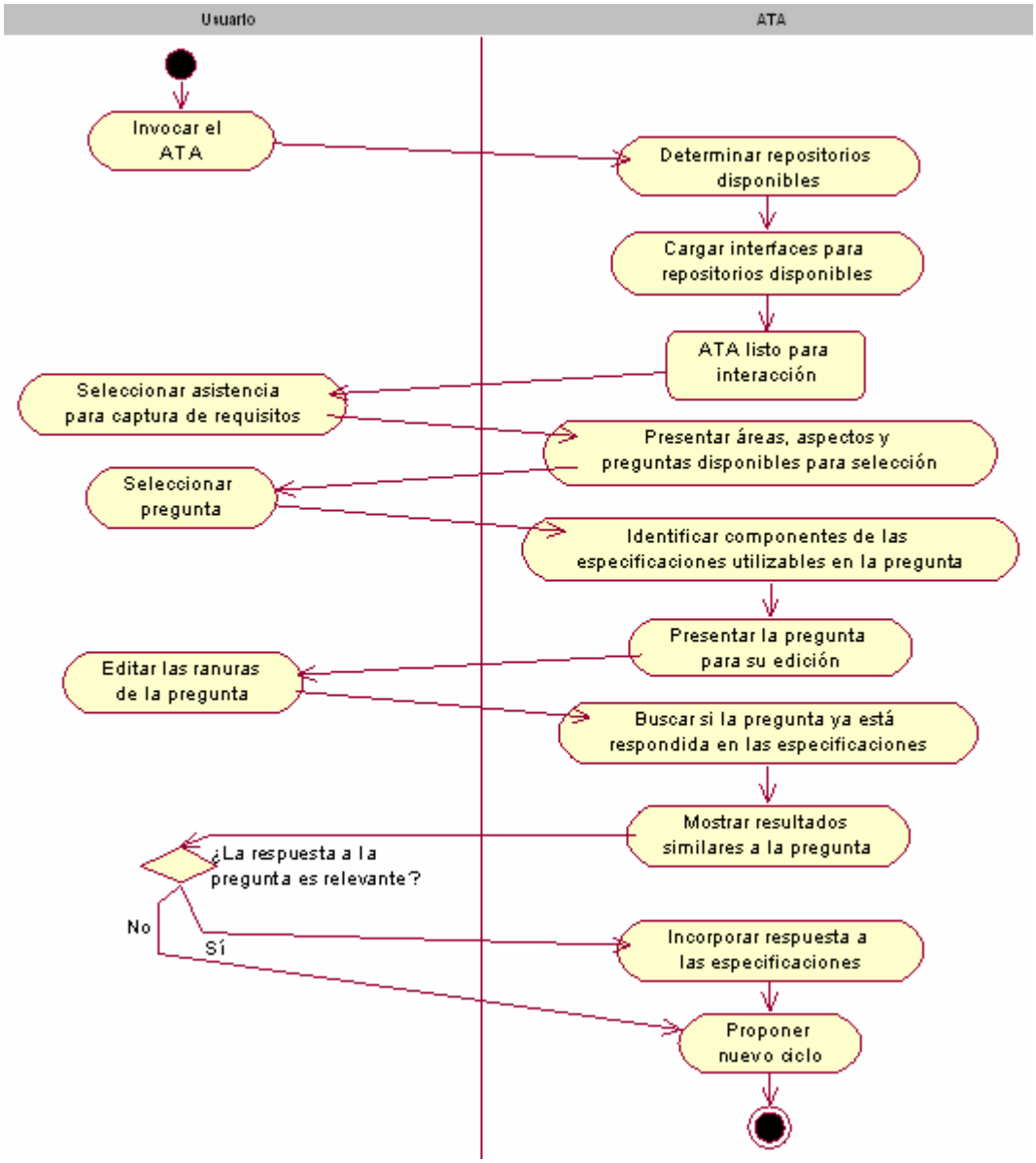


Fig. 80. Diagrama de Actividad UML: Interacción entre el usuario y el ATA con la GCR.

La captura de información sobre los requisitos se realiza con la GCR según el proceso mostrado en la Fig. 80. Cuando el usuario selecciona esta opción se le presenta una estructura jerárquica con *áreas*, *aspectos* y *preguntas* que reproduce la

relación de estos elementos en la GCR. A continuación el usuario selecciona una *pregunta* y el ATA le muestra su patrón estructural. Para los valores modificables del patrón, el asistente sugiere posibles candidatos ya presentes en las especificaciones. El usuario edita a continuación la *pregunta*, fijando algunos o todos los valores de las ranuras variables. Una vez hecho esto, el ATA comprueba las especificaciones, buscando porciones que contengan la misma o similar información que la *pregunta* respondida. Las informaciones similares se le presentan al usuario para que decida si quiere incorporar la respuesta a la *pregunta* a las especificaciones. Finalmente, si el usuario así lo decide, se actualizan las especificaciones con la respuesta.

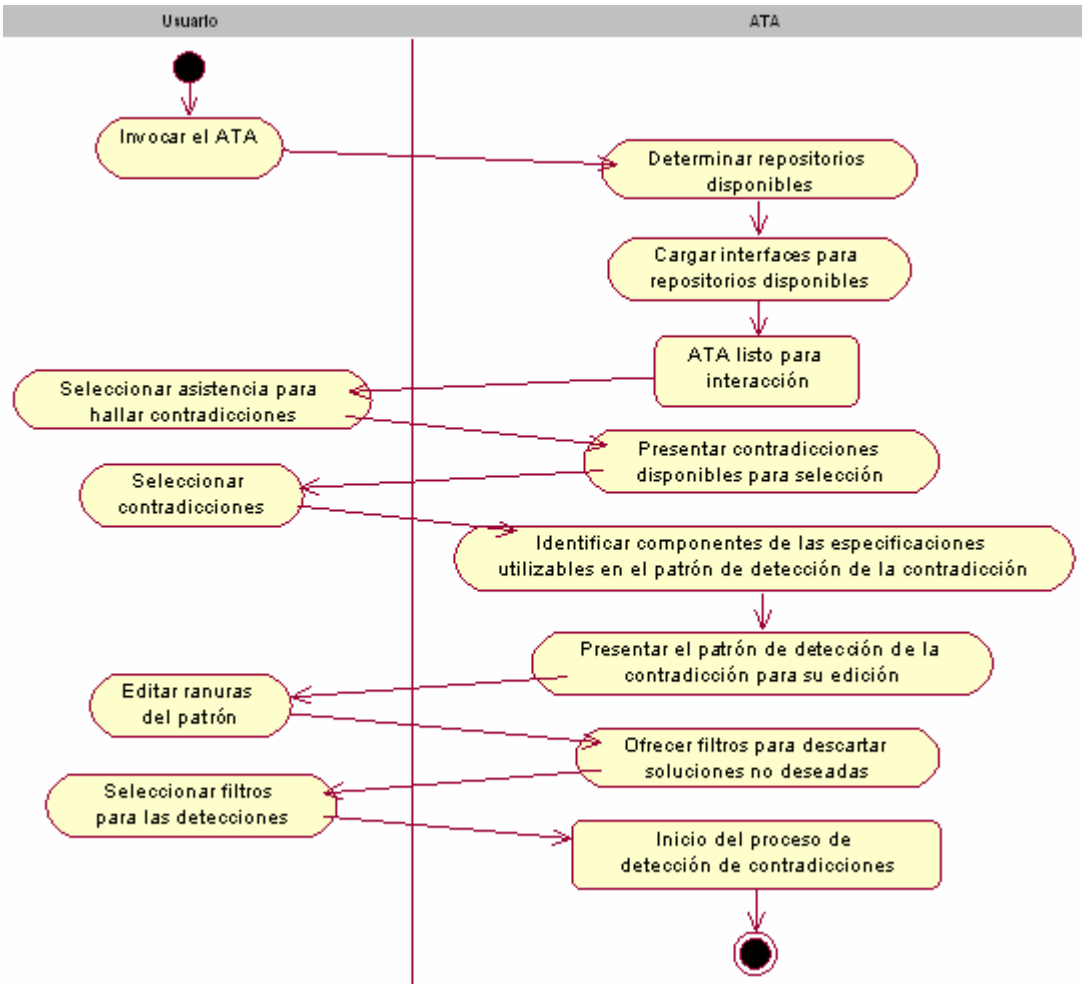


Fig. 81. Diagrama de Actividad UML: Selección de parámetros para la comprobación de contradicciones en el ATA.

Otro modo posible de trabajo con el ATA es la comprobación de contradicciones. El proceso comienza estableciendo los parámetros de la comprobación, tal y como se

ve en la Fig. 81. Esta primera parte guarda bastantes similitudes con la edición de las preguntas de la GCR en la Fig. 80. Al usuario se le presentan en primer lugar una lista de contradicciones disponibles en el sistema para que seleccione aquellas que desea comprobar. A continuación edita los patrones de detección de las contradicciones seleccionadas, fijando los valores de las ranuras variables que desee. Cuando concluye la edición, el usuario elige los filtros a aplicar en las detecciones. Estos filtros determinan características de los resultados que se desea examinar. Con estos parámetros se inicia el proceso de detección y solución de la Fig. 82.

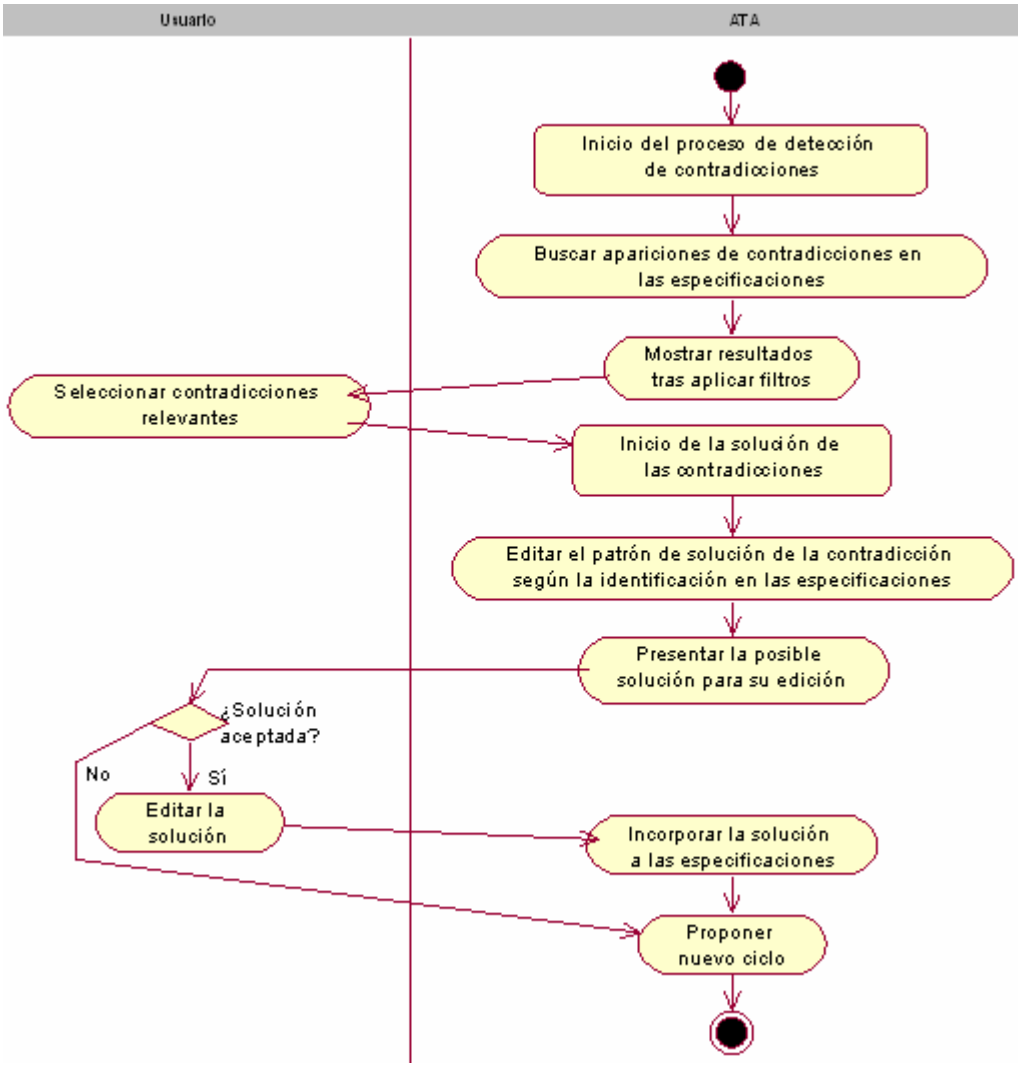


Fig. 82. Diagrama de Actividad UML: Interacción en el ATA para resolver contradicciones.

La Fig. 82 muestra la continuación del proceso de tratamiento de las contradicciones iniciado en la Fig. 81. El ATA busca las contradicciones seleccionadas por el usuario con los valores que éste ha fijado. Después descarta las identificaciones que no satisfacen los filtros y presenta los resultados restantes. Entre la lista de resultados el usuario selecciona las contradicciones relevantes, es decir, aquellas que reflejan un problema en el dominio real del SMA en desarrollo. Para cada contradicción relevante, se presentan sus patrones de solución. Estos patrones de solución se instancian sustituyendo las variables de sus ranuras según los valores fijados en la identificación del patrón de detección de dicha contradicción. La solución se le presenta al usuario que puede incorporarla a las especificaciones, quizás con modificaciones adicionales, o bien rechazarla.

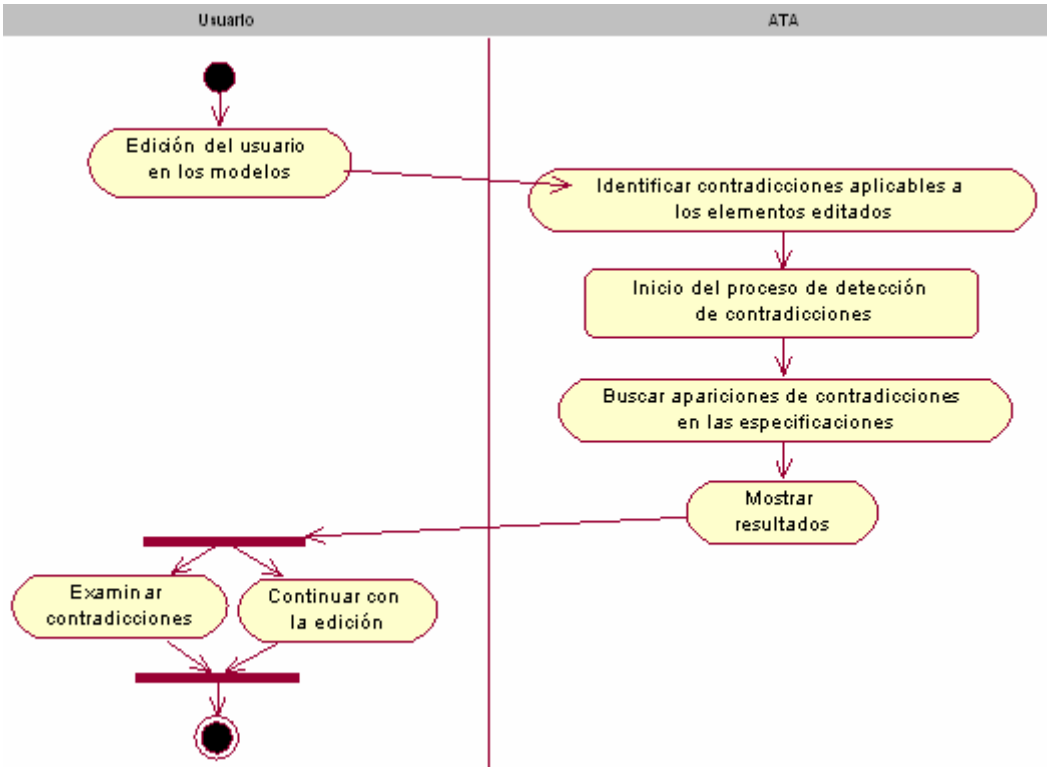


Fig. 83. Diagrama de Actividad UML: Interacción entre el usuario y el ATA en la monitorización de la edición.

La última interacción que se considera entre el usuario y el ATA corresponde a la situación en la que el asistente monitoriza la edición de las especificaciones en un segundo plano. Esta interacción se muestra en la Fig. 83. El usuario está trabajando con la herramienta de modelado de SMAs propia de su metodología. En un momento dado se genera un evento en la herramienta que es percibido por el ATA. Cuándo se producen estos eventos y a qué responden depende de la interfaz que se construya entre ambos sistemas. El evento de la herramienta origina un ciclo de detección de

contradicciones en el ATA. Si el evento incluye información sobre cuáles han sido los elementos editados se pueden determinar antes de la comprobación las contradicciones aplicables. Por ejemplo, si se ha añadido un nuevo agente en las especificaciones se tratará de contradicciones en las que aparezcan los *roles* de la TA *sujeto* o *comunidad*. Si se desconoce cuáles son los elementos editados se comprobarán todas las contradicciones. Una vez concluido el proceso de identificación, el ATA muestra al usuario las contradicciones halladas de forma no intrusiva. Ello quiere decir que lo hace sin alterar el flujo normal de trabajo del usuario en la edición de los modelos. Por tanto, e usuario ha de tomar la decisión de examinar la información que el ATA le ha suministrado.

Una vez vistos los requisitos de la sección 6.2, los principios de diseño de la 6.3 y las interacciones con el usuario de esta sección, se pasa a examinar la arquitectura del ATA.

6.5. Arquitectura

El ATA se implementa como un *intérprete* (ver el patrón de diseño en [Gamma *et al.* 1995]) de un lenguaje genérico de descripción grafos. El lenguaje de grafos describe los patrones de las *propiedades sociales* y las especificaciones. Las operaciones de traducción y detección se convierten en interpretaciones operacionales de las primitivas de dicho lenguaje. Es decir, en función de las primitivas involucradas en la estructura se adoptan las decisiones relativas a los procesos.

La arquitectura basada en el *intérprete* aprovecha que el lenguaje de la "gramática" de grafos es simple (como se verá en la siguiente sección) y permite ampliar fácilmente los "modos de interpretar" (i.e. las operaciones) sobre dicha "gramática".

En función del patrón de diseño dominante en la implementación del ATA se ha optado por dividir esta presentación en dos partes:

1. *Gramática de grafos*. Describe un lenguaje genérico de descripción de grafos. La decisión de adoptar este lenguaje se fundamenta en la necesidad de representar lenguajes que no están basados en UML y supone adoptar la hipótesis de que la mayor parte de las especificaciones de SMAs pueden representarse como grafos. La presentación de la gramática incluye la del paquete que la implementa y que proporciona la base para el resto de la implementación.
2. *El asistente como intérprete*. Las funcionalidades del asistente están implementadas como intérpretes del lenguaje de grafos. Aquí se tratan los paquetes que realizan la traducción, detección de propiedades, obtención de información y presentación de la misma.

El patrón *intérprete* no es el único usado en la arquitectura del ATA. Los patrones de diseño de [Gamma *et al.* 1995] han sido utilizados en distintos componentes del sistema. Para explicar la arquitectura en términos de estos patrones los componentes del ATA se etiquetan con estereotipos con el mismo nombre que los componentes de los patrones en [Gamma *et al.* 1995].

6.5.1. Gramática de grafos

El ATA maneja fundamentalmente tres tipos de información:

- *Patrones estructurales de propiedades sociales*. La representación de las *propiedades sociales* incluye patrones de detección y solución que se representan en UML-TA.
- *Especificaciones de SMAs en UML-TA*. Las especificaciones traducidas desde las metodologías de la ISOA o la información capturada con la GCR se representa también con UML-TA.
- *Especificaciones de SMAs en lenguajes basados en agentes*. Se trata de las especificaciones de SMA no traducidas aún para su procesamiento en el ATA.

La diversidad de lenguajes empleada obliga a considerar un metalenguaje capaz de representar los tres tipos de información. La elección recae sobre el lenguaje de metamodelado GOPRR. GOPRR (*Graph*, *Object*, *Property*, *Relationship*, *and Role*) [Lyytinen & Rossi 1999] define el vocabulario permitido en un grafo. La Fig. 84 muestra un resumen de sus primitivas de modelado. Las especificaciones se conciben como grafos (entidad *Graph*), en términos de asociaciones (*Relationships*) cuyos extremos se definen con roles (Role) y que conectan entidades (*Objects*). Todos estos elementos contienen propiedades (*Properties*) que pueden ser otros objetos, grafos, roles o relaciones. Existen más elementos en GOPRR como las restricciones en gráficos (e.g. número de elementos, presencia o ausencia de elementos) o la cardinalidad de las relaciones.

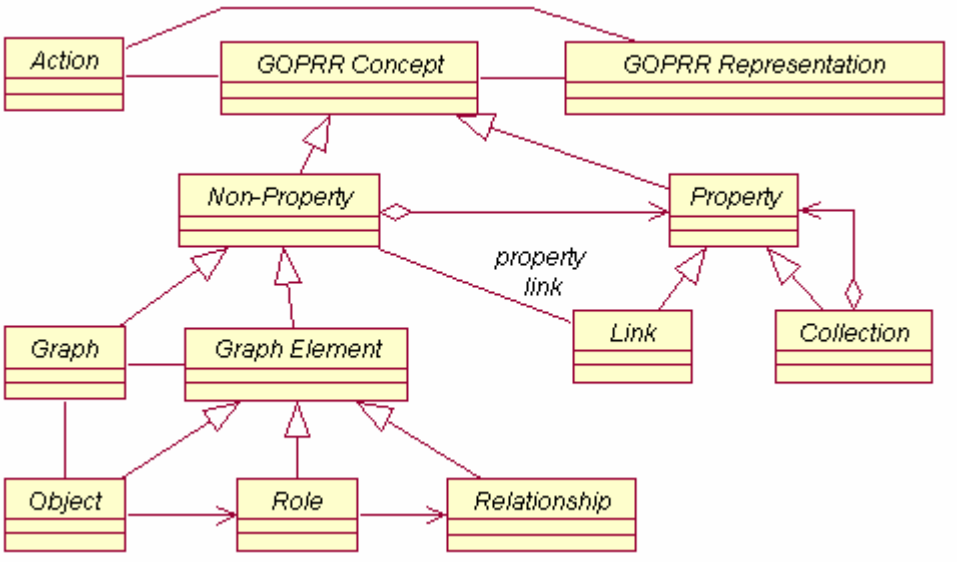


Fig. 84. Primitivas de GOPRR.

La decisión de usar GOPRR parte de la hipótesis de que al menos una parte importante de los lenguajes usados para representar los tres tipos de informaciones

anteriores puede ser vista como grafos. Así, UML-TA está especificado con UML. La especificación estándar de UML [OMG 2003] está realizada con el lenguaje de metamodelado MOF [OMG 2002]. GOPRR tiene un poder expresivo equivalente al de MOF, lo que le permite especificar UML. La elección de GOPRR sobre MOF se debe a que GOPRR dispone en la actualidad de herramientas de metamodelado (ver METAEDIT+ [Lyytinen & Rossi 1999]) mientras que MOF no. En cuanto a los lenguajes para la especificación de SMAs, la especificación con grafos de GOPRR permite representar las modelos basados en diagramas y buena parte de los formalismos. Hay que tener en cuenta que un predicado lógico puede ser visto como una relación n-aria entre sus argumentos y por tanto representarse con GOPRR. No obstante, ésta es sólo una hipótesis de implementación que se aplica a muchos lenguajes pero cuya comprobación queda pendiente como trabajo futuro.

La implementación del metalenguaje GOPRR está incluida en el paquete *graph* del ATA. La estructura de *graph* puede verse en la Fig. 85.

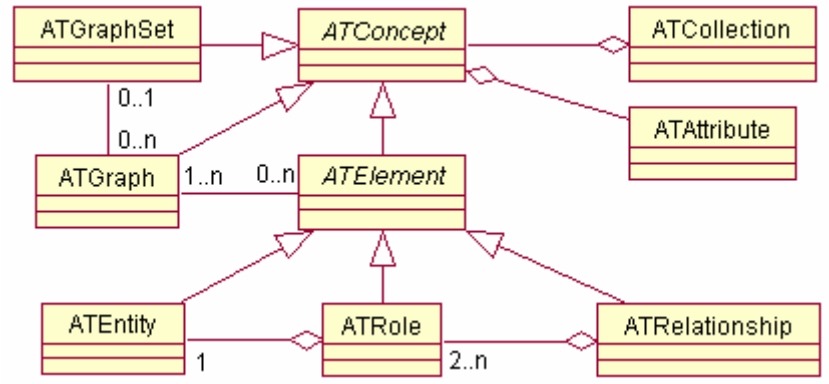


Fig. 85. Implementación de GOPRR en el paquete *graph* del ATA.

Las clases de *graph* reproducen prácticamente las primitivas de GOPRR aunque existen algunas excepciones. Además de las primitivas omitidas, la principal es la incorporación de la clase *ATGraphSet* para representar unas especificaciones como un conjunto de diagramas, i.e. *ATGraphs*. Para una mayor claridad, la Fig. 85 sólo muestra las relaciones entre clases de *graph* inspiradas por GOPRR.

En el diseño de *graph* se han empleado varios patrones de diseño orientado a objetos [Gamma *et al.* 1995] tendentes a potenciar la flexibilidad en el manejo de los grafos necesaria para satisfacer los principios de diseño del ATA. A continuación se detallan estos patrones.

En primer lugar se han considerado las clases *ATGraph* y *ATGraphSet* como *fachadas* ("*facades*"). El trabajo con los grafos requiere obtener informaciones de los diversos elementos involucrados y combinarlas. Hacer a las clases clientes del paquete *graph* responsables de estas tareas implica sobrecargarlas con mucha información relativa a los grafos. Para evitar este acoplamiento, el acceso a la funcionalidad a nivel de un grafo se confía a la clase *ATGraph* y a nivel de las especificaciones completas a *ATGraphSet*. Este papel como fachada del *ATGraph* es el representado en la Fig. 86.

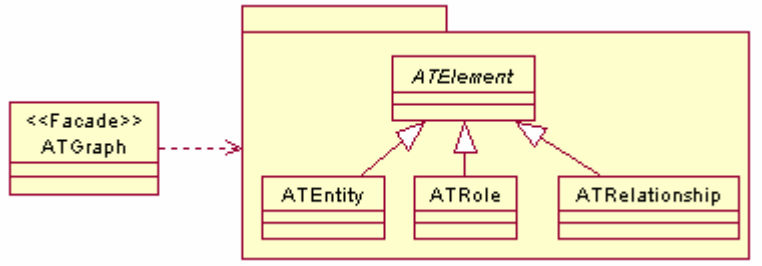


Fig. 86. Patrón de diseño *fachada* con el *ATGraph* en el paquete *graph*.

El otro patrón importante es la *cadena de responsabilidad* ("*chain of responsibility*"). Este patrón se emplea cuando una petición puede ser manejada por objetos de distintas clases y éstas no pueden ser conocidas *a priori*. En el caso de *graph* existen peticiones que pueden hacerse a cualquier *ATElement*, tales como saber si un *ATElement* está conectado con otro, si posee cierto atributo o si aparece en un cierto *ATGraph*. La forma en la que se manejan estas peticiones cambia según la subclase de *ATElement*: en unos casos la petición es atendida por el objeto que la recibe pero en otros ha de ser redirigida a otra subclase. En lugar de diferenciar según las subclases, el patrón *cadena de responsabilidad* ofrece una forma más flexible de manejar la interacción. Los objetos de *ATElement* reciben la petición y la responden o la pasan a su *sucesor*. Esta forma de proceder queda reflejada en el patrón de la Fig. 87. Al igual que ocurría con el patrón *fachada*, el patrón *cadena de responsabilidad* aparece para algunas operaciones con la jerarquía basada en la clase *ATConcept* en lugar de en *ATElement*. La jerarquía con *ATConcept* incluye también a las clases *ATGraph* y *ATGraphSet*.

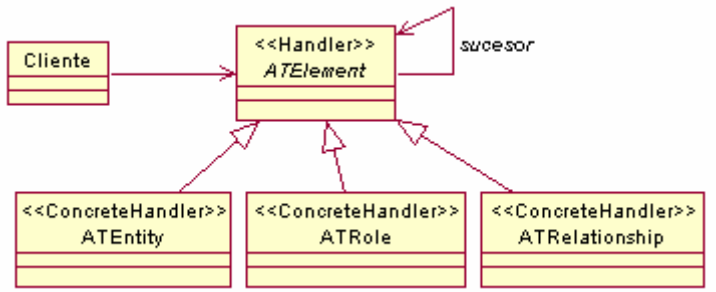


Fig. 87. Patrón de diseño *cadena de responsabilidad* para *ATElement* en el paquete *graph*.

Con la descripción de estos patrones de diseño concluye la presentación del soporte para la representación de grafos. El paquete *graph* es el lenguaje sobre el que se representan en el ATA los diagramas UML-TA y las especificaciones de SMAs. La siguiente sección muestra como se construye la funcionalidad vista en los requisitos sobre esta base.

6.5.2. El asistente como intérprete

La funcionalidad del ATA se implementa con *intérpretes* [Gamma *et al.* 1995] del lenguaje de grafos del apartado anterior. A lo largo de esta sección se introduce primero la estructura general que realiza esta funcionalidad, después se detallan algunos aspectos importantes de la estructura con los patrones de diseño empleados y finalmente se describen a alto nivel las secuencias de acciones que llevan a cabo las interacciones con el usuario (vistas en la sección 6.4).

6.5.2.1. Estructura general

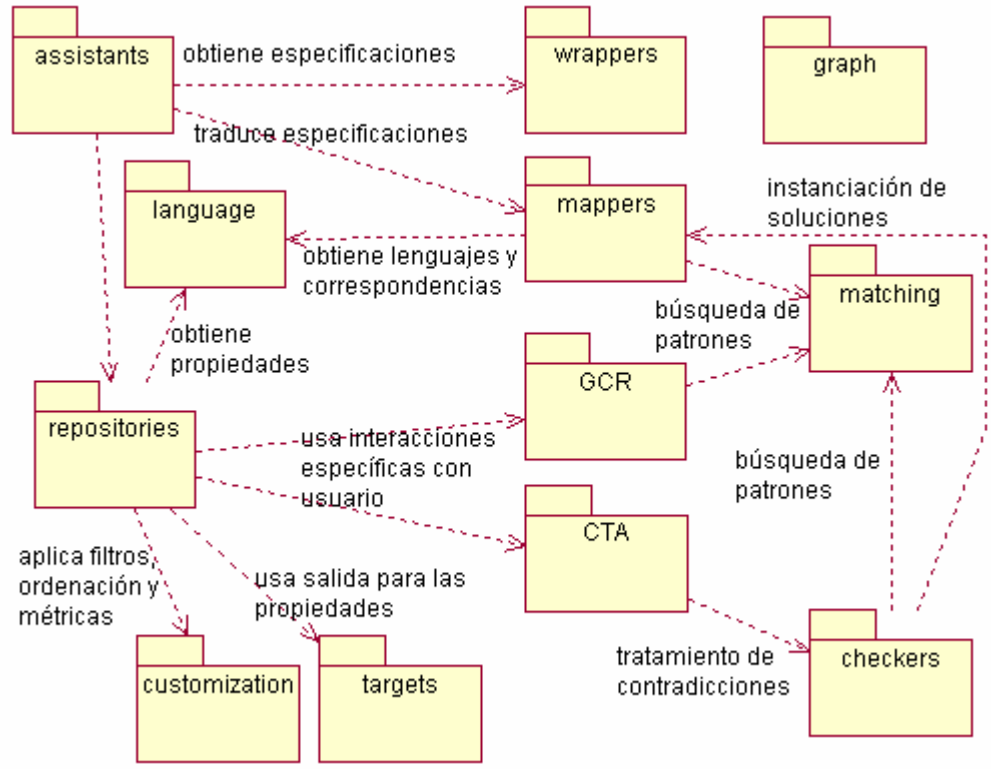


Fig. 88. Paquetes del ATA.

El ATA se estructura mediante los módulos que pueden verse en la Fig. 88. Estos paquetes se relacionan con: los lenguajes involucrados en los procesos con la TA, es decir, el lenguaje UML-TA usado para describir los patrones y los lenguajes de las metodologías de SMAs usados en las especificaciones (i.e. *language*); el proceso de traducción (i.e. *mappers*); los repositorios de conocimiento acerca de las *propiedades sociales* (i.e. *repositories*, GCR y CTA); las fuentes de las especificaciones del sistema (i.e. *wrappers*); la detección de patrones (i.e. *matching*); la comprobación de contradicciones (i.e. *checkers*); las métricas y filtros (i.e. *customization*); y los recursos de entrada/salida para los procesos de la TA (i.e. *targets*). El sistema es

controlado globalmente por los asistentes en el paquete *assistants*. El paquete *graph* es usado por el resto del sistema como ya se indicó en el apartado anterior.

A continuación se explican los paquetes de la Fig. 88 junto con las interfaces definidas por ellos. Aquí sólo describe su funcionalidad. Las relaciones de utilización entre los paquetes se muestran después. Los paquetes del ATA son:

- *wrappers*. Este paquete define las interfaces que proporcionan un recubrimiento uniforme de las fuentes que suministran las especificaciones. Estas fuentes pueden corresponder tanto a un componente estático, e.g un fichero, como dinámico, e.g. una herramienta de modelado. *wrappers* define las interfaces *Wrapper* y *WrappersManager*. Cada *Wrapper* recubre un tipo de fuente y es capaz de proporcionar entrada/salida sobre ella para obtener y modificar una especificación. El *Wrapper* respeta el lenguaje original de las especificaciones y sólo transforma la representación a otra adecuada para las interfaces internas del asistente. El *WrappersManager* gestiona varios *Wrappers* de manera que puede componer una especificación a partir de las especificaciones parciales que le suministren varios *Wrappers*.
- *language*. Proporciona interfaces para acceder a la definición de los vocabularios, (i.e. *Vocabulary*), las correspondencias entre vocabularios (i.e. *Correspondences*) y las *propiedades sociales* (i.e. *Properties*). Los vocabularios se emplean a efectos de validación de las definiciones de grafos en un lenguaje dado; indican cuáles son las primitivas válidas de modelado en el lenguaje, i.e. las entidades, las relaciones, los roles y sus propiedades. Las correspondencias expresan de una forma declarativa cómo traducir descripciones de un vocabulario a otro. Se basan en grafos con elementos marcados. En las especificaciones a traducir se buscan porciones que encajen con el patrón fuente. Cuando se encuentran, la traducción es el patrón destino donde las variables son sustituidas según la porción de especificación que se ha identificado con el patrón fuente. Las descripciones de las *propiedades sociales* son las estructuras vistas en la Fig. 23. También se expresan mediante grafos con elementos marcados.
- *matching*. Es el paquete que maneja la identificación de los patrones de detección de las *propiedades sociales*. Define las interfaces *PatternFinder* y *PatternMatch*. La interfaz *PatternFinder* es implementada por entidades capaces de examinar unas especificaciones para buscar el patrón estructural que se les pasa como parámetro. Su salida es un conjunto de *PatternMatchs* que representa las posibles instanciaciones del patrón estructural buscado con la información de los modelos.
- *mappers*. El paquete *mappers* gestiona la traducción entre lenguajes. Este paquete sigue el proceso de traducción por niveles de la sección 3.5.3. *mappers* define una interfaz *Mapper*, la cual se instancia con vocabularios y correspondencias entre ellos. Un *Mapper* es capaz de traducir las especificaciones que se le suministran según sus propias correspondencias. Generalmente se trata de una correspondencia traducible en un solo paso. La interfaz *Mappings* representa al gestor de varios *Mappers*. Las clases que implementan la interfaz *Mappings* son las encargadas de gestionar la traducción por niveles de los diferentes *Mappers*. Aplican los *Mappers* que tienen asociados en secuencia para lograr la traducción completa de unas especificaciones.
- *repositories*. Es el paquete donde se describe la funcionalidad genérica de los repositorios de *propiedades sociales*. Define la interfaz *Repository* que ha de

implementar una de estas librerías. Esta interfaz incluye métodos para dar la lista de propiedades disponibles, obtener su representación, aplicar una técnica de la TA y devolver los resultados de dicha aplicación. Las clases que implementan *Repository* obtienen sus propiedades usando el paquete *language*.

- *GCR*. Este paquete contiene la especificación de la Guía de Captura de Requisitos para SMAs. Describe las *áreas*, *aspectos* y *preguntas* de la misma, sus asociaciones y da la representación de las *preguntas* como estructuras de *propiedades sociales*.
- *CTA*. Es el paquete que describe las contradicciones de la TA mediante estructuras de *propiedades sociales*.
- *checkers*. Es el paquete que maneja el trabajo con las contradicciones. Define las interfaces *Checker* y *CheckersManager*. El *Checker* implementa un marco genérico que se instancia con la descripción de una contradicción concreta. Una vez instanciado es capaz de buscar el patrón de detección en unos modelos y plantear sus patrones de solución según las identificaciones. Como se verá más tarde, estas tareas se realizan reutilizando la funcionalidad de los paquetes *matching* y *mappers*. El *CheckersManager* gestiona varios *Checkers*. Dada una especificación, aplica todos sus *Checkers* y da una lista de posibles identificaciones en la especificación.
- *customization*. Se trata de los filtros, criterios de ordenación y métricas disponibles en el ATA. Tienen la doble intención de permitir un trabajo flexible por parte del usuario y proporcionar los datos para evaluar la aproximación de esta tesis. Entre los filtros para patrones están los que consideran el tamaño del diagrama en elementos, la inclusión de elementos de cierto tipo (e.g. presencia de *herramientas* o *comunidades*) o que sólo incluyan ciertos elementos de las especificaciones. Los criterios de ordenación son nombre de la *propiedad social* o aparición de un *rol* de TA. Por último, las métricas hacen referencia a la medición del uso de los elementos de los repositorios.
- *targets*. Este paquete recubre los posibles recursos para mostrar entrada/salida relacionada con las *propiedades sociales*, tales como la definición de las propias *propiedades sociales*, correspondencias con las especificaciones o propuestas de modificación de los modelos. Cada *Target* recubre un recurso y es capaz de mostrar en él una especificación de modelos de TA, así como de recoger modificaciones. El *TargetsManager* gestiona varios *Targets* de manera que puede utilizar varios destinos.
- *assistants*. Es el paquete con la funcionalidad común de los asistentes. Define la interfaz *Assistant* que deben implementar los asistentes. Estos organizan el uso de los paquetes anteriores para aplicar las técnicas de la TA. En el estado actual de la implementación sólo está definido el asistente *ATAssistant* incluido en el ATA.

6.5.2.2. Detalles arquitectónicos

La implementación de la estructura anterior (ver Fig. 88) se ha hecho siguiendo patrones de diseño [Gamma *et al.* 1995] para solventar algunos de los problemas presentados en su realización.

El primer problema es el relativo a la creación de los objetos complejos tales como las definiciones de los vocabularios, las correspondencias y las estructuras de las

propiedades sociales. Se trata de componentes que han de ser construidos en varios pasos y ensamblados siguiendo una cierta estructura. Para aislar este proceso de creación se ha usado el patrón *constructor* ("*builder*"). En la Fig. 89 se muestra la construcción de las correspondencias dirigida por la clase *ATAssistant* (en general por las que implementan la interfaz *Assistant*) mediante la interfaz *CorrespondencesFactory* y la clase *FileCorrespondences*. El *ATAssistant* también dirige la creación de los vocabularios mediante la interfaz *VocabularyFactory* y la clase *FileVocabulary*. Ambas interfaces *CorrespondencesFactory* y *VocabularyFactory* y las clases *FileVocabulary* y *FileCorrespondences* forman parte del paquete *language*. La creación de las *propiedades sociales* se realiza mediante una interfaz *PropertyFactory* en el paquete *repositories*.

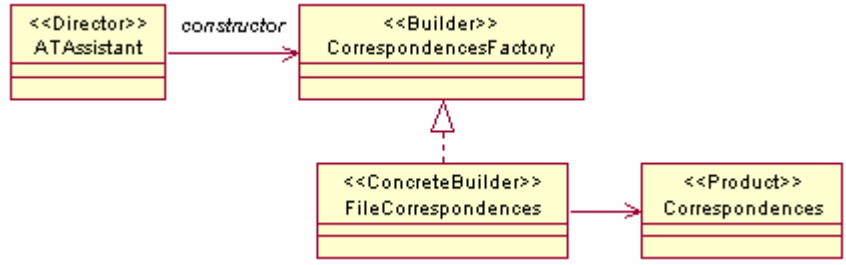


Fig. 89. Patrón de diseño *constructor* para *ATAssistant*.

Otro problema del diseño era la utilización de herramientas de modelado externas. Las necesidades del ATA respecto a esas herramientas estaban definidas por los requisitos y las interacciones. Se deseaba, por ejemplo, obtener especificaciones, guardar modificaciones o recibir eventos sobre la edición de los modelos. Por otra parte no se deseaba/podía alterar las herramientas originales para modelado de las metodologías de la ISOA a fin de adaptarlas al uso con el ATA. La solución fue implementar el patrón *adaptador* ("*adapter*"). Se creó una interfaz *Wrapper* que encapsulaba la funcionalidad requerida por el ATA a las herramientas. Para hacer que las herramientas de modelado puedan ser usadas a través de *Wrappers* se han creado implementaciones específicas de la interfaz. En el caso del IDK usado en la experimentación, la herramienta es representada por la clase *Browser* y el adaptador es el *IngeniasIDKWrapper*. La implementación del patrón puede verse en la Fig. 90.

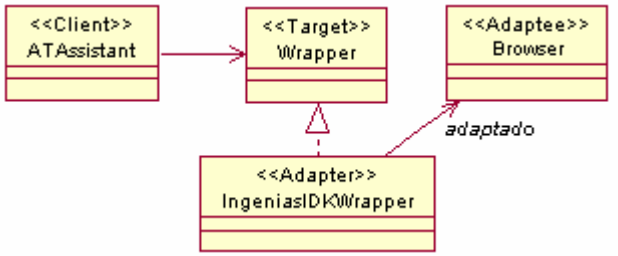


Fig. 90. Patrón de diseño *adaptador* para el IDK.

Una observación adicional acerca de las clases que implementan la interfaz *Wrapper* y las herramientas de modelado es que su implementación debería considerar también el patrón *observador*. Uno de los modos de trabajo contemplados con el ATA es que éste monitorice la edición del usuario y analice las contradicciones según surgen en los modelos (ver sección 6.4). La implementación eficiente de este proceso aconseja que las herramientas den la posibilidad de suscribir observadores a algunos de sus eventos. En caso contrario, el observador que implementa la interfaz *Wrapper* debería interrogar periódicamente a la herramienta para conocer el estado de la edición, con el consiguiente consumo de recursos.

El patrón central en el diseño del ATA se relaciona con el uso del lenguaje de grafos para representar la información. Los problemas planteados al asistente se describen con especificaciones en dicho lenguaje. Los procesos, como la identificación o traducción de patrones, son acciones cuya realización viene determinada por la "frase" sobre la que trabajan. Fruto de contemplar la acción del ATA como transformaciones de estructuras en el lenguaje de grafos, se ha organizado el diseño en torno al patrón intérprete ("*interpreter*"). La Fig. 91 refleja la implementación del *intérprete* para la localización de patrones (interfaz *PatternFinder*). El mismo patrón es adoptado para la traducción de estructuras (interfaz *Mapper*). La adopción del patrón *intérprete* proporciona sobre todo flexibilidad para contemplar nuevas operaciones sobre los grafos.

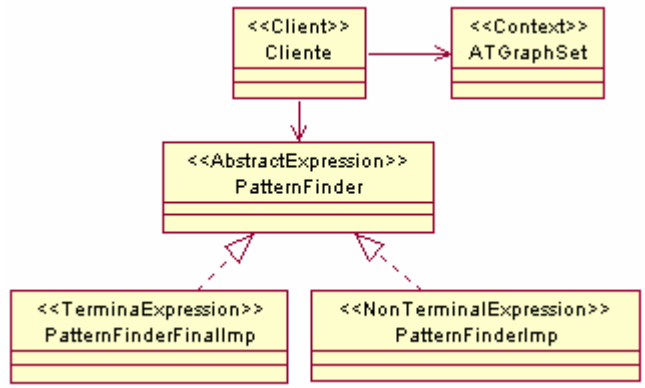


Fig. 91. Patrón de diseño *intérprete* para *PatternFinder* en el paquete *matching*.

El último patrón a destacar en la implementación es la *fachada* ("*facade*"). Las clases del ATA proporcionan funcionalidades de traducción, verificación de lenguajes, localización de patrones o aplicación de distintos repositorios de *propiedades sociales*. Para evitar que los posibles clientes de estos servicios deban conocer toda la jerarquía de clases y sus métodos, el ATA cuenta con dos clases que actúan como intermediarios para los clientes. Estas *fachadas* son la interfaz *Assistant* y dentro del ATA la interfaz *Repository*. Ambas actúan como punto común de acceso a la funcionalidad de las clases del ATA y se ocupan de organizar y redireccionar las peticiones de los clientes a las clases capaces de satisfacerlas.

6.5.2.3. Interacciones de alto nivel

Los componentes de los paquetes vistos en la Fig. 88 realizan las interacciones que llevan a cabo las peticiones del usuario. Como se puede deducir de la estructura de paquetes, las implementaciones de las técnicas de la TA (en este caso la GCR y las contradicciones) guardan muchos puntos en común. Estos aspectos comunes del sistema también se reflejan en las interacciones que se muestran en este apartado.

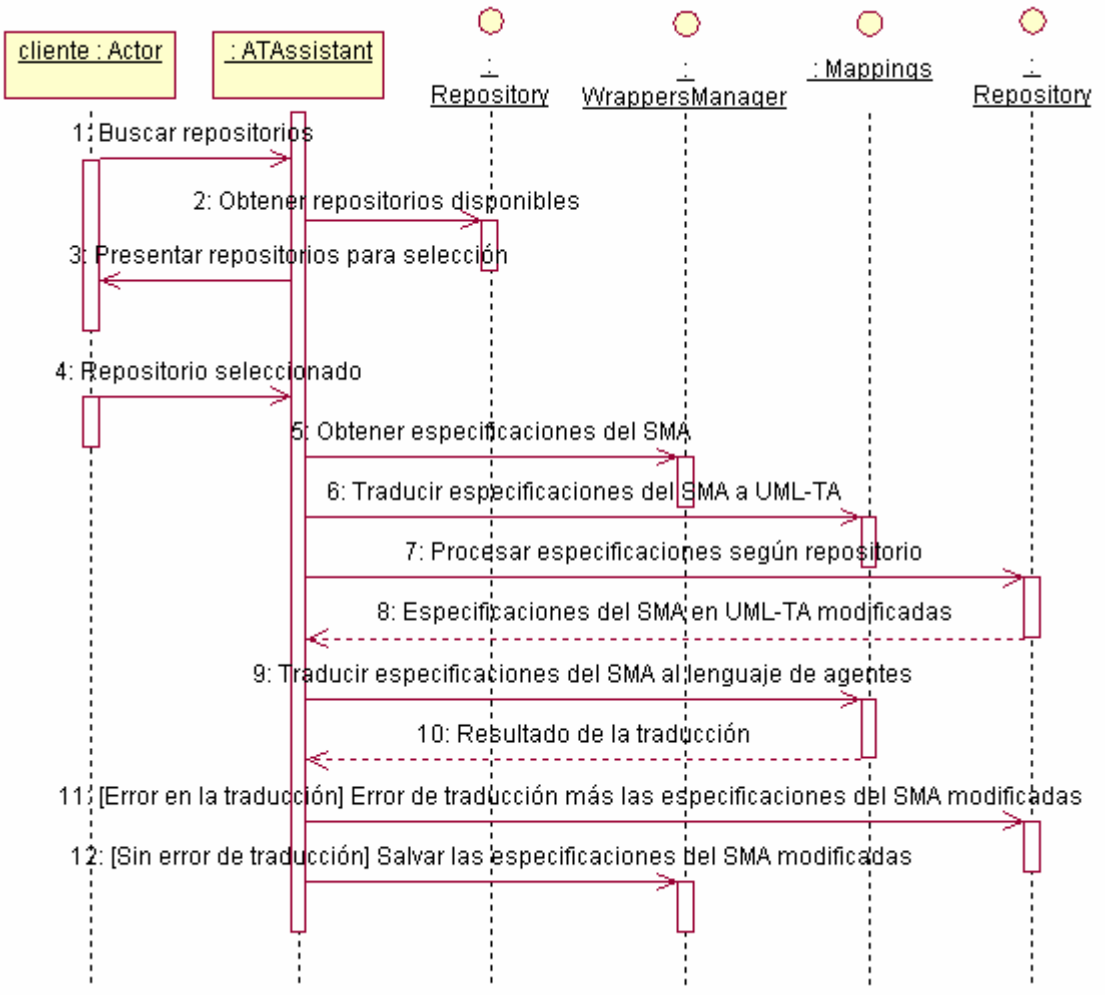


Fig. 92. Interacciones comunes a los repositorios del ATA con origen en el usuario.

La Fig. 92 refleja la parte común de las interacciones sobre los repositorios del ATA cuando el usuario invoca explícitamente al asistente. En este caso se empieza seleccionando la técnica a aplicar, representada por un repositorio. Después se obtienen las especificaciones del SMA y se traducen a UML-TA. A continuación se cede el control al repositorio para que trabaje sobre los elementos traducidos. Cuando

el repositorio termina de actuar devuelve unas especificaciones modificadas. Éstas se intentan traducir al lenguaje original basado en agentes. Aquí pueden surgir errores de traducción debido a que las especificaciones han sido cambiadas incorrectamente. En caso de error las especificaciones se devuelven al repositorio con una indicación de los elementos que han producido el problema. El proceso volvería de este modo al mensaje 7. Si no hay errores en la traducción, las especificaciones se devuelven a la fuente de origen de los modelos.

Aunque no se ha mostrado, la interacción de la Fig. 92 incluye en el paso de traducción (mensaje 6) el acceso a las funciones del paquete *language* para obtener las definiciones de los vocabularios y correspondencias a utilizar.

El otro posible modo de operación del asistente es reactivo a los cambios en las especificaciones. El ATA realiza una monitorización permanente de la herramienta de modelado e informa al usuario de las contradicciones que va detectando. Este modo queda reflejado en la Fig. 93.

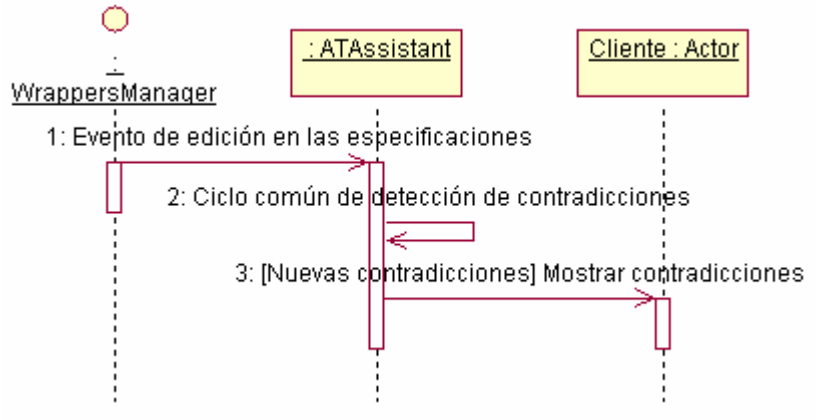


Fig. 93. Monitorización del proceso de edición por el ATA.

Los eventos de edición que se pueden detectar vienen determinados por las clases que implementan la interfaz *Wrapper* gestionadas por el *WrappersManager*. Cuando se detecta uno de estos eventos el *ATAssistant* inicia un ciclo de detección de contradicciones. Este ciclo es similar a la interacción de la Fig. 92 pero el usuario no selecciona ningún repositorio, ya que se trabaja directamente con el de contradicciones (el *CTA* de la Fig. 88). El repositorio de contradicciones detecta los posibles nuevos problemas en las especificaciones. Puesto que el repositorio puede guardar memoria de las contradicciones detectadas en su última invocación es capaz de señalar solamente las nuevas. En el caso de que existan nuevas contradicciones, el ATA las muestra en su propia interfaz (el paquete *targets* de la Fig. 88). Así, si el usuario lo desea puede pasar a examinar las contradicciones detectadas.

La eficiencia de la monitorización depende en buena medida de la información suministrada desde la herramienta de modelado. Detectar todas las posibles contradicciones en unas especificaciones es un proceso que consume un tiempo considerable. Este tiempo se puede reducir con la información adecuada en el evento. Por ejemplo, si se conoce el elemento editado en la herramienta de la ISOA se puede limitar la detección a las contradicciones en las que pueda participar dicho elemento.

A continuación se pasan a examinar las interacciones específicas en los repositorios para la captura de requisitos con la GCR y la detección de contradicciones. En la Fig. 92 serían las acciones que ocurrirían a partir de la llegada al *repository* del mensaje 7 para procesar las especificaciones. Al igual que en la Fig. 92, las siguientes interacciones omiten el proceso en el cual las implementaciones de la interfaz *repository* usan el paquete *language* para obtener la definición de sus *propiedades sociales*.

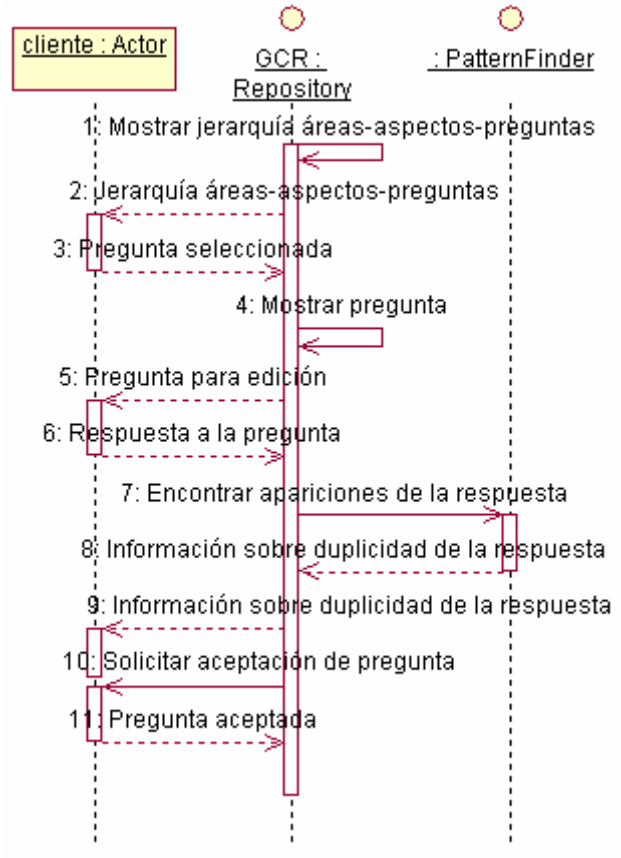


Fig. 94. Interacción para el uso de la GCR.

En primer lugar se describen las interacciones específicas para el uso de la GCR con la Fig. 94. En este caso se trata de una interacción que siempre ha de ser solicitada por el usuario (ver Fig. 92). La interfaz del repositorio para la GCR (*GCR* en la Fig. 94) presenta al usuario la jerarquía de *áreas*, *aspectos* y *preguntas* de la GCR. Con esta información el usuario selecciona la *pregunta* a responder. A continuación, la GCR selecciona en las especificaciones elementos que puedan ser usados en las ranuras de la *pregunta*[6] y presenta ésta al usuario para su edición. La

[6] Las *preguntas* se representan con patrones que son marcos con ranuras modificables. Para más información sobre el uso de la GCR ver la sección 4.4.

respuesta a la *pregunta* se busca después en las especificaciones existentes para evitar duplicidades en la información. Esta situación puede surgir cuando sólo se da una respuesta parcial a una *pregunta* que anteriormente fue concretada con más detalle. En ese caso, la nueva respuesta es posiblemente redundante. Las identificaciones de la respuesta en las especificaciones también son mostradas al usuario. Por último se le solicita que acepte la incorporación de la respuesta a las especificaciones si considera que aporta información relevante.

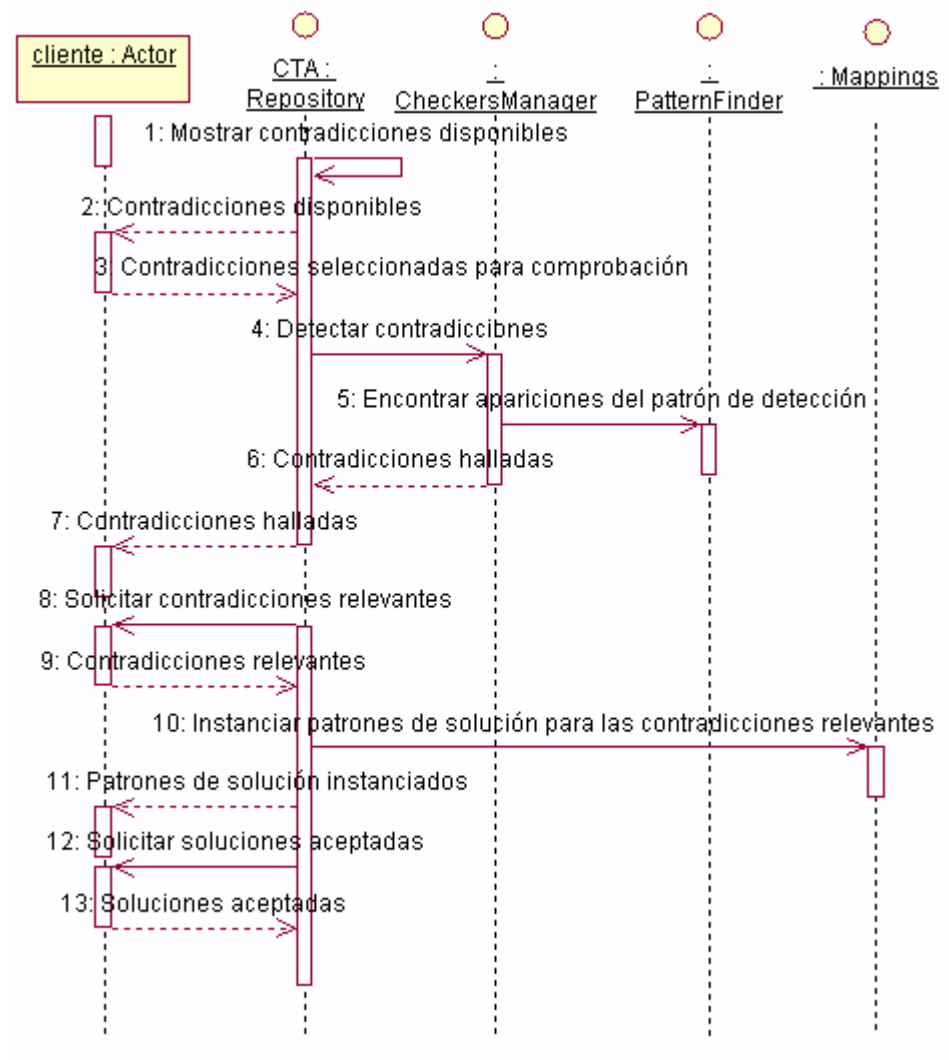


Fig. 95. Interacción para la comprobación de contradicciones.

La Fig. 95 describe las interacciones para comprobar las contradicciones cuando la acción es solicitada por el usuario (ver Fig. 92). La interfaz del repositorio para las contradicciones *(CTA* en la Fig. 95) comienza solicitando al usuario que elija las

contradicciones a detectar. Aunque se ha resumido y no se muestra en el diagrama, este paso implica que el usuario pueda editar los patrones de detección de las contradicciones. En esta edición puede reemplazar parte de las variables en los patrones por valores ya presentes en las especificaciones[7]. Una vez editados, los patrones son buscados en las especificaciones mediante clases que implementan la interfaz *PatternFinder*. Las contradicciones halladas son sometidas al juicio del usuario. Si éste las encuentra relevantes, se le plantean los patrones de solución de las contradicciones instanciados. Esta instanciación es hecha mediante los *Mappings*. La contradicción es usada como una correspondencia con la que traducir el patrón de detección instanciado. En esta correspondencia el patrón de detección actúa como patrón origen y el de solución como patrón destino. Finalmente, las soluciones aceptadas por el usuario son incorporadas a las especificaciones.

La interacción de la Fig. 95 cambia para la monitorización de la Fig. 93. En ese caso sólo se intercambian los mensajes 4 a 7 de la Fig. 95. Se detectan todas las posibles contradicciones y sólo se notifican.

6.5.3. Integración con metodologías

La implementación actual del ATA se encuentra integrada en el entorno *INGENIAS Development Kit* (http://ingenias.sourceforge.net). Para este entorno y la metodología INGENIAS se han creado implementaciones de los paquetes *wrappers* y *targets*, se ha descrito el lenguaje de INGENIAS y se han dado sus correspondencias. El resto de la funcionalidad básica descrita en los apartados anteriores de esta sección está también operativa, aunque quedan varios puntos en construcción relativos a la interacción gráfica con el usuario y la integración entre componentes. Así, la creación de las *propiedades sociales* de los repositorios sólo se puede hacer visualmente en el IDK en cuanto a los diagramas UML-TA. Una muestra de la interfaz del ATA en el IDK para la edición de los patrones puede verse en la Fig. 96. Una vez hecha esta edición hay que modificar el fichero XML de las propiedades sociales (ver sección 6.6.3) para dar valores a algunos de sus elementos. Del mismo modo, los repositorios carecen en la actualidad de una interfaz gráfica, siendo la comunicación con el usuario meramente textual o a través de la edición de los ficheros de configuración.

La utilización del ATA con otras metodologías sólo requiere proporcionar las interfaces para acceder a sus herramientas y configurar el uso de su vocabulario.

La adaptación a la herramienta requiere escribir nuevos componentes de los paquetes *wrappers* y *targets* que implementen las APIs documentadas al respecto. Las funcionalidades contempladas por estas interfaces fueron vistas en la sección 6.5.2.

El uso del vocabulario de nuevas metodologías se puede configurar mediante ficheros XML. Gracias a ello, buena parte del trabajo invertido puede ser reutilizado en otros proyectos que usen la misma metodología de agentes. También las *propiedades sociales* se describen con XML y pueden ser reutilizadas entre proyectos que traten de comprobar las mismas contradicciones. La descripción de estos ficheros puede verse en la siguiente sección.

[7] Los patrones son marcos cuyas ranuras pueden ser editadas por los usuarios. Para más información sobre el proceso de comprobación de contradicciones ver la sección 5.3.2.

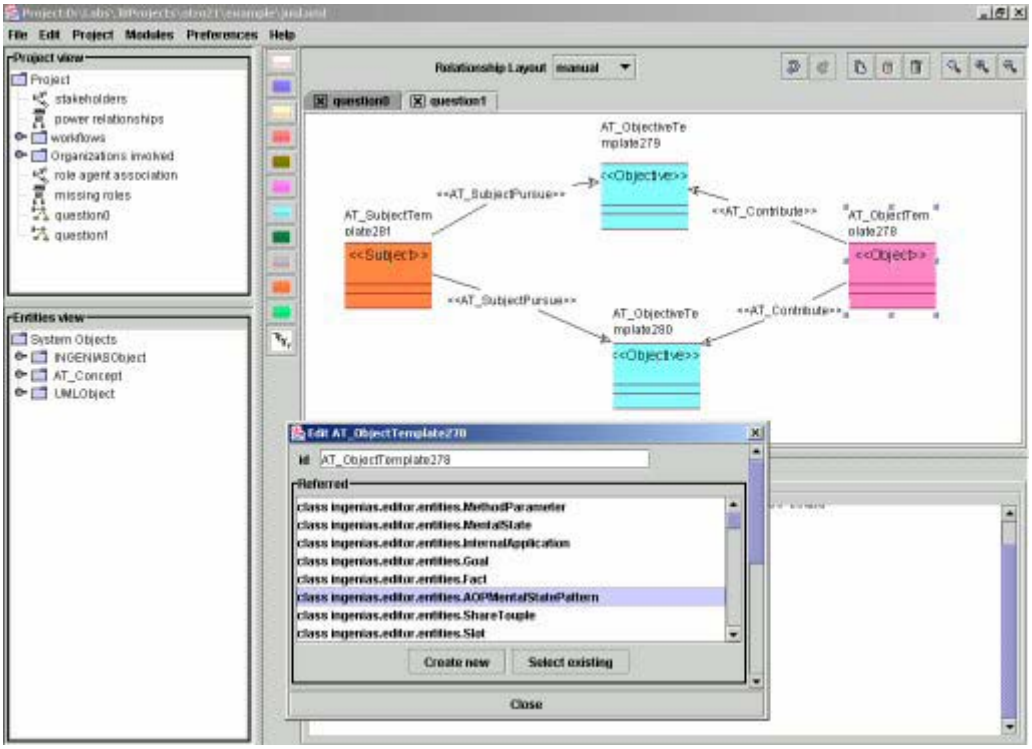


Fig. 96. INGENIAS IDK con el ATA integrado.

6.6. Especificaciones y configuración

Como se indicó en la sección anterior, el funcionamiento del ATA es configurado parcialmente mediante ficheros XML. Los aspectos que pueden ser especificados de este modo son:

- *Lenguajes*. En la sección 6.5 sobre la arquitectura del asistente, se presentaron los lenguajes manejados por el ATA como lenguajes de descripción de grafos especificados con el metalenguaje GOPRR [Lyytinen & Rossi 1999]. Estos lenguajes son UML-TA y los de las metodologías orientadas a agentes. Sus especificaciones permiten realizar validaciones básicas sobre las primitivas usadas en los modelos de los SMAs y en las correspondencias.
- *Correspondencias*. Su especificación con XML permite alterar declarativamente las identificaciones entre estructuras de lenguajes para la traducción.
- *Propiedades sociales*. Las *propiedades sociales* de los repositorios se especifican con el lenguaje UML-TA. Crear nuevas propiedades en los repositorios existentes sólo implica alterar estos ficheros. La creación de nuevos repositorios

supone además de crear los ficheros para sus propiedades, crear paquetes para su interfaz gráfica en el ATA.

El procesamiento de los ficheros que definen vocabularios, correspondencias y *propiedades sociales* es realizado en el paquete *languages*.

Los siguientes apartados en esta sección presentan las DTDs que especifican estos ficheros de configuración y las explican mediante ejemplos.

6.6.1. Lenguaje

El asistente necesita lenguajes para describir dos tipos de informaciones: por un lado la forma UML de las *propiedades sociales* en términos de los conceptos de la TA; por otra parte las especificaciones de los SMAs. En ambos casos se trata de información que puede ser vista como la descripción de un grafo con GOPRR. Estas definiciones se usan para realizar una verificación básica de las especificaciones.

La definición de los lenguajes se realiza mediante ficheros XML que siguen la DTD de la Fig. 97. Los elementos principales de esta definición son "*language*", "*entity*", "*relationship*", "*role*" y "*attribute*".

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT vocabulary (language, entity+, relationship*)>
<!ELEMENT language EMPTY>
<!ATTLIST language name CDATA #REQUIRED>
<!ELEMENT attribute (name, value?, type?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT entity (type, attribute*)>
<!ELEMENT relationship (type, attribute*, role+)>
<!ELEMENT role (type, attribute*)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

Fig. 97. DTD para la especificación de lenguajes en el ATA.

Los elementos de la DTD se explican a continuación con el ejemplo de la Fig. 98, que muestra parte de la definición del lenguaje de la TA.

El elemento "*language*" sólo establece el nombre del lenguaje especificado. En este caso se trata del lenguaje "AT", i.e. "*Activity Theory*".

El elemento "*entity*" describe las entidades del lenguaje proporcionando un nombre para su tipo. Es el caso de la entidad "*AT_Artifact*".

El elemento "*relationship*" establece las relaciones válidas en el lenguaje. Al igual que se hacía con "*entity*" se le suministra un nombre a su tipo. Un ejemplo es la relación "*AT_Contribute*". La descripción de la relación también incluye los roles (en el sentido de GOPRR) a través de los cuales se conecta la relación con las entidades.

El elemento "*role*" describe la conexión de las relaciones con las entidades. Los roles también poseen un atributo para el nombre de su tipo. Además este elemento incluye un atributo para indicar la entidad que juega ese rol, i.e. atributo "*player*", y si

es origen o destino de la relación en el caso de que sea relevante. Un rol de "*AT_Contribute*" es "*AT_Contribute_Source*".

El último elemento considerado es "*attribute*", que se emplea para asociar atributos a cualquiera de los elementos anteriores. Los atributos quedan definidos por su nombre y su tipo, elegido entre varios predefinidos, e.g. *String* o *Boolean*.

```
<!DOCTYPE vocabulary SYSTEM "D:\LanguageVocabulary.dtd">
<vocabulary>
   <language name="AT"/>
   <entity>
      <type> AT_Artifact </type>
   </entity>
...
   <relationship>
      <type> AT_Contribute </type>
      <attribute>
         <name> Contribution </name>
         <type> String </type>
      </attribute>
      <role>
         <type> AT_Contribute_Source </type>
         <attribute>
            <name> origin </name>
            <value> true </value>
         </attribute>
      </role>
      <role>
         <type> AT_Contribute_Target </type>
         <attribute>
            <name> origin </name>
            <value> false </value>
         </attribute>
         <attribute>
            <name> Player </name>
            <value> AT_Artifact </value>
         </attribute>
      </role>
   </relationship>
...
</vocabulary>
```

Fig. 98. Ejemplo de XML para la especificación de lenguajes en el ATA.

6.6.2. Correspondencias

Tanto el proceso de captura de requisitos como el de comprobación de contradicciones trabajan sobre la base del lenguaje de la TA. Su aplicabilidad a las metodologías de agentes depende de la existencia de correspondencias para la traducción desde el vocabulario de la metodología al de la TA.

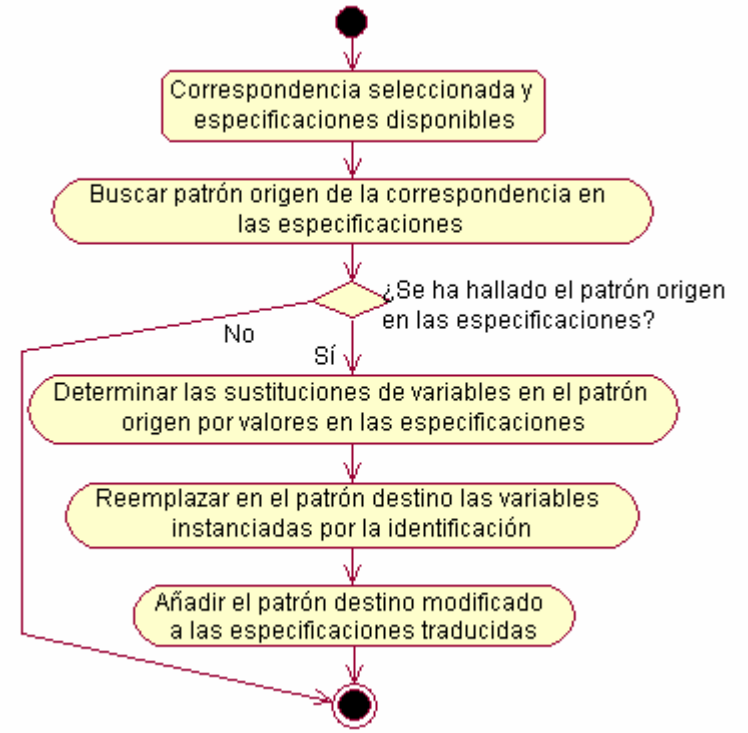


Fig. 99. Traducción de una correspondencia en el ATA.

Las correspondencias describen pares de subgrafos, i.e. patrones estructurales, en los lenguajes de origen y destino de la traducción. Estos grafos pueden incluir como valores de sus propiedades variables o valores fijos. El proceso de traducción de una correspondencia en el ATA se muestra en la Fig. 99. El proceso comienza buscando grupos de elementos en las especificaciones que tengan la misma estructura que el patrón de origen de la correspondencia. Si se realiza una identificación, hay que comprobar que los valores del patrón y de la estructura de los modelos son compatibles. En el caso de que lo sean, se puede establecer una función de sustitución de las variables en las ranuras del patrón de origen por los valores obtenidos de las especificaciones. La parte del modelo original que se identificó con el patrón de origen se traduce por una instanciación del patrón de destino de la correspondencia. En esta instanciación se aplica la función de sustitución anterior sobre el patrón de destino, reemplazando los valores de las variables por aquellos que hayan quedado fijados según la identificación del patrón de origen.

Hay que recordar que las correspondencias se aplican en el orden determinado por su nivel de traducción. El uso de estos niveles no aparece en la Fig. 99 pero sí en el proceso completo de traducción (ver sección 3.5.3) y es contemplado en la arquitectura del ATA con el paquete *wrappers* y sus interfaces *WrappersManager* y *Wrappers* (ver sección 6.5.2).

El lenguaje para especificar correspondencias sigue la DTD de la Fig. 100. Parte del lenguaje de correspondencias, como los elementos "*relationship*" o "*role*", son como los ya comentados en la sección anterior sobre especificación de vocabularios. Los elementos nuevos se desglosan a continuación. Para ello se emplea el ejemplo de la Fig. 101. Éste describe una correspondencia para una relación de la metodología INGENIAS [Pavón & Gómez-Sanz 2003] con el lenguaje de la TA (ver sección 3.4).

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT correspondences (language_source,
   language_target, match+)>
<!ELEMENT language_source EMPTY>
<!ATTLIST language_source name CDATA #REQUIRED>
<!ELEMENT language_target EMPTY>
<!ATTLIST language_target name CDATA #REQUIRED>
<!ELEMENT match (level, source, target)>
<!ELEMENT level EMPTY>
<!ATTLIST level value (0 | 1 | 2 | 3 |4) #REQUIRED>
<!ELEMENT source (entity | relationship)+>
<!ELEMENT target (entity | relationship)+>
<!ELEMENT entity (name, type)>
<!ELEMENT relationship (name, type, attribute*, role+)>
<!ELEMENT role (name, type, attribute*)>
<!ELEMENT attribute (name, value?)>
<!ELEMENT name (#PCDATA | v)*>
<!ELEMENT type (#PCDATA | v)*>
<!ELEMENT value (#PCDATA | v)*>
<!ELEMENT v (#PCDATA)>
```

Fig. 100. DTD para la descripción de correspondencias.

"*language_source*" y "*language_target*" establecen los lenguajes origen y destino de la traducción.

Por cada correspondencia (i.e. "*match*") se incluyen varios datos. El elemento "*level*" indica nivel del proceso de traducción (ver sección 3.5.3). Estos niveles permiten traducir unas especificaciones en varias iteraciones y simplifican por tanto los patrones de traducción. Los otros elementos obligatorios son "*target*" y "*source*" que delimitan el patrón de origen y su traducción.

La definición de los patrones de origen y destino/traducción se realiza con el lenguaje de la TA definido parcialmente en la sección anterior. En el ejemplo se puede apreciar el uso de valores constantes ' "*AT_Contribute*" ' y variables ' *<v> "v3" </v>* '. En el ejemplo de la Fig. 101, la variable "*v3*" permite asociar el mismo valor del jugador (atributo "*player*") al rol "*GTSatisfiessource*" y a su

traducción, el rol "*AT_Contribute_Source*". Los atributos de entidades y relaciones son las ranuras de los patrones de las *propiedades sociales* (ver sección 3.6).

```
<!DOCTYPE vocabulary SYSTEM "D:\Translation.dtd">
<correspondences>
   <language_source name="Ingenias"/>
   <language_target name="AT"/>
   <match>
      <level value="1"/>
      <source>
         <relationship>
            <name> <v> v0 </v> </name>
            <type> GTSatisfies </type>
            <role>
               <name> <v> v1 </v> </name>
               <type> GTSatisfiessource </type>
               <attribute>
                  <name> player </name>
                  <value> <v> v3 </v> </value>
               </attribute>
            </role>
         </relationship>
      </source>
      <target>
         <relationship>
            <name> <v> v0 </v> </name>
            <type> AT_Contribute </type>
            <attribute>
               <name> Contribution </name>
               <value> Needed </value>
            </attribute>
            <role>
               <name> <v> v1 </v> </name>
               <type> AT_Contribute_Source </type>
               <attribute>
                  <name> player </name>
                  <value> <v> v3 </v> </value>
               </attribute>
            </role>
         </relationship>
      </target>
   </match>
...
</correspondences>
```

Fig. 101. Ejemplo de XML para la descripción de correspondencias.

6.6.3. Propiedades sociales

Las *propiedades sociales* que el asistente ha de considerar son otro aspecto descrito mediante XML. Una vez más se emplea el lenguaje de la TA y su representación es muy similar a la de las correspondencias del apartado anterior. La Fig. 102 muestra la DTD para los ficheros de definición de *propiedades sociales* y la Fig. 103 un ejemplo con parte de la contradicción del *Significado Dual* vista en la sección 5.4.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT repository (language, ATProperty*)>
<!ELEMENT language EMPTY>
<!ATTLIST language name CDATA #REQUIRED>
<!ELEMENT ATProperty (description, match, solution)>
<!ATTLIST ATProperty name CDATA #REQUIRED>
<!ELEMENT match (description, source, target)>
<!ELEMENT solution (description, source, target)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT source (entity | relationship)*>
<!ELEMENT target (entity | relationship)*>
<!ELEMENT entity (name, type, attribute*)>
<!ELEMENT relationship (name, type, attribute*, role+)>
<!ELEMENT role (name, type, attribute*)>
<!ELEMENT attribute (name, value?)>
<!ELEMENT name (#PCDATA | v)*>
<!ELEMENT value (#PCDATA | v)*>
<!ELEMENT type (#PCDATA | v)*>
<!ELEMENT v (#PCDATA)>
```

Fig. 102. DTD para la descripción de *propiedades sociales*.

Como se aprecia en el ejemplo de la Fig. 103, la definición de *propiedades sociales* comparte numerosos elementos con las de los lenguajes y las correspondencias. Los elementos específicos de las propiedades son las marcas "*ATProperty*", "*match*" y "*solution*".

El elemento "*ATProperty*" encierra la definición de una *propiedad social*. Incluye atributos para dar su nombre y descripción. Los patrones estructurales asociados a la propiedad se encierran con los elementos "*match*" (i.e. patrón de detección) y "*solution*" (i.e. patrón de solución) que también incluyen atributos para su descripción.

La definición de los patrones de detección y solución es como la vista en el apartado 6.6.2 para las correspondencias, con elementos "*source*" y "*target*". En el ejemplo, tanto el patrón "*source*" como el "*target*" corresponden al patrón de detección de la *propiedad social* del *Significado Dual*. La justificación de usar un par de patrones para describir estos elementos estriba en que, aunque las contradicciones se aplican sobre especificaciones ya traducidas, es posible que sean necesarias transformaciones adicionales para su presentación. En el ejemplo de la Fig. 103, sería posible que un *sujeto* persiguiese un objetivo porque juega un rol. Del mismo modo, una *actividad* puede incluir varias tareas de los diagramas. En estos casos, ambas

informaciones sólo se necesitan para la explicación de la contradicción o su solución, pero no para caracterizarlas. Debido a ello, en ocasiones, resulta adecuado reformular la identificación con los modelos para facilitar la comprensión del patrón.

```
<!DOCTYPE vocabulary SYSTEM "D:\Pattern.dtd">
<repository>
   <language name="Contradictions"/>
   <ATProperty name="Twofold Meaning">
      <description> Twofold Meaning </description>
      <match>
         <description> The match pattern </description>
         <source>
            <entity>
               <name> <v> v0 </v> </name>
               <type> AT_Subject </type>
            </entity>
            <relationship>
               <name> <v> v2 </v> </name>
               <type> AT_Pursue </type>
               <role>
                  <name> <v> v3 </v> </name>
                  <type> AT_Pursue_Source </type>
                  <attribute>
                     <name> player </name>
                     <value> <v> v0 </v> </value>
                  </attribute>
               </role>
...
            </relationship>
...
         </source>
         <target>
...
         </target>
      </match>
      <solution>
...
      </solution>
   </ATProperty>
...
</repository>
```

Fig. 103. Ejemplo de XML para la descripción de *propiedades sociales*.

6.7. Conclusiones

Este capítulo ha mostrado una implementación de las técnicas de la TA descritas anteriormente. El Asistente de la Teoría de Actividad (ATA) proporciona soporte a lo largo de un proceso software de la ISOA para el tratamiento de las *propiedades sociales*: en la captura de requisitos mediante la GCR y en el resto del proceso con la detección y solución de contradicciones. En ambos casos el proceso comienza con el ATA proponiendo al usuario patrones en UML-TA que éste edita. En el caso de la GCR, el patrón editado es la respuesta a una *pregunta* y puede ser incorporada a las especificaciones. Para las contradicciones, el ATA busca los patrones editados en los modelos del SMA. Aquellos patrones que encuentra son contradicciones potenciales que ha de valorar el usuario. Para las contradicciones relevantes se plantea un patrón de solución que puede incorporarse a las especificaciones.

Las técnicas consideradas (la GCR y las contradicciones) basadas en la edición, detección y transformación de patrones estructurales han llevado a plantear el ATA como intérpretes de un lenguaje de grafos. Los grafos representan la información de las *propiedades sociales* y de las especificaciones. Los intérpretes realizan la funcionalidad del ATA según las "frases" que son sus argumentos, de acuerdo con las primitivas y las relaciones entre ellas.

Desde el punto de vista de su diseño, la característica más destacable del asistente es su capacidad de adaptación a las necesidades concretas de cada desarrollo en distintos aspectos:

- *Información sobre las metodologías de la ISOA*. La información sobre los modelos de la metodología que se necesita para trabajar con el ATA puede cambiarse mediante especificaciones en XML. De esta manera, los potenciales usuarios pueden realizar modificaciones que se adapten a sus necesidades, considerando nuevas metodologías de agentes o formas de traducirlas.
- *Repositorios de propiedades sociales*. También los repositorios, que representan las técnicas de la TA, se pueden modificar con ficheros XML. Las propiedades asociadas a los repositorios se describen con ficheros. Así se pueden cambiar las *preguntas* de la GCR o añadir nuevas contradicciones. En caso de aplicar nuevas técnicas podría también ser necesario desarrollar las clases que implementaran su interfaz gráfica específica.
- *Interfaces con las herramientas de desarrollo*. La definición de interfaces que permiten al ATA interactuar con distintas herramientas de desarrollo facilita su utilización sobre diferentes metodologías de la ISOA. Los desarrolladores no quedan limitados a usar una determinada herramienta; pueden utilizar conjuntamente la herramienta que deseen y el ATA sin más que implementar los componentes de interfaz entre ambos.
- *Filtros y métricas*. El problema de estos elementos está en su enorme variedad y la necesidad en ocasiones de adaptarlas a *propiedades sociales* concretas. Para mitigar este problema se han descrito las interfaces que se deben implementar al crear estos elementos. Así los desarrolladores pueden añadir nueva funcionalidad al asistente aunque la deben implementar.

Esta forma de trabajar altamente flexible era necesaria porque a lo largo de la presentación de los métodos de la TA se ha incidido en que no se trataba de formulaciones cerradas que no debieran ser alteradas y asociadas a metodologías concretas. Por el contrario, su uso ha de adaptarse a las necesidades específicas de cada problema y realizarse en combinación con otras técnicas de Ingeniería del Software. Tener una herramienta altamente configurable permite adaptarse a estas circunstancias.

Una ventaja adicional de la forma de trabajar descrita es su carácter intuitivo. Las técnicas se configuran mediante la edición de patrones y las demostraciones consisten en su detección ("*pattern matching*"). Esta forma de trabajar es fácil de comprender tanto para los desarrolladores como para los clientes.

La propuesta de implementación también tiene algunas limitaciones. La principal es que los paquetes para desarrollar nuevos repositorios están orientados hacia intérpretes del lenguaje de grafos. Implementar nuevas técnicas de la TA que tengan otras necesidades necesitará un mayor esfuerzo ya que habrá que generar nuevos paquetes auxiliares.

Capítulo 7. Experimentación

Los capítulos anteriores de esta tesis han presentado la infraestructura y métodos genéricos para trabajar con las propiedades sociales, técnicas específicas para la captura de requisitos y la resolución de contradicciones en especificaciones de SMAs, así como herramientas software que implementan los elementos anteriores. El propósito de este capítulo es validar la aproximación global mediante el uso de las técnicas anteriores en la elaboración de un caso de estudio. Éste consiste en el desarrollo de la gestión electrónica de una librería en sus relaciones con clientes y proveedores. El estudio incluye la captura de los requisitos sobre el sistema y el desarrollo de sus especificaciones mediante la detección y eliminación de contradicciones. También se refleja el proceso de traducción de los modelos entre el vocabulario de la TA y el de una metodología de agentes. Tras la presentación del desarrollo se incluye una evaluación del uso de los distintos elementos de la propuesta, i.e. la GCR y las contradicciones, y su aportación a las especificaciones finales. El capítulo concluye con una discusión de los beneficios y limitaciones de la propuesta a la vista de su uso integrado en el caso de estudio.

7.1. Introducción

Este capítulo recoge la experimentación para evaluar las propuestas de esta tesis. Se trata de un caso de estudio que desarrolla la especificación de un SMA con las técnicas de la TA para la ISOA vistas en capítulos anteriores. El propósito de este caso de estudio es proporcionar las evidencias para evaluar las ventajas y limitaciones del uso de la TA.

Los métodos de la TA contemplados en el caso de estudio son:

- *Uso de la infraestructura básica*. Se trata de los elementos recogidos en el Capítulo 3. Incluyen la representación de las *propiedades sociales* y la traducción entre éstas y los lenguajes de las metodologías basadas en agentes. La representación de las *propiedades sociales* permite realizar los procesos semi-automáticos relacionados con la captura de requisitos y el tratamiento de contradicciones. Por otra parte, una de las claves de la propuesta de esta tesis es que las técnicas de la TA sean aplicables sobre las metodologías existentes de la ISOA. Las traducciones constituyen este nexo entre la TA y la ISOA. Para comprobar su uso, el desarrollo del caso de estudio se realiza sobre la metodología de agentes INGENIAS complementada con la TA. El paso de información entre ambos lenguajes se muestra con ejemplos de traducciones de especificaciones entre ellos.

- *Captura de requisitos*. Con las técnicas del 0. La captura de requisitos con la GCR se ilustra con el desarrollo de las especificaciones del SMA a partir del planteamiento del problema en lenguaje natural. Se proponen y responden *preguntas* para crear los modelos y se finaliza con su traducción a INGENIAS.
- *Desarrollo guiado por contradicciones*. Con las técnicas del Capítulo 5. Este método consiste en hacer evolucionar las especificaciones mediante la detección y resolución de las contradicciones de la TA en ellas. Este estudio se hace sobre las especificaciones generadas con la GCR.

Tras esbozar en qué va a consistir la experimentación hay que decidir cómo se va a evaluar la propuesta con ella. Se trata de ver en qué aspectos el uso de estas técnicas en un proceso de desarrollo de la ISOA reporta una mejora, qué clase de mejoras y cuáles son sus limitaciones. Para ello tenemos en cuenta los objetivos de estas técnicas. La intención de la GCR es capturar nueva información sobre *propiedades sociales* que afectan al análisis y funcionamiento del SMA. El propósito de usar las contradicciones es detectar problemas en las especificaciones acerca de las *propiedades sociales* y plantear soluciones a los mismos.

Una posible forma de evaluación podría consistir en hacer desarrollos paralelos, con y sin las técnicas de la TA, en varias metodologías de la ISOA. De este modo podrían compararse las diferencias entre las especificaciones en cuanto a la información final y los problemas detectados y resueltos. Sin embargo esta forma de proceder presenta ciertos problemas:

- *Coste*. Realizar las especificaciones con diferentes metodologías requiere contar con desarrolladores que las conozcan suficientemente como para aprovechar sus características particulares. Si los desarrollos se hacen sin suficiente experiencia se generan especificaciones que no explotan las cualidades de las metodologías y, por tanto, conducen a falsas conclusiones en la evaluación.
- *Diferencias entre los desarrolladores*. Analizar el impacto de la TA sobre una metodología requiere utilizar diferentes grupos de personas en el trabajo con la TA y sin ella. Por otra parte, el punto de partida del experimento es la descripción inicial del problema, que es normalmente vaga ya que se sitúa antes de la captura de requisitos. Partiendo de esta situación, los desarrolladores pueden conducir el análisis a unos resultados finales muy diferentes, lo que haría difícil una comparación efectiva.
- *Criterios de evaluación*. Determinar la "calidad" de unas especificaciones es siempre una labor difícil. Se trata de medir la adecuación a las necesidades del cliente y factores de diseño de la solución (e.g. flexibilidad a cambios, consumo de recursos o seguridad). Son características muy diversas cuya importancia depende del problema concreto que se está considerando y que pueden llegar a ser contradictorias (e.g. rapidez vs seguridad). Por tanto resulta prácticamente imposible establecer unos criterios de evaluación centrados en las especificaciones y que sean independientes del problema.
- *Dependencia de la metodología considerada*. Las diferentes metodologías de la ISOA cuentan con métodos y herramientas muy diversos. Los criterios establecidos para comparar las especificaciones pueden resultar de poca utilidad en propuestas donde esos aspectos carecen de relevancia.

Una alternativa al método anterior es centrar la evaluación en estadísticas de uso de las *propiedades sociales* de la GCR y de las contradicciones, así como su relevancia en las especificaciones. Esta evaluación resulta independiente de la metodología en el sentido de que se centra en las técnicas de la TA. Por otra parte, la subjetividad de determinar cuándo una *propiedad social* es "relevante" en un desarrollo es la misma que puede surgir al considerar en la propuesta anterior cuándo un criterio es mejor que otro. Esta alternativa se cristaliza en las siguientes estadísticas de uso que podrían ser proporcionadas por el ATA[8].

En el caso de las *preguntas* se contabiliza:

- *Veces usada*. Indica si la formulación de la *pregunta* pareció útil *a priori*.
- *Veces resuelta*. Sirve de indicador acerca de la facilidad de uso de la *pregunta* y la adecuación de su forma UML-TA.
- *Útil en el desarrollo*. Responder una *pregunta* no implica necesariamente que se capture información útil. Los usuarios son interrogados en la fase de evaluación de la experimentación acerca de la utilidad real que han tenido las respuestas a las *preguntas* en el producto final de la especificación.

En el caso de las contradicciones se contabiliza:

- *Veces usada*. Apunta a si *a priori* pareció que la contradicción debía comprobarse.
- *Veces detectada*. Refleja la frecuencia del problema representado por la contradicción en las especificaciones de SMAs.
- *Veces relevante*. Da una idea de si la descripción del patrón de detección de la contradicción está capturando información que refleja una contradicción real en unas especificaciones.
- *Veces resuelta*. Para eliminar las contradicciones relevantes se plantea su patrón de solución instanciado. Esta cifra indica si la solución constituye una modificación útil de las especificaciones para resolver el problema.

Esta forma de evaluar las técnicas de la TA es simple en cuánto a su realización y sólo en el caso de determinar la información que ha sido relevante requiere un análisis adicional. Dadas estas características en comparación con la propuesta previa se ha optado por usar este modo de evaluación para el experimento.

En cuanto al caso de estudio elegido, éste se basa en el *Juul Møller Bokhandel A/S* [Espen 2001], empleado por una escuela de negocios noruega. Se trata de estudiar la evolución de una librería para estudiantes universitarios que quiere entrar en el negocio de la venta de libros en Internet. El enunciado original del problema se usa para fomentar la discusión entre los alumnos y que estos vayan identificando y solucionando los posibles problemas desde el punto de vista empresarial. Como consecuencia, el enunciado tiene lagunas y errores de apreciación que hay que cubrir con nueva información. Las especificaciones que se plantean con este problema han de describir el SMA que usará *Juul Møller* para la gestión electrónica de su librería.

[8] En la sección 6.2 se introdujeron estas estadísticas como requisitos funcionales del ATA.

En las siguientes secciones de este capítulo se presenta en primer lugar el caso de estudio tal y como se plantea para la escuela de negocios, incluyendo los objetivos de su aplicación. A continuación se analiza la captura de requisitos y la comprobación de contradicciones según las líneas expuestas en esta introducción. Después se presentan las estadísticas de utilización de las técnicas de la TA en el caso de estudio. El capítulo termina con una valoración del uso de las técnicas basadas en la TA para el soporte de metodologías de desarrollo de SMAs a la vista de los resultados obtenidos.

7.2. Caso de estudio

El caso de estudio seleccionado es el *Juul Møller Bokhandel A/S* [Espen 2001]. El problema trata sobre una compañía de librerías noruega (*Juul Møller Bokhandel A/S*) que trabaja con una escuela de negocios. La compañía tiene un acuerdo con la escuela para que los profesores le suministren la lista de libros que utilizarán en sus cursos. A cambio la compañía vende los libros a los estudiantes a un precio especial. Como consecuencia de la irrupción en su negocio de empresas de venta de libros basadas en comercio electrónico, *Juul Møller* se plantea hacer evolucionar también su modelo de negocio. Su objetivo es definir un sistema propio de venta electrónica de libros que se integre y potencie las prácticas actuales de la empresa.

El problema involucra a diferentes actores, cada uno con su propio punto de vista. Hay que considerar cinco grandes grupos: la compañía *Juul Møller Bokhandel A/S*; la Escuela de Negocios Noruega ("*Norwegian School of Management*", NSM); los estudiantes de la NSM agrupados en la "*Student Union*" (SU); otras librerías, no necesariamente competencia de la librería estudiada; los editores y distribuidores de los libros que adquiere la librería. Dentro de *Juul Møller Bokhandel A/S* hay que distinguir también varios actores. Por un lado está Dag Juul Møller que es el propietario. Otro actor importante es Dagfinn Nettland, un recién licenciado en la NSM que propuso a Dag el salto al comercio electrónico de su librería. Los últimos implicados son los veinticinco empleados de *Juul Møller Bokhandel A/S*. La librería se organiza en dos departamentos, uno encargado de ventas y otro de la logística. En cuanto a su distribución física, la compañía tiene cuatro tiendas, tres en los campus de la NSM y una en el centro de Oslo.

El interés de Dagfinn por el comercio electrónico para *Juul Møller Bokhandel A/S* viene de su propia experiencia como estudiante de la NSM. Uno de sus cursos comenzó sin que los libros que usarían para las discusiones estuvieran disponibles en la librería. El profesor optó por no retrasar la discusión de dichos libros. Entonces muchos estudiantes tomaron la decisión de comprarlos por Internet, en librerías como Amazon.com (http://amazon.com) o Blackwell's (http://www.blackwell.com). Dagfinn quedó sorprendido porque esas compañías, con sedes en EEUU o el Reino Unido, podían proporcionar los libros solicitados en muy poco tiempo y a precios muy similares a los de *Juul Møller*. Dagfinn comentó sus observaciones a Dag y éste decidió informarse con mayor profundidad.

Dag analizó la situación de su compañía ante el reto del comercio electrónico en su contexto particular. *Juul Møller* es una compañía de tamaño medio que no puede competir con los recursos de grandes empresas de ámbito mundial. Su punto fuerte

está en su especialización. Tienen una larga experiencia en el mercado noruego de libros de negocios y administración, además de una fuerte implantación en la NSM.

La NSM y *Juul Møller* tienen un acuerdo acerca de la venta de libros. La NSM comunica a *Juul Møller* el catálogo del curso unos meses antes de que empiece el año. Para cada curso se indican los libros obligatorios y recomendados. El problema es que estas especificaciones suelen llegar con retraso y no son fiables, ya que están sujetas a cambios de última hora por las facultades o los profesores de las asignaturas. A cambio de esta información *Juul Møller* paga un alquiler establecido por sus tres tiendas en la NSM y un porcentaje fijo de sus ventas a la SU.

Otra de las grandes dificultades a las que ha de enfrentarse *Juul Møller* es la obtención de los libros de los distribuidores. La distribución cambia mucho según los casos. Normalmente los libros extranjeros son manejados desde grandes centros de distribución a nivel europeo. Si *Juul Møller* necesita libros envía una petición por fax o correo electrónico al centro del correspondiente distribuidor. La respuesta del distribuidor por estos medios ordinarios puede tardar mucho, hasta dos semanas. Muchas veces este tiempo no es admisible y hay que llamar por teléfono para obtener una información más rápida. También la imprecisión de los inventarios automatizados puede originar consultas directas al distribuidor. Esta clase de llamadas sólo son realizadas por el propio Dag y unos pocos empleados que conocen los contactos necesarios en los distribuidores. En todo caso, dependiendo de dónde estén disponibles, los libros pueden venir finalmente del centro de distribución europeo o, en ciertos casos, de su editorial americana. Esto implica una fuerte variación en el tiempo de envío, que puede oscilar desde una semana hasta una par de meses. Finalmente, en caso de no haber ejemplares en ningún punto de la cadena de distribución, *Juul Møller* puede tratar de conseguirlos en otras librerías.

En cuanto a las tecnologías de la información en *Juul Møller*, se puede decir que dos de sus veinticinco empleados trabajan fundamentalmente con ellas. Sus labores se relacionan sobre todo con mantenimiento y captura de datos.

La exposición del caso de estudio concluye con algunas reflexiones de Dag. Tras pensar mucho en el problema, Dag no deseaba simplemente hacer otra web que sólo contuviera información acerca de la empresa. Por otra parte tampoco deseaba poner fin a una larga historia de cooperación y servicio con la NSM y sus estudiantes cuando estos últimos pudieran salir fuera del sistema normal y comprar sus libros directamente en Internet.

En este preciso punto se sitúa el comienzo de la especificación del SMA para comercio electrónico destinado a la empresa *Juul Møller Bokhandel A/S*.

7.3. Captura de requisitos

La captura de requisitos aplicada en este problema se basa en la GCR introducida en el 0. En esta sección se incluyen las *preguntas* planteadas a los usuarios y sus contestaciones, así como diagramas que sintetizan los resultados obtenidos.

En primer lugar se plasma con la ayuda de las *preguntas* de la GCR el conocimiento del enunciado del problema. Después el análisis se centra en aclarar las motivaciones de los involucrados que no están claras en dicho enunciado. Cuando se

termina de establecer el contexto, se busca determinar los objetivos que se persiguen con el sistema y esbozar cómo ha de ser su operacionalización. Finalmente se empieza a desarrollar la arquitectura para el sistema en términos de sus agentes y de los conceptos asociados a estos.

7.3.1. Modelo inicial de los requisitos

El enunciado del caso de estudio aborda varios aspectos fundamentales: quiénes son los actores implicados; la motivación del salto al comercio electrónico; cómo funciona el negocio de *Juul Møller* en relación con la NSM y los estudiantes; los problemas con la distribución de libros; también hace un breve repaso de la situación actual en la empresa en cuanto a tecnologías de la información; por último esboza las expectativas de Dag acerca de la nueva orientación de su compañía. A continuación pasamos a analizar en detalle estos puntos.

Todos los aspectos anteriores están relacionados con el contexto del futuro sistema. En la GCR corresponden fundamentalmente a las *áreas* de *Medios/fines* y *Entorno*.

El primer punto a considerar son los participantes. En este caso el *aspecto* 1.1 interroga sobre los actores que participan en las actividades relacionadas con el nuevo sistema. Estas *preguntas* permiten recoger qué actores aparecen en el entorno y cómo se agrupan a partir de la descripción inicial de los individuos y grupos involucrados en el problema. A modo de ejemplo de cómo se han resuelto estas *preguntas* se incluyen la 1.1.2 y la 1.1.4.

La *pregunta* 1.1.2 "¿Quiénes serán los actores cuyas actividades generen los datos/entradas necesarios para el componente? ¿Quiénes serán los actores que participen en los procesos que generan los datos/entradas necesarios para el componente?" se refleja en la Fig. 104. En este caso se está recogiendo quiénes serán los actores, i.e. *Distribuidor*, que proporcionen los datos sobre los libros en los almacenes para el SMA, i.e. *Sistema*. El patrón de la *pregunta* en la GCR tiene como ranuras variables iniciales los nombres de las cinco entidades distintas de la figura. En este caso el usuario ha cambiado los elementos *Actividad* y *Sistema* de la *pregunta*. Como se indicó en el 0, se pueden dar respuestas parciales a las *preguntas*.

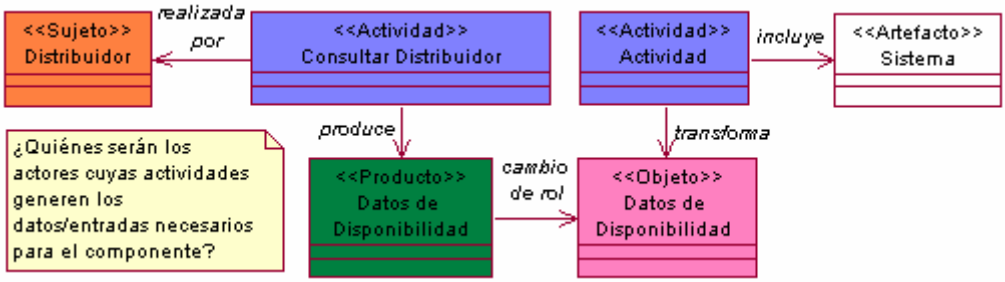


Fig. 104. *Pregunta* 1.1.2 sobre los actores que generan datos en *Juul Møller Bokhandel A/S*.

A propósito de la *pregunta* 1.1.2 y para seguir obteniendo nuevos actores relacionados con el problema puede plantearse la *pregunta* 1.1.4 "¿Quiénes son los actores que interactúan con los anteriores?". Esta *pregunta* ayuda a descubrir actores

que en principio no parecen relacionados con el sistema pero que sí tienen una influencia indirecta en el problema. Una posible respuesta para el *sujeto Distribuidor* de la Fig. 104 aparece en la Fig. 105. La interacción entre ambos *sujetos* se representa con la *Actividad*. En este caso tampoco se ha dado un nombre a la interacción.

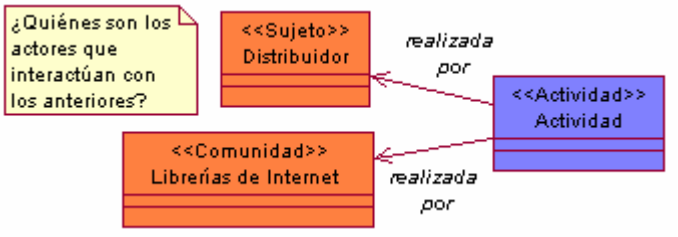


Fig. 105. *Pregunta* 1.1.4 sobre actores interrelacionados en *Juul Møller Bokhandel A/S*.

La Fig. 106 sintetiza una vista parcial de los resultados de las *preguntas* del *aspecto* 1.1. Se muestra la estructura de la empresa y de las otras comunidades con las que se relaciona.

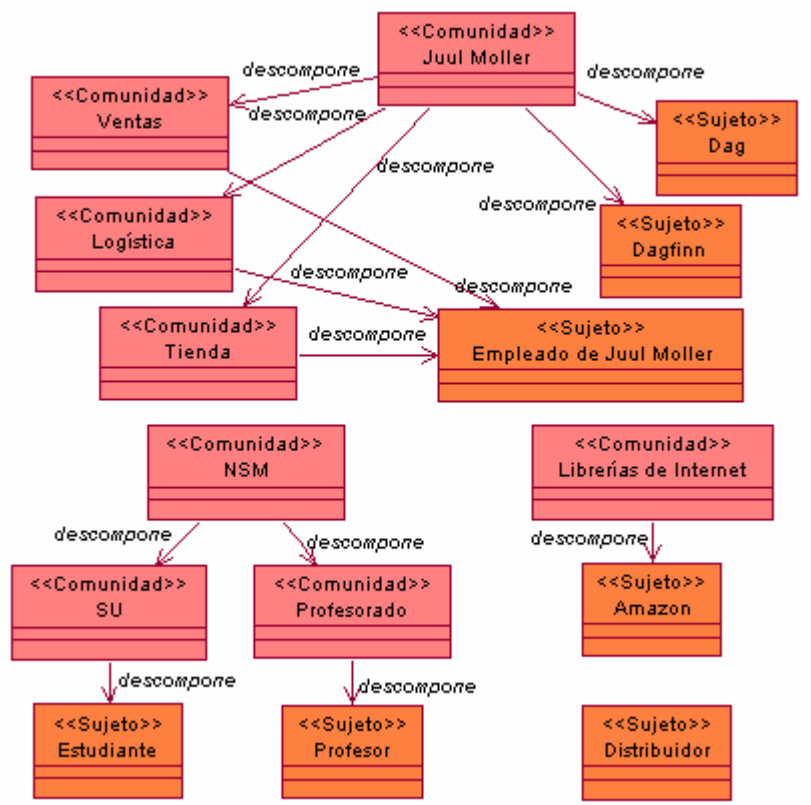


Fig. 106. Actores en el *Juul Møller Bokhandel A/S*.

El segundo punto trata parcialmente sobre la motivación para dar el salto al comercio electrónico y construir el SMA. Para incluir esta información en los requisitos se trabaja con el *aspecto* 1.2. Este *aspecto* contiene *preguntas* sobre los motivos de los actores involucrados en las *actividades*. Con ellas se introducen tanto *objetivos* perseguidos por los *sujetos* a los cuales contribuye directamente el componente (el comercio electrónico o el SMA), como posibles *actividades* en las cuales intervienen esos *sujetos*, *objetivos* y componentes. A continuación se presentan como ejemplos del *aspecto* 1.2 posibles respuestas a las *preguntas* 1.2.1 y 1.2.4.

La *pregunta* 1.2.1 "¿Por qué desea el actor/organización crear el componente? ¿Cuáles son sus objetivos?" puede usarse para establecer asociaciones entre los *sujetos* y sus *objetivos* con la relación *persigue* o bien con el componente a través de *actividades*. La Fig. 107 ilustra la primera de estas interpretaciones. Uno de los *objetivos* de *Dag* es *Preservar la Empresa* de su familia, *Juul Møller*.

Fig. 107. *Pregunta* 1.2.1 sobre objetivos de los actores en *Juul Møller Bokhandel A/S*.

La *pregunta* 1.2.4 "¿Cuáles son los motivos de los actores para emplear el nuevo componente? ¿Qué beneficios obtienen de su uso?" refleja *objetivos* del *sujeto* respecto al componente en el contexto de una *actividad* dada. La respuesta a esta *pregunta* queda reflejada en la Fig. 108. Aquí se muestra parte de la motivación de los *Estudiantes* para usar el *Sistema* de *Juul Møller*. Frente a la *pregunta* 1.2.1 se ha optado por usar un patrón UML-TA donde aparece el *Sistema* a construir. A la hora de elegir los patrones que contendrán la respuesta a una *pregunta* hay que ser cuidadoso para no forzar la inclusión de información por el mero hecho de rellenar el patrón. En este caso quizás hubiera sido más adecuado usar la *pregunta* 1.2.1 o con la 1.2.4 vincular directamente el *sujeto* y su *objetivo* o dar una respuesta parcial donde no se fijara ni la *actividad* ni la *herramienta*. No obstante hay que recordar (ver sección 3.4) que la Fig. 108 permite deducir que el *sujeto persigue* el *objetivo*.

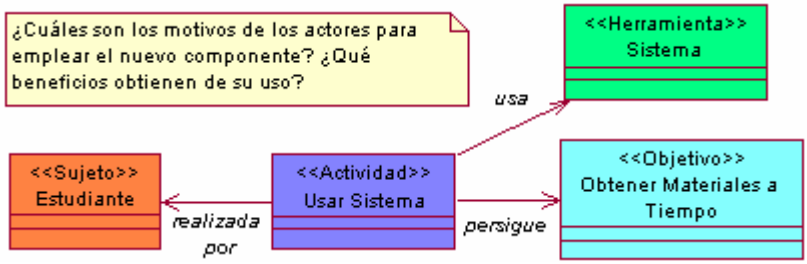


Fig. 108. *Pregunta* 1.2.4 sobre objetivos de usar el componente en *Juul Møller Bokhandel A/S*.

Las *preguntas* del *aspecto* 1.2 muestran dos puntos importantes citados en las secciones 4.3.2 y 4.5 acerca de la GCR. Por un lado, estas *preguntas* tratan de capturar información muy similar sobre los *objetivos* desde diferentes perspectivas, a fin de facilitar al usuario plasmar su conocimiento sobre la situación desde su propio punto de vista. Por otra parte, estas *preguntas* cuentan con varias representaciones en

UML-TA. Por ejemplo, aunque se pregunta por los objetivos en relación con el componente, en esta fase inicial de la captura de requisitos puede ser más claro vincular directamente *objetivos* y *sujetos* que hacerlo a través de *actividades* que usan el componente y persiguen los *objetivos*. Estas diferencias son mostradas en las representaciones gráficas de las preguntas.

Las respuestas a las *preguntas* del *aspecto* 1.2 pueden verse en la Fig. 109.

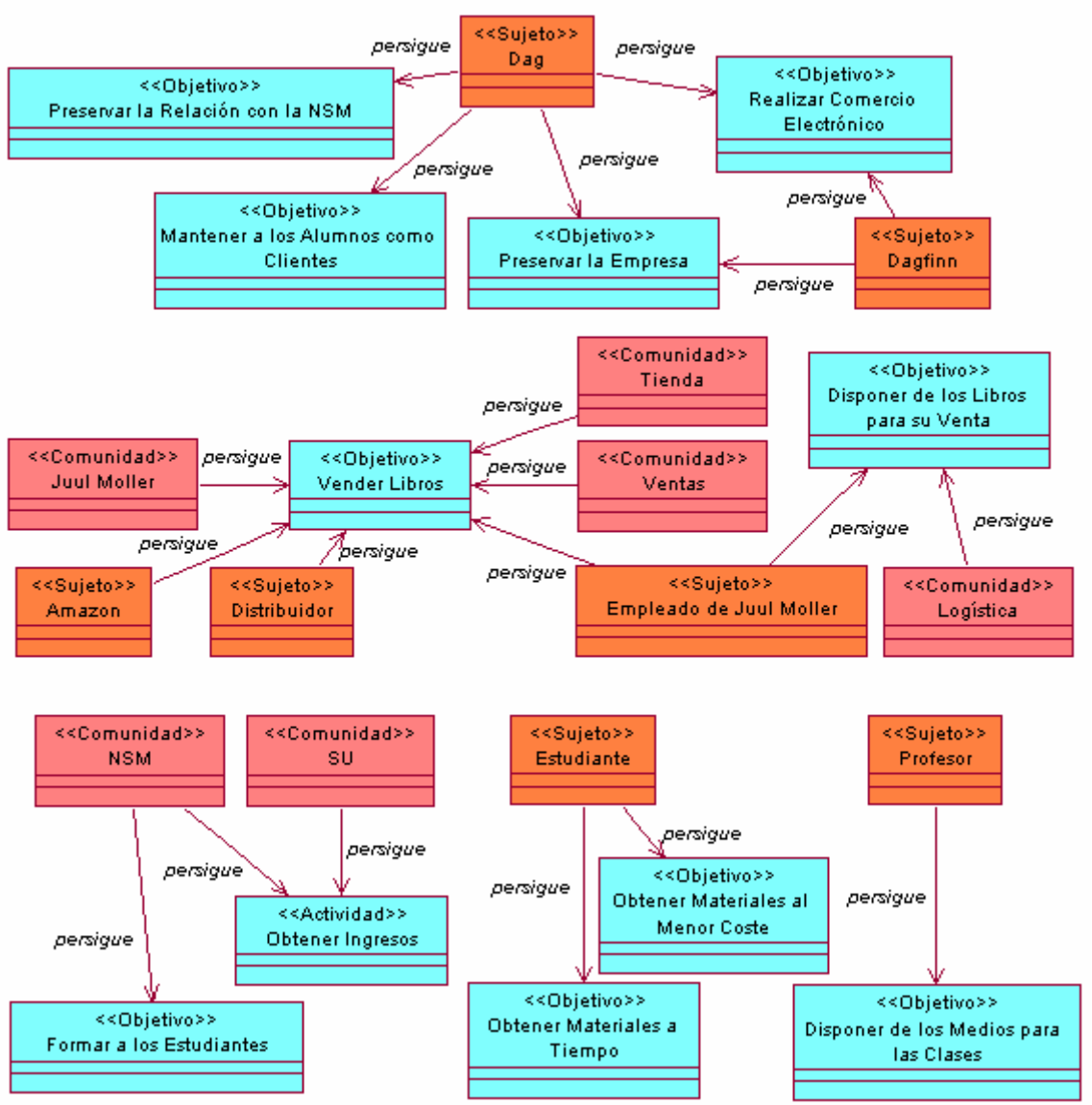


Fig. 109. Objetivos de los actores para *Juul Møller Bokhandel A/S*.

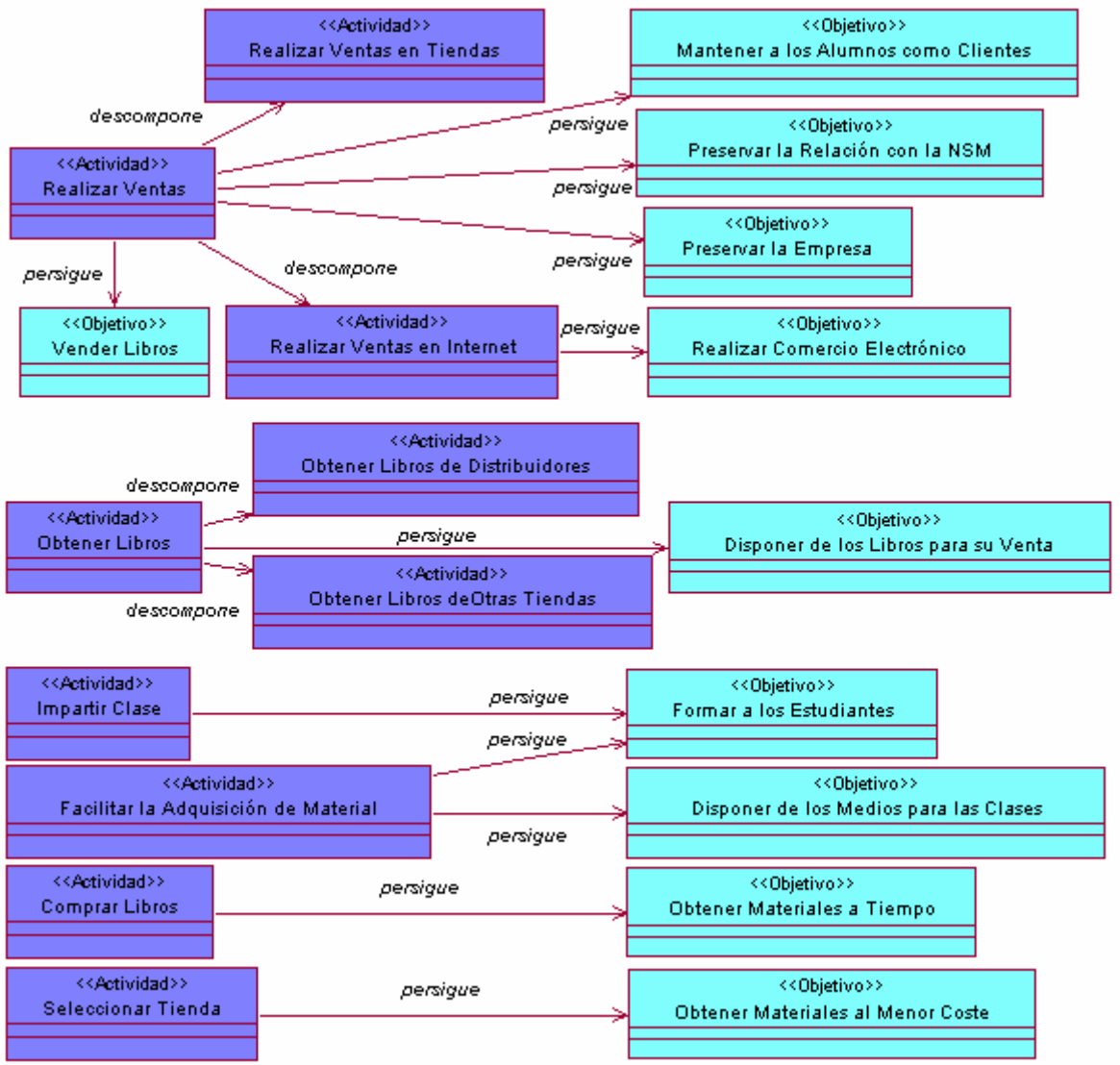


Fig. 110. *Actividades* que operacionalizan los *objetivos*.

El tercer asunto a considerar es el funcionamiento del negocio de la librería. Esta información va a relacionar los *sujetos* anteriores a través de *actividades*. Además, este es el momento en el que empezarán a considerarse elementos del contexto de la *actividad* como *objetos* y *herramientas*. La captura de este conocimiento comenzaría con las *preguntas* del *aspecto* 1.8, sobre las *actividades* que operacionalizan los *objetivos*, y el 1.3, que hacen referencia a las *actividades* relacionadas. Luego se usarían los *aspectos* 2.2, sobre los elementos que se requieren para llevar a cabo las *actividades*, y 2.4, sobre las *herramientas*. En función de las respuestas obtenidas,

también puede ser necesario plantear *preguntas* del *aspecto* 1.12. Éste trata del contexto de los *objetivos*, es decir, qué evidencias pueden indicar que los *objetivos* pueden satisfacerse o no o de qué *objetos* contribuyen a dichos *objetivos*.

Las *preguntas* del *aspecto* 1.8 proporcionan la información recogida en la Fig. 110. Se trata sólo de la traducción de aquellas *actividades* de la especificación original del problema a UML-TA.

Los *aspectos* 1.3, 2.2 y 2.4 no aportan aquí información adicional dada la simplicidad del enunciado inicial. Sabemos por ejemplo que la *actividad Obtener Libros* implica el uso de *herramientas* como el fax, el correo electrónico o el teléfono. Su *producto* puede ser tanto una notificación de aceptación del pedido, como una notificación de imposibilidad de satisfacerlo o el propio pedido. La descripción de estos elementos se abordará más adelante.

La situación de las tecnologías de la información en la empresa, tal y como se ha enunciado, es en buena medida *división del trabajo* dentro de las *actividades*. El *aspecto* relacionado en este caso es el 2.7.

Finalmente, la exposición del problema termina tratando las expectativas de Dag a propósito del comercio electrónico y su empresa. Una vez más se trata de los *objetivos* de los *sujetos* que considera el *aspecto* 1.2. Un resumen de ellos puede verse también en la Fig. 109.

7.3.2. Refinamiento de la motivación de los actores

Uno de los aspectos críticos a considerar tras el análisis inicial son las relaciones entre los motivos de los actores involucrados en el problema. Una pronta detección de las contribuciones, prioridades y conflictos en los objetivos permite establecer negociaciones sobre ellos y juzgar las mejores alternativas de diseño. Por supuesto, este análisis requiere también establecer con más detalles los objetivos esbozados en la sección anterior.

Los *aspectos* 1.2 y 1.7 permiten detallar los *objetivos* perseguidos por los *sujetos*. Nos centramos en el *aspecto* 1.7 acerca del refinamiento de los *objetivos*. Por ejemplo, la *pregunta* 1.7.4 se orienta a obtener los *objetivos* concretos que plasman los *objetivos* de la organización. Con esta *pregunta* se descubre que el *objetivo* de Dag relativo a preservar su empresa no depende directamente de una *actividad*. Realmente se alcanza mediante otros *objetivos* como *Obtener Beneficios* y *Tener Satisfechos a los Clientes*. Repitiendo esta clase de *preguntas* se va creando una jerarquía de *objetivos* y alternativas en su realización. Este proceso se repite hasta alcanzar *objetivos* que puedan ser operacionalizados. Por supuesto, hay que tener en cuenta que algunos de los *objetivos* incluidos son líneas maestras para los actores involucrados, que no tienen una operacionalización en su labor diaria.

El refinamiento de los *objetivos* permite dar una visión más precisa de cuáles están directamente involucrados en las *actividades*. También es preciso prestar atención a cuáles de esos *objetivos* son prioritarios para los usuarios y cuáles responden simplemente a la necesidad de satisfacer los anteriores. Esta sería la información trabajada por el *aspecto* 1.6. Estas *preguntas* reordenan la jerarquía de *objetivos* y añaden quizás refinamientos adicionales. El *objetivo* fundamental de Dag es *Preservar la Empresa* y a él se subordinan el resto de sus *objetivos* en la Fig. 109. No

obstante, como se puede deducir de su actuación en otras situaciones previas (léase en [Espen 2001] su actuación con respecto a su participación en *GC Data*), su único *objetivo* no es preservar el *statu quo*. También trata de *Mejorar su Nivel de Vida*. Puesto que *Juul Møller Bokhandel A/S* es la empresa de su familia, es de esperar que exista un cierto equilibrio entre estos dos *objetivos* esenciales de Dag. No obstante, la relación de *Mejorar su Nivel de Vida* con otros *objetivos* no queda clara a este nivel. Se trata de un factor subjetivo de Dag que influye de una forma indeterminada sobre los demás *objetivos*.

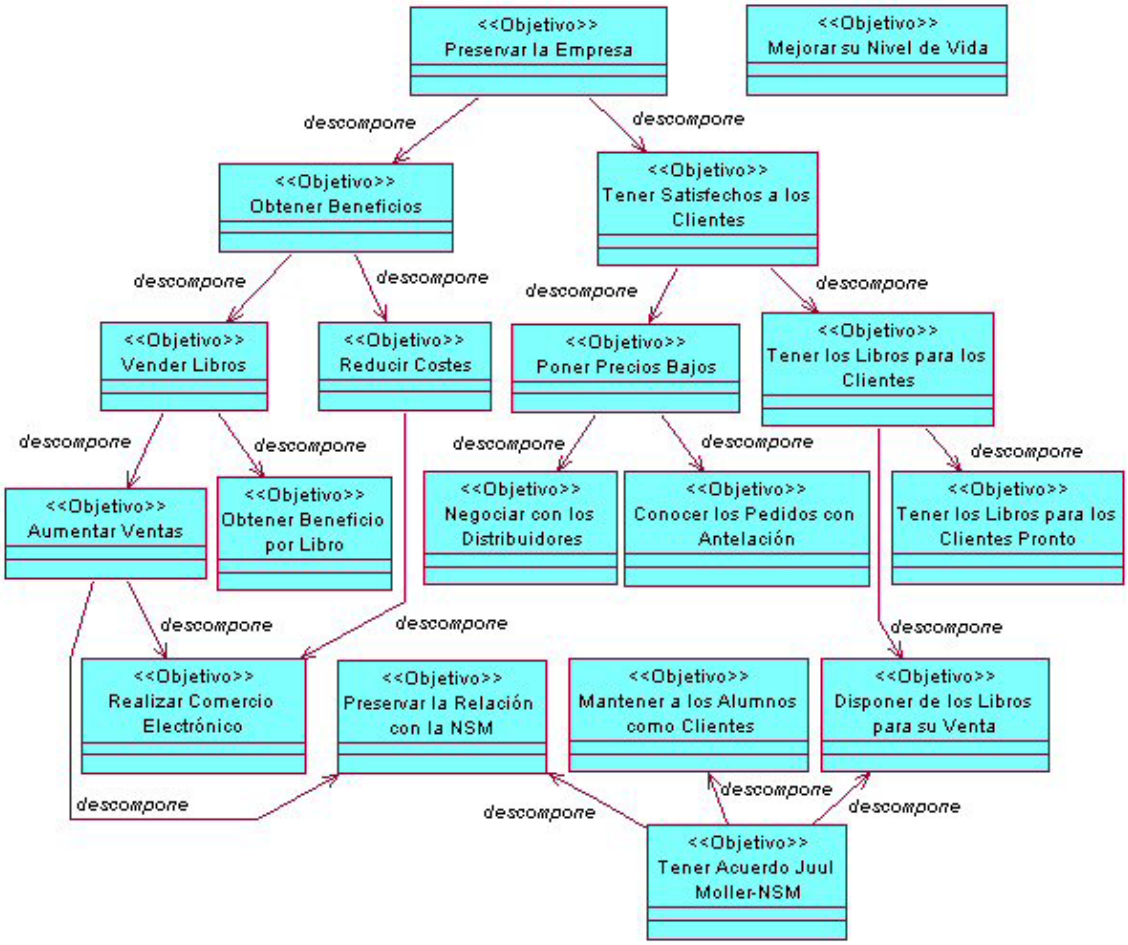


Fig. 111. Refinamiento de los *objetivos* de *Juul Møller Bokhandel A/S*.

La Fig. 111 muestra el resultado del refinamiento y reordenación de los objetivos de Dag y *Juul Møller*. Un aspecto de interés en la Fig. 111 es el *objetivo Tener Acuerdo Juul Møller–NSM*. Éste no es un objetivo exclusivo de Dag y su empresa, sino que es compartido con otras organizaciones del problema tales como la NSM, la SU y el profesorado, aunque cada uno de ellos busca que dicho acuerdo contribuya a

sus propios intereses. Ello hace que aunque se trate de un *objetivo* compartido por los distintos actores, vaya a ser un foco de conflictos como se verá más adelante al analizar las contribuciones entre los *objetivos*.

Una vez que se ha desarrollado suficientemente la jerarquía de los *objetivos* de los distintos actores y organizaciones se deben establecer los posibles conflictos entre ellos y sus soluciones. Para esto se cuenta con los *aspectos* 1.2, 1.13, 1.14, 1.15 y 1.16.

Como ejemplo de la detección de esta clase de conflictos entre *objetivos* se incluye la *pregunta* 1.2.3. En ocasiones el conflicto entre *objetivos* surge de la existencia de ciertos *artefactos* del entorno que pueden resultar perjudiciales para algunos *objetivos* de grupos concretos. Esta es la clase de información que intenta obtener la *pregunta* 1.2.3 "¿Existen inconvenientes para la organización o partes de ella en la construcción e implantación del componente?". La Fig. 112 da una de las posibles respuestas a la *pregunta*. Se puede ver que aunque varios actores persigan el mismo *objetivo Vender Libros*, el hecho de que uno lo satisfaga es perjudicial para los otros. Para el negocio de *Juul Møller* no es indiferente quién satisface las necesidades de los miembros de la *comunidad* NSM.

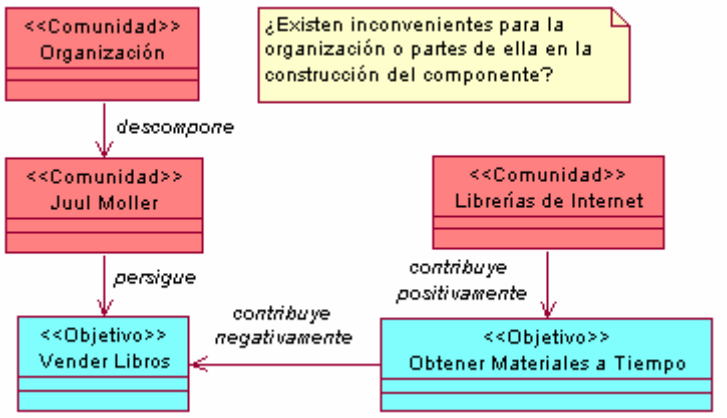


Fig. 112. *Pregunta* 1.2.3 en *Juul Møller Bokhandel A/S*.

Parte de las relaciones de contribución entre los *objetivos* se sintetizan en la Fig. 113. Como se indicó anteriormente, el *objetivo Tener Acuerdo Juul Møller–NSM* trata de satisfacer *objetivos* diversos para distintos actores, muchas veces contradictorios. En primer lugar, existen conflictos entre obtener ingresos para todos los implicados y el precio de los libros. Para *Juul Møller* se trata de una forma de incrementar las ventas de libros y por tanto de aumentar sus beneficios. Sin embargo ese acuerdo le fuerza a proporcionar ingresos tanto a la NSM (mediante el alquiler de las tiendas en sus campus) como a la SU (mediante un porcentaje de sus ingresos por venta de libros). A su vez, todos los miembros de la NSM buscan en el acuerdo rebajar el precio de los libros para los estudiantes. La segunda fuente de conflictos está en los profesores. Para estos elaborar el catálogo del curso supone la ventaja de que sus alumnos puedan *Obtener Materiales a Tiempo* y ellos mismos *Disponer de los Medios para las Clases*. Entregar el catálogo tiene la desventaja de que perjudica a

Tener Flexibilidad en los Medios del Curso. El tercer foco de conflicto en la Fig. 113 aparece con los alumnos. Los intereses de estos son *Obtener los Materiales a Tiempo* y *Obtener los Materiales al Menor Coste*. Ambos *objetivos* son favorecidos por *Comprar en la Mejor Tienda*. Este fue el caso que vivió Dagfinn cuando sus compañeros de curso eligieron usar los servicios de librerías en Internet. Aunque esta libertad de selección beneficia a los alumnos a nivel individual, les perjudica a nivel colectivo. La SU deja de *Obtener Ingresos* por su porcentaje en las ventas de *Juul Møller*. Además, el propio *Juul Møller* ve reducidas sus ventas, lo que le puede llevar a renunciar a mantener su acuerdo con la NSM.

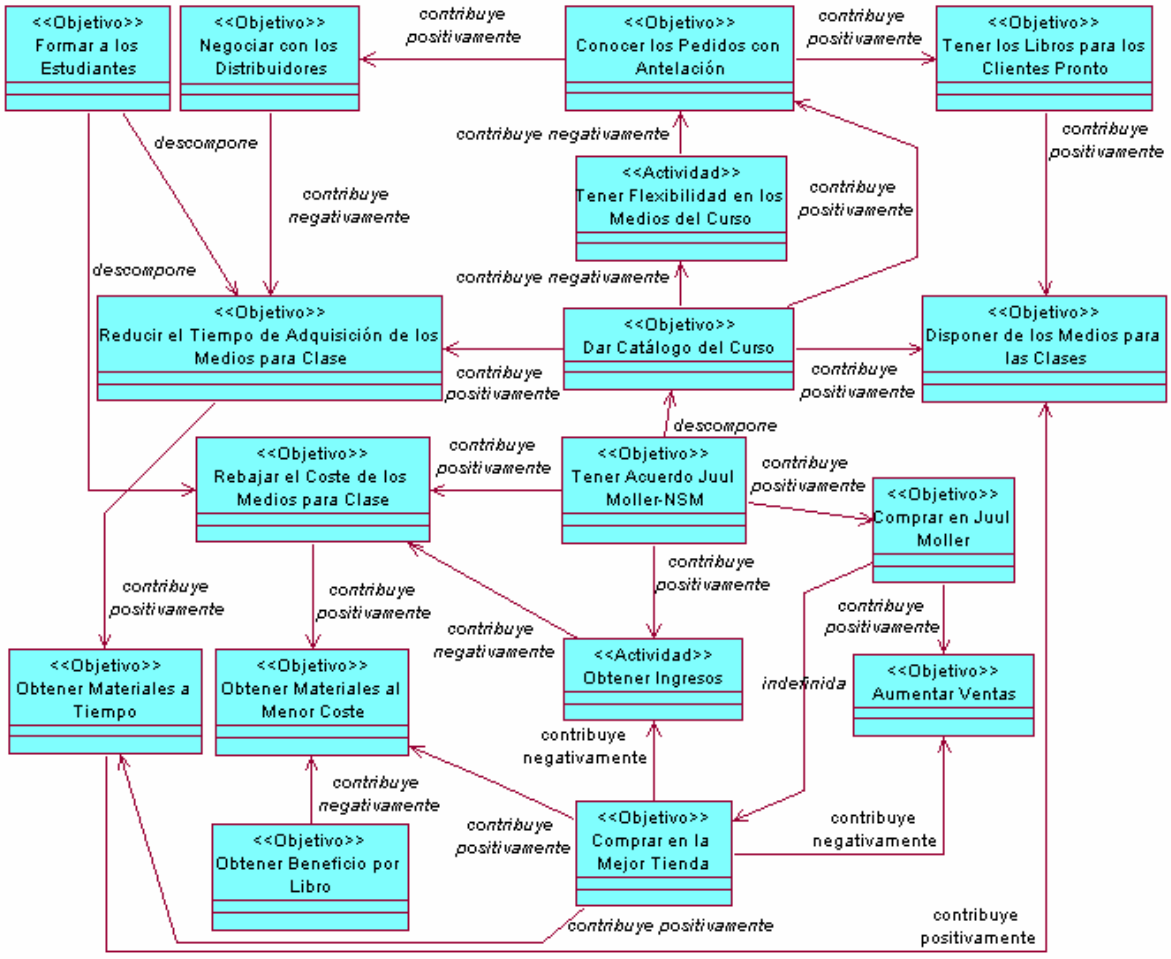


Fig. 113. Relaciones de contribución entre los *objetivos*.

La Fig. 113 sólo muestra la situación actual de las relaciones entre los *objetivos*. Por supuesto, los conflictos pueden ser resueltos mediante, por ejemplo, la actuación de nuevos *sujetos*, e.g. sustituyendo a *Juul Møller*, nuevas *actividades*, e.g. mejora de

los mecanismos de distribución, o el uso de nuevas herramientas, e.g., publicaciones en formato electrónico.

En cuanto a la resolución de conflictos, una muestra de este problema es la *pregunta* 1.16.2 "En caso de conflictos, ¿puede establecer prioridades entre los objetivos?". Existe un conflicto, no recogido en la Fig. 113, entre el *objetivo* de *Obtener Beneficio por Libro* y *Tener los Libros para los Clientes Pronto*. Ambos objetivos aparecen en la Fig. 111. En [Espen 2001] se explica como suministrar el libro al cliente puede llevar a perder dinero en su venta, dependiendo de las vicisitudes de la distribución. Esta situación no ocurre siempre por lo que aunque se recoja en la Fig. 114, ha de entenderse matizada por las circunstancias. Al responder a esta *pregunta* no se han podido establecer cuáles son las evidencias que permiten establecer la orientación de la relación *supera*. Es posible que al avanzar el estudio el usuario pueda responder a *preguntas* como la 1.16.3 que establecen esos criterios.

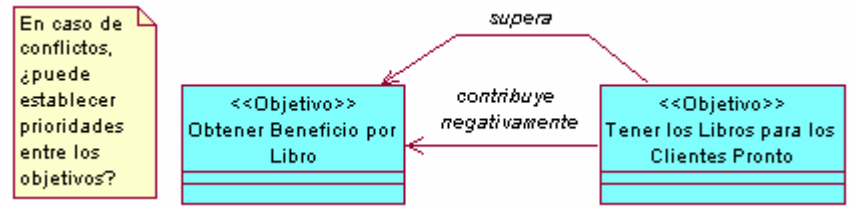


Fig. 114. *Pregunta* 1.16.2 sobre conflictos entre dar servicio al cliente y obtener beneficios.

Una vez que se han trabajado las *preguntas* de estos *aspectos* relacionados con los *objetivos*, se tiene una visión amplia de la motivación de los actores y de las relaciones entre sus *objetivos*. Este conocimiento es usado en el diseño del SMA de la solución para optar por aquellas decisiones de diseño que minimicen los conflictos en el *sistema de actividad* global.

7.3.3. Requisitos del SMA para la librería en Internet

Las dos secciones anteriores en la captura de requisitos han usado la GCR para establecer el contexto en el cual se insertará el SMA para *Juul Møller*. Esta sección tratará de obtener los requisitos globales para el sistema que se está diseñando y relacionarlos con la información obtenida anteriormente.

Puesto que los actores que se relacionan con el SMA están fijados por las etapas anteriores, este análisis comienza tratando de capturar los beneficios que esperan obtener del sistema. Se recurre al *aspecto* 1.2 del *área* de *Medios/fines* que contiene *preguntas* relativas a las ventajas que esperan obtener los actores del uso del sistema. Hasta este momento sólo se sabía que *Juul Møller* esperaba *Aumentar Ventas* y *Reducir Costes* al *Realizar Comercio Electrónico* como se ve en la Fig. 111. Al examinar las expectativas de los participantes se obtienen dos grupos de respuestas relacionadas pero que han de considerarse por separado. Unas hacen referencia al SMA de *Juul Møller* mientras que otras se refieren al comercio electrónico en general, donde pueden estar incluidos los competidores de *Juul Møller*. El resultado de estas primeras preguntas se ha desglosado para comentarlo en las figuras Fig. 115,

Fig. 116 y Fig. 117. Estas figuras muestran la primera captura de requisitos del SMA. Al igual que se hizo en secciones anteriores, las relaciones entre los *objetivos* se analizarán posteriormente.

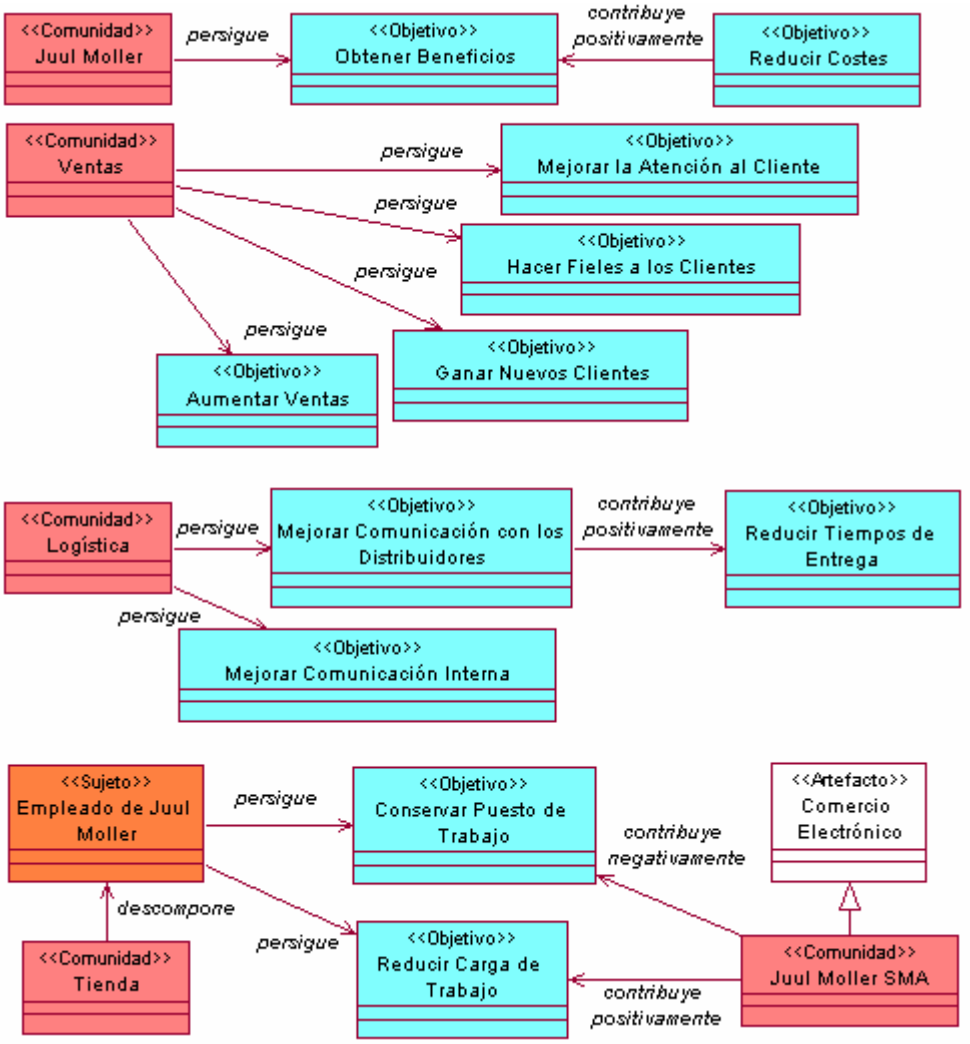


Fig. 115. Expectativas de los actores de *Juul Møller* acerca del comercio electrónico y el nuevo SMA.

La Fig. 115 se refiere a los *objetivos* de las *comunidades* y *sujetos* dentro de *Juul Møller Bokhandel A/S*. Como empresa, *Juul Møller* trata de *Obtener Beneficios*. Este objetivo global se traduce de diferentes formas para sus departamentos. Para el departamento de ventas se trata de vender muchos ejemplares y una forma de lograrlo es incrementar y preservar sus clientes. El departamento de logística trata de reducir el tiempo que necesita para proporcionar sus libros. Finalmente, los empleados individuales están preocupados porque el nuevo sistema no implique una mayor carga

de trabajo ni haga peligrar sus empleos. En este caso puede apreciarse que los objetivos de la Fig. 115 son concreciones de los mostrados anteriormente en las Fig. 111 y Fig. 113. Por ejemplo, *Negociar con los Distribuidores* en la Fig. 111 se plasma en la Fig. 115 en *Mejorar la Comunicación con los Distribuidores* para *Reducir Tiempos de Entrega*. Del mismo modo, *Aumentar Ventas* de la Fig. 113 se refina en la Fig. 115 en varias formas de lograr este incremento, tales como *Ganar Nuevos Clientes* o *Mejorar la Atención al Cliente*.

El segundo actor importante en relación con el sistema es la NSM. La NSM, como organización, sus alumnos y profesores tienen sus propios problemas que desearían ver tratados por la el SMA. Es decir, esperan que el SMA les reporte ciertos beneficios que son modelados como *objetivos*. Por supuesto, estos son diferentes de los de la empresa *Juul Møller*. La Fig. 116 muestra estos objetivos.

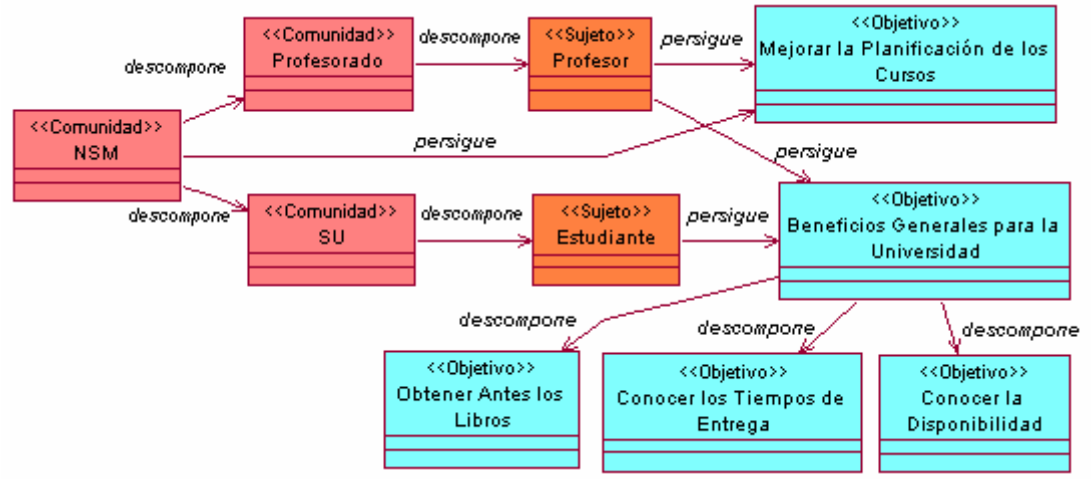


Fig. 116. Expectativas de los actores de la NSM acerca del comercio electrónico y el nuevo SMA.

La NSM espera globalmente mejorar la planificación de los cursos. Disponer con anticipación del catálogo del curso y saber con mayor precisión el tiempo que tardará en disponerse de los libros y poderlos tener más rápido permitirá que los cursos dispongan a tiempo de su material y puedan cumplir su planificación. Para la NSM respetar estos plazos tiene sobre todo una influencia administrativa.

Los profesores y alumnos también persiguen el beneficio de *Mejorar la Planificación de los Cursos*. Además tratan de obtener unos *Beneficios Generales para la Universidad*. Se trata de ventajas tangibles para sus clases como *Conocer los Tiempo de Entrega* u *Obtener Antes los Libros*.

En general se observa que los objetivos de los actores que forman parte de la NSM respecto al futuro sistema están directamente relacionados con los que aparecieron anteriormente en las Fig. 110 y Fig. 113. Es decir, son sus objetivos previos reformulados para que los satisfaga el SMA. Por ejemplo, los profesores expresaban en la Fig. 113 su deseo de *Disponer de los Medios para las Clases*; en la Fig. 116 esperan que ese objetivo se logre porque el SMA permita *Obtener Antes los Libros*.

El último grupo de actores considerados en este apartado son los distribuidores que trabajan con *Juul Møller* y sus competidores en la red. Los distribuidores se encuentran entre los *sujetos* que deberán relacionarse con el SMA. Posiblemente no lo hagan de forma directa sino a través del personal de *Juul Møller*. En todo caso, su participación en las interacciones con el sistema ha de ser contemplada. El otro actor en este grupo son los competidores de *Juul Møller*. Estas librerías son las que pueden arrebatar a la librería noruega sus clientes. Su análisis permitirá saber la clase de servicios que ofrecen y cómo interactúan con distribuidores y clientes. La Fig. 117 muestra sus objetivos.

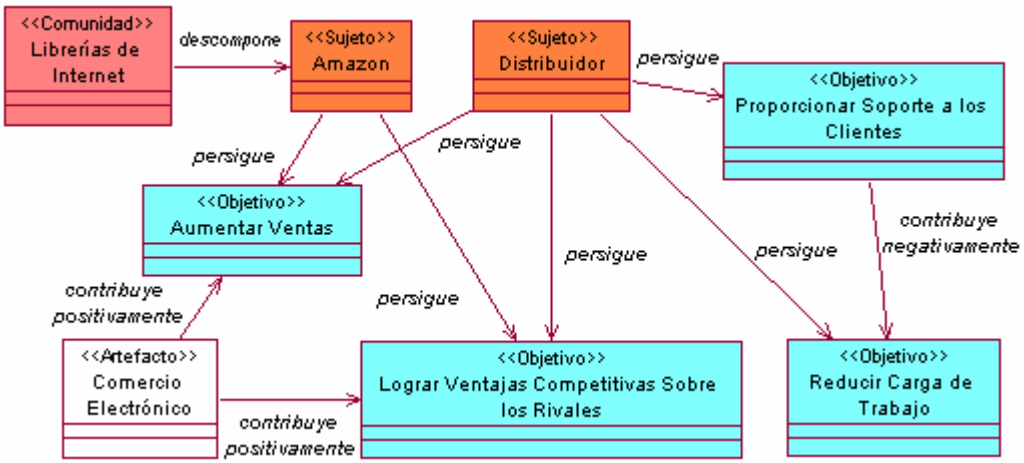


Fig. 117. Expectativas de otros actores acerca del comercio electrónico.

Las librerías en Internet tienen unos objetivos generales muy similares a los de la propia *Juul Møller*: *Aumentar Ventas* y *Lograr Ventajas Competitivas Sobre los Rivales*. Estas ventajas competitivas son relativas a la reducción de costes y tiempos, al igual que las que perseguía *Juul Møller*.

Los distribuidores pasan a ser colaboradores en el proceso de negocio que quieren *Proporcionar Soporte a los Clientes*. Sin embargo, habrá que vencer sus reticencias a realizar más trabajo. *Juul Møller* es una librería de cierta importancia pero no la suficiente como para forzar a los distribuidores a usar su sistema, especialmente si éste no les aporta beneficios. Este aspecto deberá ser tenido en cuenta en el posterior diseño del SMA, por ejemplo siendo *Juul Møller* quien implemente las interfaces para conectarse a los distribuidores y minimizando los cambios en los pedidos.

El refinamiento y la puesta en relación de estos objetivos para el SMA se basan, como ocurrió al analizar el contexto, en las *áreas* de *Medios/fines* y *Entorno*. Como este paso ya se ha visto anteriormente, ahora se describirá la jerarquía resultante de objetivos en relación con los vistos en el contexto de las secciones 7.3.1 y 7.3.2. El resultado se ha descompuesto una vez más en varias figuras para facilitar su explicación. Se trata de las figuras comprendidas entre la Fig. 118 y la Fig. 122.

La Fig. 118 muestra la jerarquía de objetivos de la empresa *Juul Møller* para el SMA. La Fig. 115 ya mostró los objetivos genéricos de la empresa respecto al sistema. Dichos objetivos aparecían en formulaciones muy genéricas que no estaban

directamente relacionadas con el SMA. Era preciso determinar objetivos más próximos al SMA y la jerarquía de esos objetivos.

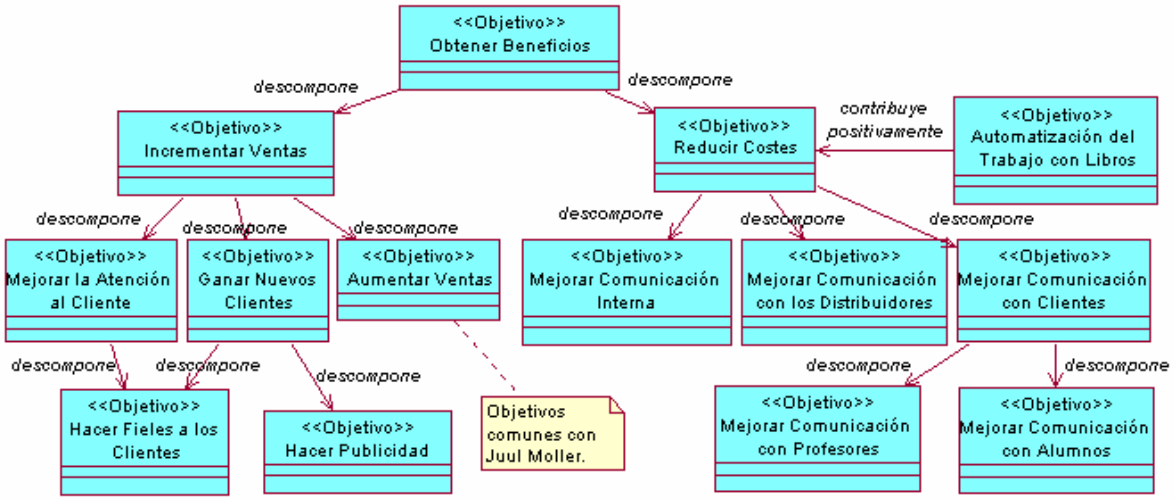


Fig. 118. Objetivos de *Juul Møller* para el SMA relacionados con los del entorno.

Así nos encontrábamos con que el objetivo de *Juul Møller* para el SMA era simplemente *Obtener Beneficios* para lo cual resultaba positivo *Reducir Costes*. Por supuesto, existen más formas de aumentar los beneficios usando el SMA, tales como hacer publicidad o ganar nuevos clientes. Esto hizo pensar en la existencia de más alternativas para satisfacer el *objetivo Obtener Beneficios*. Entonces se pensó en aplicar el *aspecto* 1.7 sobre *Refinamiento de Objetivos* y en él las *preguntas* 1.7.5 y 1.7.6, relativas a formas alternativas de satisfacer los *objetivos*. Las respuestas permitieron detectar que faltaba *Incrementar Ventas* como manera de *Obtener Beneficios*.

Las *preguntas* del *aspecto* 1.3, sobre las acciones del nuevo componente, llevaron de forma indirecta a plantear también nuevos objetivos. Se había incidido en la necesidad de mejorar la comunicación con los distribuidores y dentro de la propia empresa. Por otra parte se vio que había que *Mejorar la Atención al Cliente*, entendida como informarle acerca de sus pedidos. Sin embargo, al ser interrogado sobre otras acciones que debía soportar el sistema, el cliente advirtió que los usuarios también podían dar información a *Juul Møller* indicándole sus pedidos: el material que necesitaban, en que fechas y precios orientativos en caso de que tuvieran un límite. Toda esta información agilizaba los trámites con los distribuidores, reduciendo costes y tiempo de provisión. De este modo se introdujo un nuevo *objetivo Mejorar Comunicación con Clientes*.

Por último, el cliente introdujo dos objetivos que apuntan algunos deseos acerca del sistema pero que no quedan concretados. Uno es la *Automatización del Trabajo con Libros*. Este objetivo pretende capturar la idea de que el SMA reducirá el esfuerzo para llevar a cabo las tareas rutinarias. Ello permitirá además que los trabajadores se dediquen a tareas más próximas al trato personal con clientes y distribuidores. Los otros son *Hacer Fieles a los Clientes* y *Hacer Publicidad*. En ambos casos *Juul*

Møller quiere sacar provecho del conocimiento que tiene de su negocio para ofrecer servicios personalizados a sus clientes. De este modo planea competir con las grandes librerías de la red mediante un trato cercano e individualizado.

El siguiente paso en el análisis de estos objetivos es asociarlos mediante relaciones de contribución. Aquí se han usado las *preguntas* de los *aspectos* 1.7, 1.11, 1.12 y 1.13. Estas *preguntas* hacen referencia a los elementos que ayudan o determinan el éxito o fracaso de los objetivos. La Fig. 119 muestra el resultado de estas *preguntas*. Se han remarcado con notas algunos de los *objetivos* ya contemplados previamente para algunos actores. Parte de la información en lo que se refiera a estos objetivos anteriores procede de la Fig. 113.

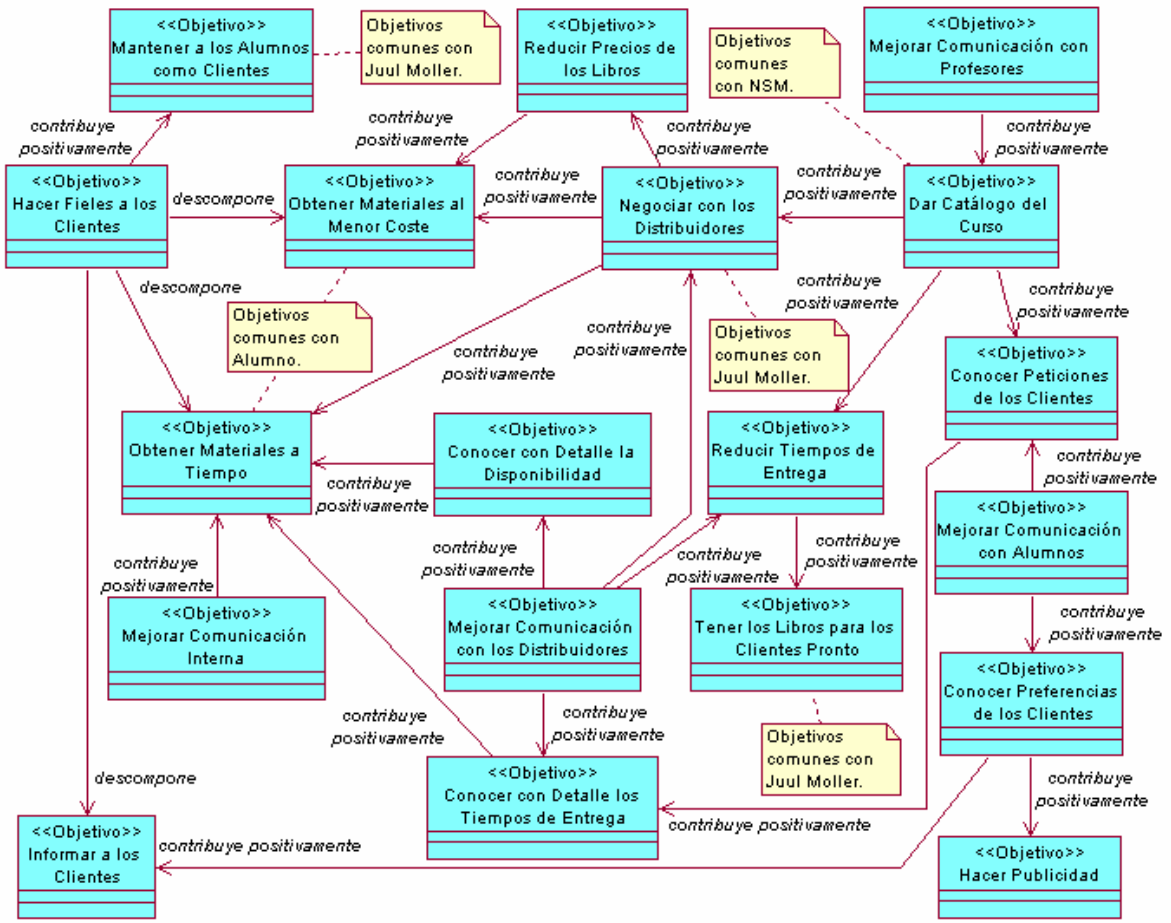


Fig. 119. Objetivos de *Juul Møller* para el SMA con relaciones de contribución.

A modo de ejemplo de los refinamientos que han conducido a la Fig. 119 se describen el uso de las *preguntas* 1.7.6, 1.10.1 y 1.12.4.

La *pregunta* 1.8.6 es "¿Existen objetivos que ayuden a satisfacer estos aunque no garanticen dicha satisfacción?" ayudó establecer relaciones entre varios objetivos que

habían sido introducidos previamente y que estaban inconexos. Así, ayudó a poner de manifiesto el motivo de introducir el objetivo de *Mejorar la Comunicación con los Distribuidores*. En primer lugar conectaba con uno los *objetivos* previos, el relativo a *Negociar con los Distribuidores* que apareció en la Fig. 111. Se espera que el sistema proporcione de forma sencilla datos relativos a pedidos estimados, disponibilidad en los distribuidores y plazos de entrega. Esta información es la base sobre la que *Juul Møller* negoció acuerdos en el pasado. Además debe facilitar su intercambio con los sistemas informáticos de los distribuidores a fin de no incrementar su carga de trabajo, lo cual seguramente estos rechazarían. También con esta *pregunta* se estableció la relación entre *Dar Catálogo del Curso* y *Reducir los Tiempos de Entrega*. Conocer los pedidos que se iban a realizar permitía iniciar antes las gestiones con los distribuidores y planificar los envíos.

Otra *pregunta* relevante en esta fase fue la 1.10.1 "¿Cuándo consideraría que el componente ha sido bien construido? ¿Cuáles serían los criterios para comprobarlo?". Esta *pregunta* se planteó discutiendo el *objetivo* de *Mejorar la Comunicación con los Distribuidores*. Uno de los criterios que surgió para considerar que existía una mejora era el *objetivo Conocer con Detalle la Disponibilidad*. Como se apuntó a propósito de la *pregunta* 1.7.6, el SMA que se diseñase debería comunicarse con los sistemas informáticos de los proveedores. En este caso, no obstante, el motivo apuntado fue conocer el *stock* de material en el distribuidor.

La última *pregunta* cuya aplicación recogemos en esta fase es la 1.12.4 "¿Qué elementos piensa que contribuyen *a priori* a que el objetivo se satisfaga?". Esta *pregunta* se formuló a propósito de uno de los *objetivos* introducidos en la Fig. 118, *Hacer Publicidad*. A fin de diferenciarse de otras librerías, *Juul Møller* pretendía reaprovechar sus conocimientos sobre sus clientes. Se determinó que la obtención de este conocimiento se representaría mediante el objetivo *Conocer las Preferencias de los Clientes*. *Conocer las Preferencias de los Clientes* es un factor previo que ayuda al éxito de *Hacer Publicidad*. Al introducir el nuevo *objetivo Conocer las Preferencias de los Clientes* se planteó la *pregunta* 1.7.7 acerca de si el nuevo *objetivo* ayudaba a satisfacer otros. La respuesta a esta *pregunta* hizo introducir el *objetivo Informar a los Clientes*, al cual contribuye positivamente *Conocer las Preferencias de los Clientes*. Luego se relacionó *Informar a los Clientes* con *Hacer Fieles a los Clientes*.

Una vez concluido el análisis de los objetivos de *Juul Møller* como empresa, se prosiguió con el resto de los actores involucrados en el problema, comenzando por los trabajadores de la empresa.

Los objetivos de los trabajadores respecto al SMA pueden verse en la Fig. 120. Viendo la Fig. 115, se recordará que los objetivos primordiales de los trabajadores eran *Conservar Puesto de Trabajo* y *Reducir Carga de Trabajo*. Los trabajadores temían que la automatización que trajera el SMA pudiera significar que menos de ellos serán necesarios. Al relacionar sus objetivos con los de la empresa y el propio SMA, se observó que *Obtener Beneficios* y *Tener Satisfechos a los Clientes* era visto como positivo para sus puestos de trabajo. La empresa no tendría que reducir plantilla, los trabajadores podrían mejorar sus condiciones económicas e incluso era posible que se contratase más personal si el negocio crecía. Acerca de la carga de trabajo, se recalcó que la automatización sólo podría suplir al personal en las partes más rutinarias de su trabajo. Por el contrario, al liberarles de esa clase de trabajo seguramente la empresa les requeriría que trabajasen más en contacto directo con

clientes y distribuidores. Aunque no aparece en la Fig. 120, éste podría ser el origen de un conflicto para aquellos trabajadores que se sintieran más cómodos realizando el trabajo que automatizaría el SMA.

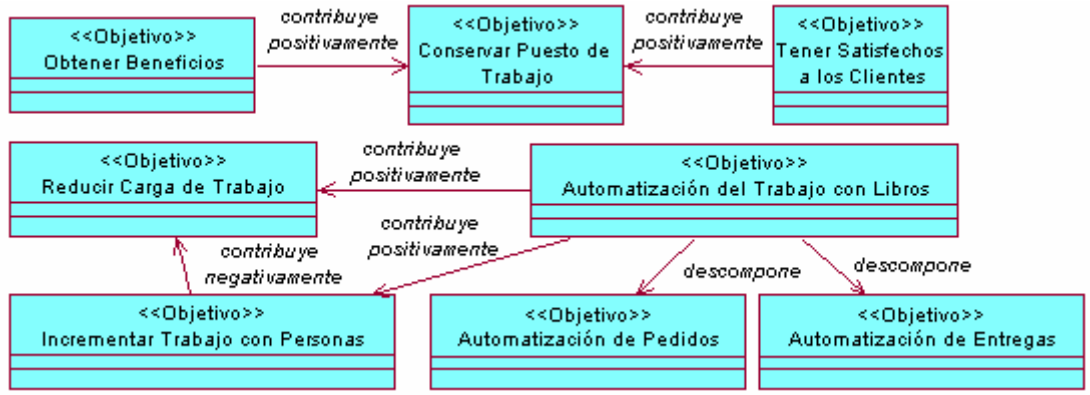


Fig. 120. Objetivos de los trabajadores de *Juul Møller* respecto al SMA.

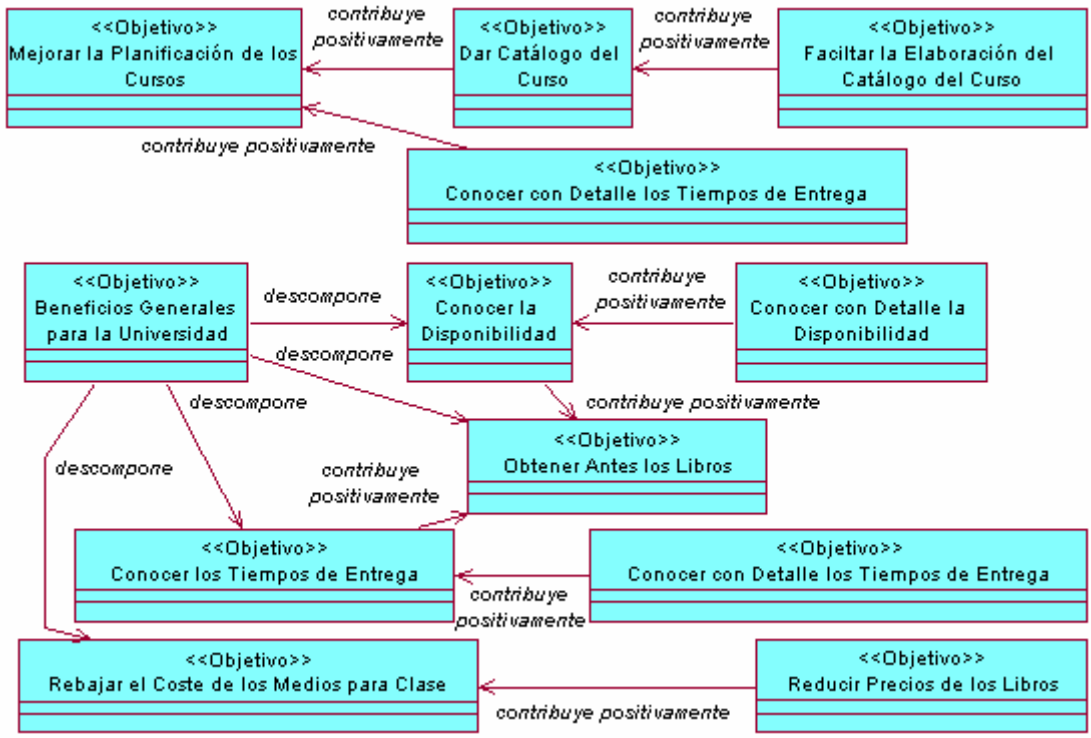


Fig. 121. Objetivos de la NSM respecto al SMA.

El siguiente grupo de actores que se consideran es el de los que forman parte de la NSM. Sus *objetivos* pueden verse en la Fig. 121.

Globalmente, el interés de la NSM es *Mejorar la Planificación de los Cursos*. Este objetivo significa que los materiales para los cursos estén a tiempo. Para lograrlo, la escuela ha de proporcionar a tiempo el catálogo y conocer tan pronto como sea posible cuánto tardarán esas entregas. A fin de mejorar la predisposición de los profesores a entregar el catálogo y minimizar el tiempo que tardan en elaborarlo, se crea un nuevo *objetivo* para el SMA, *Facilitar la Elaboración del Catálogo del Curso*. Lógicamente, lograr que los materiales estén a tiempo también depende de *Juul Møller* que ha de hacer los pedidos. Esa parte de los objetivos ha sido tratada anteriormente en esta sección.

El resto de los objetivos de la NSM corresponden a profesores y alumnos. Estos objetivos se centran en reducir el precio de los libros para clase y el tiempo necesario para obtenerlos. Muchos de esos objetivos se relacionan con estrategias que ya han sido consideradas por *Juul Møller* para hacer crecer su negocio. Así surgen algunas de las relaciones entre *objetivos* capturadas en la Fig. 121, como la que se ve entre *Reducir Precios de los Libros* y *Rebajar el Coste de los Medios para Clase* o *Conocer con Detalle la Disponibilidad* y *Conocer la Disponibilidad*.

Por último se contemplan los objetivos de los distribuidores y de otras librerías en Internet en la Fig. 122. Las preocupaciones de los distribuidores respecto al SMA de *Juul Møller* y en general al comercio electrónico, es que sea una herramienta para reducir su carga de trabajo y para mejorar el servicio a sus clientes. No obstante, se teme que estos nuevos servicios a sus clientes se traduzcan en más trabajo para sus empleados. Por otra parte, las librerías en Internet ven el comercio electrónico como una forma de alcanzar a una gran masa de potenciales clientes, satisfaciendo así *Aumentar Ventas*, y por otra parte de lograr ventajas competitivas sobre sus rivales, al abaratar costes y poder invertir entonces en otros aspectos del negocio.

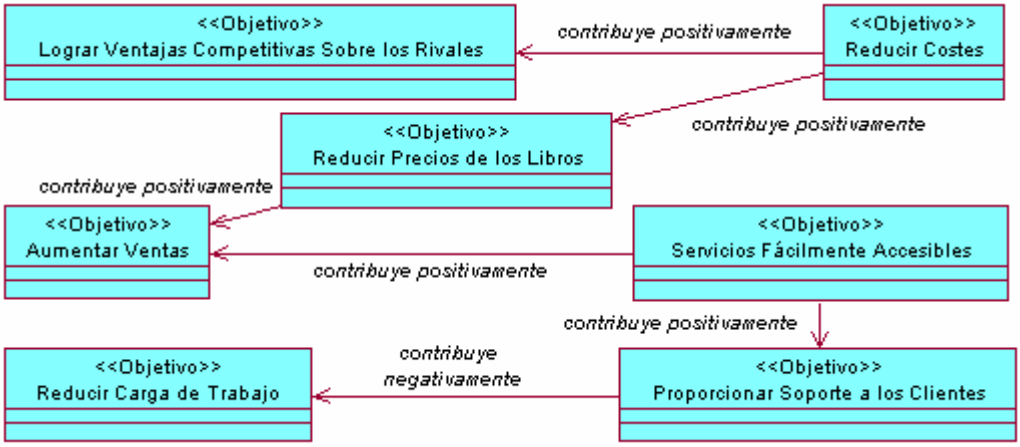


Fig. 122. Objetivos de los distribuidores respecto al SMA y de distribuidores y librerías en Internet respecto al comercio electrónico.

Esta sección ha mostrado la forma de profundizar en los objetivos iniciales de los actores para irlos aproximando hacia los requisitos del sistema. Las formulaciones de objetivos genéricos se han ido centrando en las posibilidades que debería satisfacer el sistema que se ha de construir. Estos objetivos serán la base para establecer la arquitectura del SMA en el apartado siguiente. A medida que se vayan introduciendo los componentes de la especificación del sistema, tales como agentes, actividades o recursos, estos objetivos se irán refinando y acercándose a una visión más operacional.

7.3.4. Arquitectura del SMA

En esta sección se empezará a desarrollar la arquitectura para el SMA en términos de sus agentes y de los conceptos asociados a estos. El modelado se realizará en términos de las abstracciones de la TA e irá surgiendo mediante el uso de la GCR. El resultado para una metodología de SMAs se ilustrará con la traducción de algunos diagramas resultantes a INGENIAS. Como se mostró en el apartado 3.5, las abstracciones de la TA pueden ser traducidas automáticamente a una metodología de agentes si se cuenta con las correspondencias adecuadas.

Dado el actual estado de las especificaciones, un buen punto de partida es tratar de identificar algunos de los actores del SMA asociados con los objetivos. A partir de los actores y sus objetivos se pueden asignar responsabilidades para operacionalizar dichos objetivos.

A la hora de identificar los sujetos asociados a los objetivos puede hacerse uso de *preguntas* como la 1.1.5 “Dados los objetivos, ¿qué actor debería hacerse cargo de ellos?” para asignar responsabilidades de una forma simple. Aquí es importante mostrar al cliente que debe pensar en el SMA como una extensión de su organización donde los componentes del sistema son concebidos como empleados que también desempeñan tareas y tienen misiones. La primera aproximación adoptada para repartir las responsabilidades consiste en asignar un agente representante a cada uno de los actores involucrados en la línea de las propuestos en [Maes 1998]. Se introducen entonces agentes representantes para los miembros de los departamentos de *Juul Møller*, diferenciando entre logística y ventas. Del mismo modo hay representantes para alumnos, profesores y distribuidores. Al asignar los objetivos a los agentes, se observa que existen responsabilidades comunes a varios de ellos. Por ejemplo, *Mejorar la Comunicación Interna* es un *objetivo* que tienen tanto el agente *Representante Juul Ventas* como el *Representante Juul Lógística*. A fin de facilitar la representación de estos comportamientos comunes y evitar duplicidades, se crean agentes básicos para los agentes de *Juul Møller*, i.e. *Representante Juul*, y para los de los clientes, i.e. *Representante Cliente*. Ahora, respondiendo a la *pregunta* 1.1.5, las responsabilidades del sistema quedan asignadas a los agentes. La Fig. 123 muestra parcialmente esta asignación, centrándose en los agentes para los clientes.

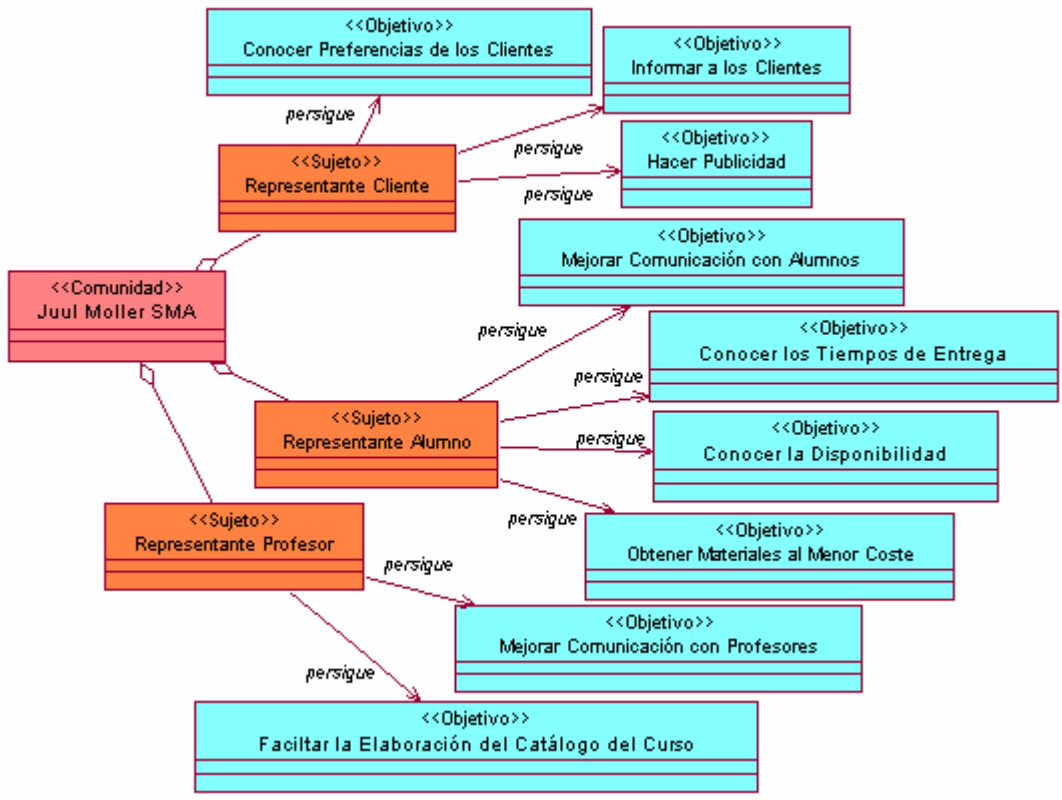


Fig. 123. Actores de soporte a los clientes en el SMA de *Juul Møller*.

Hasta este momento, sólo se ha abordado muy brevemente la operacionalización de los *objetivos* (ver Fig. 110), es decir, las asignación a los *objetivos* de *actividades* capaces de satisfacerlos. Ahora que se han introducido actores y *objetivos* conviene tener en cuenta cómo este aspecto. La operacionalización de los objetivos, y por extensión las acciones en torno al sistema, son cubiertos por varios *aspectos* de la GCR. Entre otros están: el 1.3 "Acciones del nuevo componente"; el 1.4 "Integración de acciones individuales en otras de más alto nivel"; el 1.8 "Operacionalización de los objetivos"; el 1.9 "Objetivos de las acciones del componente"; 2.7 "División de la labor, incluyendo la distribución síncrona y asíncrona del trabajo entre diferentes localizaciones"; o el *aspecto* 3.8 "Uso de representaciones compartidas para el soporte del trabajo colaborativo".

Para ilustrar el estudio sobre la operacionalización se va a usar la parte referida a la recolección de información sobre los alumnos. El resultado final de este análisis se ve en la Fig. 124. Ahora se pasa a ver cómo se han alcanzado estos resultados con la GCR.

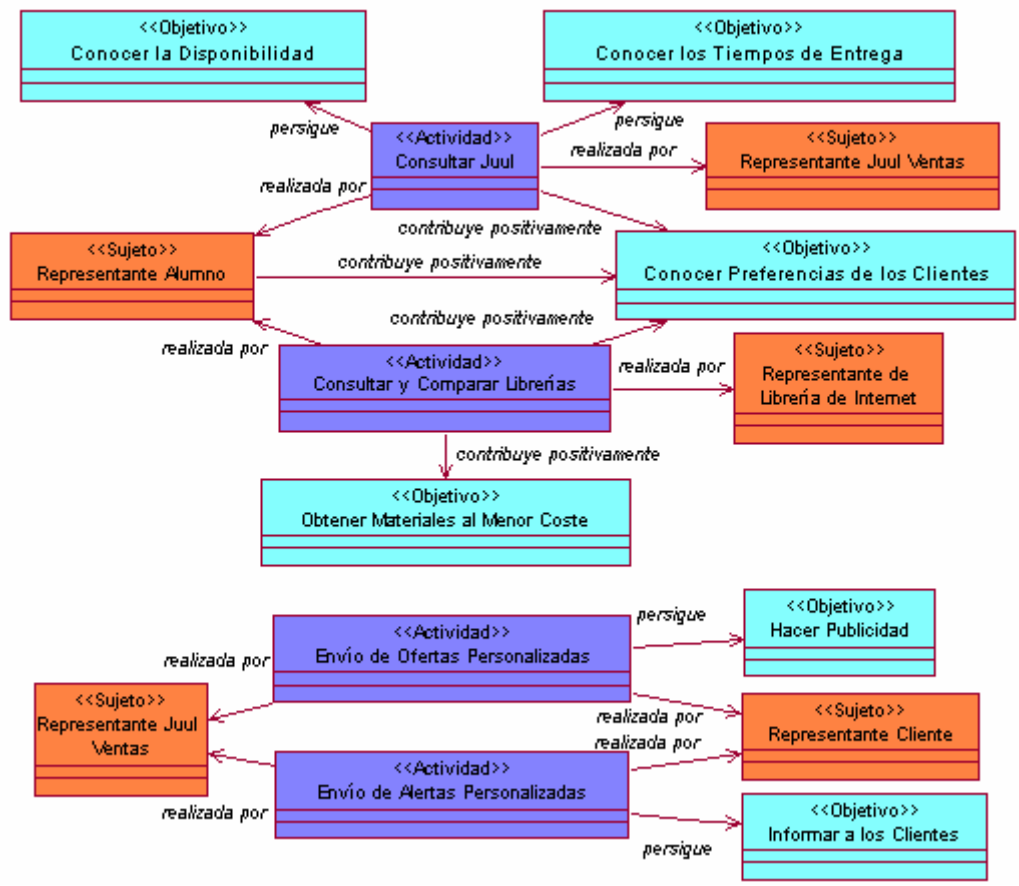


Fig. 124. Operacionalización de los *objetivos* del SMA.

Aquí se comenzará el análisis para asignar tareas a los *objetivos* identificados por el *aspecto* 1.8. Sus *preguntas* permiten establecer las *actividades* que se espera que satisfagan, impidan o influyan sobre los *objetivos*. Los alumnos realizan, entre otras actividades, consultas al sistema de *Juul Møller*. Por ejemplo, cuando realizan un encargo, tratan de conocer cuál es la disponibilidad de un libro, si ya está en la librería o cuánto tiempo pueden tardar en recibirlo. Estas *actividades* aparecen en la Fig. 124. Se trata de *Consultar Juul* y *Consultar y Comparar Librerías*.

Los *objetivos* afectados por las *actividades* de consulta no quedan limitados a los propios de los usuarios. Todas estas consultas son susceptibles de generar información acerca de los hábitos de los clientes. De este modo la empresa puede conocer la clase de libros que compran sus usuarios; qué otras librerías consultan, qué valoran de éstas, sus prioridades a la hora de elegir (e.g. precio o tiempo de entrega). Recoger esta información es un *objetivo* contemplado en la Fig. 123, *Conocer Preferencias de los Clientes*. Esta información permite a *Juul Møller* mejorar su SMA y llevar a cabo servicios de información y publicidad adaptados a los gustos de sus usuarios. La pregunta 1.10.3 "¿Contribuyen de forma indirecta estas acciones a otros

objetivos?" es la que da cuenta de esta situación. Con ella se captura que todas las acciones de consulta realizan tareas que contribuyen al *objetivo* de conocer los hábitos de los clientes como se puede ver en la Fig. 124.

Al introducir las asignaciones entre *actividades* y *sujetos* es posible que se obvie la naturaleza colaborativa de las *actividades*, es decir, el hecho de que algunas *actividades* son realizadas por varios *sujetos*. Además, la participación de los *sujetos* en las acciones influye en el posterior refinamiento de los *objetivos* y de las propias *actividades*. Aquí se ha de recordar que los desarrolladores han de aconsejar a los usuarios para lograr agentes con una granularidad adecuada. El sistema no debe estar formado por pocos agentes con muchas responsabilidades ni por muchos con escasa capacidad [Wooldridge & Jennings 1998]. *Preguntas* válidas en este contexto serían las 2.7.2 y 3.7.3 que trata de determinar qué *actividades* son colaborativas. Se decidió que un reparto adecuado del trabajo implicaría introducir como participantes en las *actividades* de los clientes a los agentes representantes de *Juul Møller* y de las librerías en Internet.

Por último, la parte inferior de la Fig. 124 muestra las *actividades* que hacen uso de la información personalizada de los clientes para enviarles avisos y publicidad personalizada. Estas *actividades* son llevadas a cabo por el agente *Representante Juul Ventas*.

Tras establecer las *actividades* de alto nivel que persiguen los *objetivos*, se prosigue con su refinamiento y se determinan los elementos de su *sistema de actividad*. Mientras que se llevan a cabo estas tareas será necesario revisar los pasos anteriores, por ejemplo con la introducción de nuevos actores y objetivos. El siguiente estudio se centra en los *objetos*, *productos* y *herramientas* de las *actividades* anteriores. Se trata de establecer los elementos que contribuyen a un *objetivo*, los que determinan si ha fallado, aquellos que establecen si ha tenido éxito o los que indican si es factible abordarlo. Esta información es abordada entre otros, por los *aspectos* 1.10 y 1.11 que hacen referencia a los criterios para la satisfacción o fallo de los *objetivos*, el 1.12 que trata de los prerrequisitos de satisfacción de los *objetivos* y el 2.2 que se refiere a los prerrequisitos de la propia *actividad*.

Para delimitar el problema seguimos con el análisis de la porción del SMA relativa al trabajo con los clientes: sus consultas, elaboración de sus perfiles e información personalizada.

La *pregunta* 1.12.1 trata de averiguar qué productos son prerrequisitos imprescindibles para intentar cumplir el *objetivo*. Se trata por tanto de elementos con una relación *esencial* con el *objetivo* (ver sección 3.4). El siguiente análisis de los *objetivos* se divide entre los propios de los clientes y los de *Juul Møller*.

En cuanto a los *objetivos* de los clientes, se encuentra que *Conocer la Disponibilidad* y *Conocer los Tiempos de Entrega* requieren que el agente/*sujeto Representante Alumno* suministre datos acerca del libro sobre el que quiere informarse. En el caso de que el libro forme parte de un pedido ya realizado con *Juul Møller* debería ser identificado. En cuanto al *objetivo Obtener Materiales al Menor Coste*, sólo requiere que el cliente identifique el libro que solicita y cuál sería el lugar donde se haría la entrega. El dato del lugar de entrega sólo es relevante para el *objetivo Obtener Materiales al Menor Coste*. Para los objetivos previos el dato de dónde realizar la entrega carece de importancia, ya que se tratará de una de las

librerías de *Juul Møller*. Sin embargo, cuando se trata de averiguar el menor coste se consultan también las librerías de Internet y entonces el dato es imprescindible.

Sobre los *objetivos* de *Juul Møller*, *Conocer Preferencias de los Clientes* necesita crear un perfil del cliente *a priori* que incluya datos como su nombre y su ocupación (i.e. administración de la NSM, profesor, alumno, ex-alumno u otros). A medida que el usuario trabaje con el sistema se van añadiendo datos a este perfil. Por ejemplo, pedidos realizados, libros consultados, librerías con las que suele comparar a *Juul Møller* o el tiempo de espera de sus peticiones. El *objetivo Informar a los Clientes* hace referencia a sus pedidos o los libros por los que ha mostrado interés. Para cumplir este *objetivo* ha de existir un objeto *Perfil de Usuario* con dicha información. Además, en el caso de los pedidos, necesita que el *Perfil de Usuario* tenga una referencia al pedido de *Juul Møller*. Por último, *Hacer Publicidad* requiere el *Perfil de Usuario* para informarle sólo de las novedades que puedan serle de interés.

Tras considerar los prerrequisitos se contemplan los datos que permiten juzgar el éxito o fallo de los *objetivos*. Para ello se incluyen preguntas como 1.11.1 "¿Cómo se puede saber que el objetivo se ha cumplido? ¿Cuáles son las evidencias/datos/criterios que muestran su satisfacción?" o "¿Hay evidencias/datos que apunten a que el objetivo ha fallado aun sin garantizarlo?". Al abordar estas preguntas se hizo evidente para el cliente que las *actividades* que consideraba la Fig. 124 necesitaban ser refinadas. Las *actividades* se podían descomponer en una parte de petición de la información, otra de creación de la misma y finalmente su recepción. Cada una de ellas tenía sus propios objetivos y criterios de éxito y fallo. Por ello se abandonaron momentáneamente las *preguntas* de los *aspectos* 1.11 y 1.12 y se volvió a las de los *aspectos* 1.3 y 1.8 para refinar las *actividades*.

Las *preguntas* de los *aspectos* 1.3 y 1.8 permitieron refinar las actividades de la Fig. 124. Una vista parcial de este refinamiento para la *actividad Consultar Juul* puede apreciarse en la Fig. 125. La ordenación de dichas *actividades* se realizó cuando fue necesario usando las *preguntas* 2.7.5 a 2.7.8 relativas a la *división del trabajo*. Las *actividades* creadas en estas respuestas se usaron como entradas para la *pregunta* 1.1.6. Con ella se asignaron agentes que fueran capaces de realizar dichas *actividades*. También fue necesario refinar los *objetivos* existentes de tal forma que reflejaran la nueva estructura de *actividades*. Esta tarea usó los *aspectos* 1.7 y 1.9.

Con el nuevo paso de refinamiento concluido y la ordenación de las *actividades*, se retornó a los *aspectos* sobre prerrequisitos de las tareas. La información que se ha de intercambiar entre las nuevas acciones puede ser vista también en la Fig. 125 para el ejemplo de la *actividad Consultar Juul*. A fin de simplificar la Fig. 125 se ha optado por colapsar los *productos* y *objetos* de las *actividades* unidos por una relación de *cambio de rol* y representar sólo los *objetos*. La representación correcta requeriría que una *actividad produce* un *producto* que mediante un *cambio de rol* se transforma en el *objeto* que *consume*/*transforma* otra *actividad*.

Finalmente se volvieron a abordar los criterios de éxito y fallo de los *objetivos* con los *aspectos* 1.11 y 1.12. La Fig. 125 muestra dos formas de expresar el criterio de satisfacción. Una corresponde al caso del *objetivo Solicitar Disponibilidad*, cuya satisfacción se asocia directamente a una *actividad*. Esta relación puede interpretarse como que el *objetivo* queda satisfecho por la mera ejecución de la *actividad* o bien por el único *producto* de dicha *actividad*. El segundo caso corresponde a *Proporcionar Datos de Disponibilidad*. En este caso la satisfacción se vincula directamente a un

objeto mediante las relaciones *garantiza* y *esencial*. Con *garantiza* se indica que la presencia del *objeto* satisface el *objetivo* y con *esencial* que su ausencia lo hace fallar.

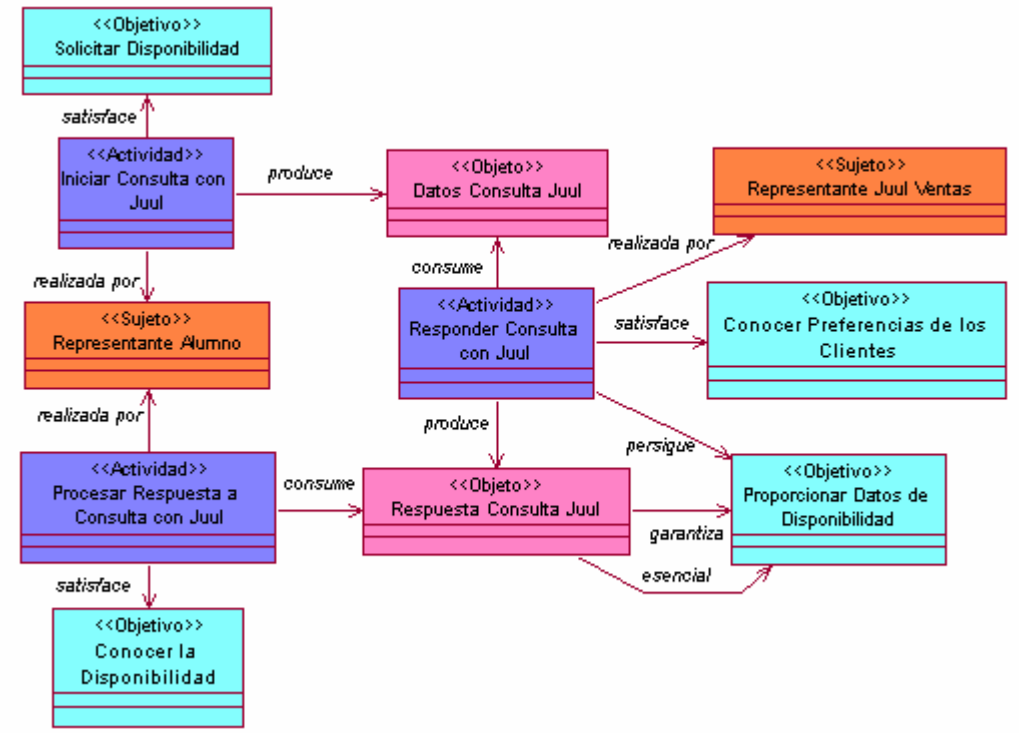


Fig. 125. Refinamiento de la operacionalización de los *objetivos* del SMA para los clientes.

INGENIAS	Teoría de Actividad
Agente, Rol	Representante Alumno Representante Juul Ventas
Tarea, Flujo de trabajo, Interacción	Iniciar Consulta con Juul Responder Consulta con Juul Procesar Respuesta a Consulta con Juul
Objetivo, Estado mental	Solicitar Disponibilidad Conocer Preferencias de los Clientes Proporcionar Datos de Disponibilidad Conocer Disponibilidad
Recurso, Aplicación, Interacción, Unidad de Interacción, Hecho, Creencia, Evento, Estado mental, Consulta de entidades autónomas	Datos Consulta Juul Respuesta Consulta Juul

Tabla 18. Traducción a INGENIAS de las consultas en *Juul Møller*.

El último paso acerca de los requisitos es su traducción al lenguaje de la metodología para SMAs. A modo de ejemplo se va a realizar la traducción de la Fig. 125. Como se indicó al introducir el caso de estudio, el lenguaje objetivo es el de la metodología INGENIAS [Pavón & Gómez-Sanz 2003]. El proceso de traducción está recogido en la sección 3.5 y las correspondencias para INGENIAS pueden verse en la sección 3.5.1.1. Estas correspondencias son sólo una primera traducción inmediata elemento a elemento. Como queda reflejado en la Tabla 18, los elementos de la Fig. 125 admiten varias traducciones. Las relaciones no han sido incluidas en esta tabla.

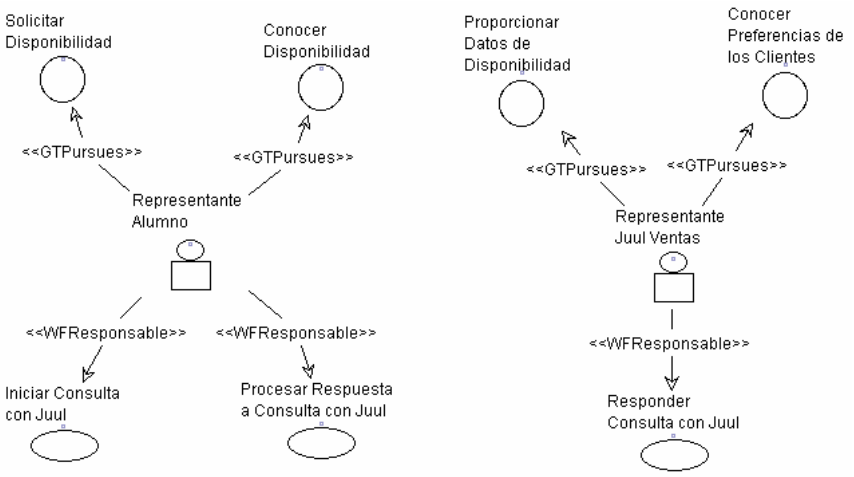
Teoría de Actividad	**INGENIAS**
actividad → realizada por → *sujeto* *actividad* → satisface → *objetivo*	- Diagrama de Agentes *agente* (*sujeto*) → GTPursues → *objetivo* (*objetivo*) *agente* (*sujeto*) → WFResponsible → *tarea* (*actividad*)
	- Diagrama de Tareas y Objetivos *tarea* (*actividad*) → GTSatisfies → *tarea* (*actividad*)
actividad → produce → *producto*	- Diagrama de Tareas y Objetivos *tarea* (*actividad*) → WFProduces → *hecho* (*producto*)
actividad → consume → *objeto*	- Diagrama de Tareas y Objetivos *tarea* (*actividad*) → WConsumes → *hecho* (*objeto*)
actividad → produce → *producto* → garantiza → *objetivo*	- Diagrama de Tareas y Objetivos *estado mental* → AContainsME → *hecho* (*producto*)
	- Diagrama de Tareas y Objetivos *tarea* (*actividad*) → GTSatisfies → *objetivo* (*objetivo*)
	- Condición de satisfacción de GTSatisfies *estado mental*
actividad → produce → *producto* → esencial → *objetivo*	- Diagrama de Tareas y Objetivos *estado mental* → AContainsME → *hecho* (*producto*)
	- Diagrama de Tareas y Objetivos *tarea* (*actividad*) → GTSatisfies → *objetivo* (*objetivo nuevo*)
	- Condición de satisfacción de GTSatisfies *estado mental*
	- Diagrama de Tareas y Objetivos *objetivo* (*objetivo nuevo*) → ContributePositively → *objetivo* (*objetivo*)

Tabla 19. Correspondencias entre estructuras de INGENIAS y la TA para *Juul Møller*.

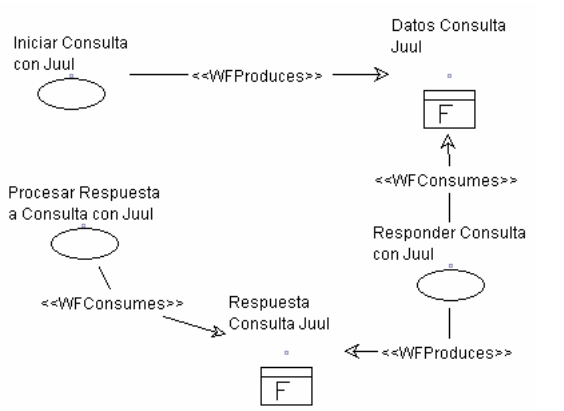
La traducción de la Tabla 18 puede mejorase sensiblemente si se emplean correspondencias que contemplen estructuras. Por ejemplo, una *actividad* que *satisface* un *objetivo* puede traducirse en INGENIAS a una *tarea* que *GTSatisfies* un *objetivo* en un diagrama de *Tareas y Objetivos*. De igual modo, el *sujeto* que realiza la *actividad* que satisface un *objetivo* es en INGENIAS un *agente* o un *rol* que es *WFResponsable* de una *tarea* y que *GTPursues* un *objetivo*. A medida que se emplean

correspondencias más complejas se reduce la ambigüedad de la traducción. No obstante, hay que recordar que las correspondencias raramente son uno a uno, incluso entre estructuras. Las correspondencias entre estructuras de INGENIAS y la TA usadas para traducir la Fig. 125 pueden verse en la Tabla 19. Como convención para la Tabla 19, los elementos asociados por relaciones se escriben en cursiva y las relaciones no.

a) Agentes



b) Flujo de trabajo



c) Relaciones de satisfacción

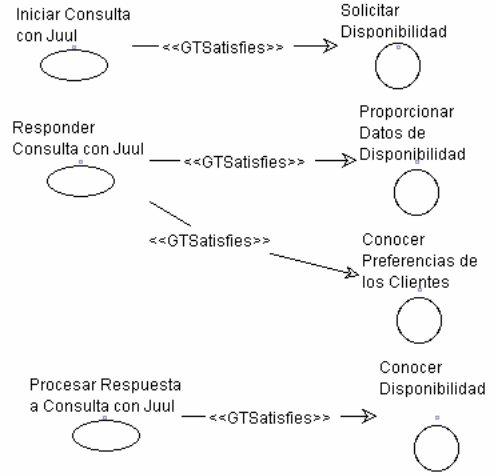


Fig. 126. Traducción a INGENIAS de la operacionalización de los *objetivos* del SMA para los clientes en la Fig. 125.

La Fig. 126 muestra la traducción de la Fig. 125 usando las correspondencias en la Tabla 19. Las relaciones *garantiza* y *esencial* entre *objetos* y *objetivos* de la Fig. 125 se incluirían en INGENIAS como *condiciones de satisfacción* de las relaciones *GTSatisfies*. Estas *condiciones de satisfacción* no se muestran en la Fig. 126.

La traducción a la metodología de agentes no es un mero cambio de nombres para entidades, relaciones y roles. El ejemplo de la Fig. 125 se tradujo a tres diagramas en la Fig. 126. Aunque se haya dado una única correspondencia, la traducción de las relaciones *produce* de la TA viene dada por el tipo de diagrama destino en INGENIAS. La selección del diagrama depende a su vez de las entidades que participan en el original de la TA y sus traducciones. En este caso el diagrama que representa el flujo de trabajo puede ser traducido en INGENIAS por uno de *Tareas y Objetivos* o por uno de *Organización*. Al seleccionar un diagrama de *Tareas y Objetivos* una traducción válida para *produce* era *WFProduces*.

Con este ejemplo de traducción termina la sección dedicada a los requisitos del SMA para *Juul Møller*. Esta sección ha mostrado cómo se pueden capturar los objetivos de los actores del entorno en relación con el sistema que se planea construir, cómo refinarlos y asignarlos a actividades y cómo comenzar el diseño de la arquitectura del sistema con el diseño de sus agentes y de los flujos de trabajo.

La siguiente sección se centra en cómo hacer evolucionar las especificaciones disponibles sobre el SMA de *Juul Møller* según las contradicciones de la TA, tal y como se vio en el Capítulo 5.

7.4. Diseño guiado por contradicciones

En el Capítulo 5 se introdujeron técnicas para trabajar con contradicciones de tal modo que estas indicarán posibles vías de evolución de las especificaciones. De este modo, siguiendo las ideas presentes en la TA acerca del conflicto como motor de cambio, cada nueva versión de las especificaciones surge como un intento de solución a las contradicciones existentes en las versiones previas.

Esta sección ofrece sobre el ejemplo del *Juul Møller Bokhandel A/S* la detección y solución de varias contradicciones detectadas en las especificaciones de la sección anterior. Concretamente se trata de las contradicciones del *Valor de Intercambio* (ver sección 5.4.2), el *Objetivo Social* (ver sección 5.4.3), la *Interrupción de Uso* (ver sección 5.4.6), el *Estado de Necesidad* (ver sección 5.4.8) y la *Información Contradictoria* (ver sección 5.4.10). El trabajo con estas contradicciones generará una nueva versión de las especificaciones donde se resuelven algunos de los problemas acerca de la colaboración con los distribuidores, el catálogo del curso, los pedidos de los alumnos y la preocupación de los trabajadores de *Juul Møller* por sus puestos. Además se detecta y solventa un nuevo problema acerca de la *actividad Consultar y Comparar Librerías*.

7.4.1. Colaboración con los distribuidores

Un aspecto importante del modelo de negocio en Internet previsto para *Juul Møller* es la colaboración con sus distribuidores. Las especificaciones de la sección 7.3

(véanse principalmente las Fig. 111, Fig. 113 y Fig. 119) muestran que objetivos como *Reducir Precios de los Libros* u *Obtener Materiales a Tiempo* dependen de *Negociar con los Distribuidores* (ver la Fig. 119).

Para que exista colaboración efectiva con los distribuidores, estos también han de ver satisfechos, o al menos no perjudicados, sus propios objetivos. La Fig. 117 mostró los principales objetivos de los distribuidores respeto al comercio electrónico en general, dentro del cual se contempla su asociación con el SMA de *Juul Møller*. La colaboración con *Juul Møller* les ayuda a cumplir sus metas de *Aumentar Ventas* (es de esperar que *Juul Møller* incremente sus pedidos), *Lograr Ventajas Competitivas Sobre sus Rivales* (esta colaboración le permitirá reducir coses administrativos) y *Proporcionar Soporte a los Clientes*. El único objetivo que puede verse perjudicado es el de *Reducir Carga de Trabajo*. El distribuidor necesitará quizás mantener el sistema para comunicarse con el SMA, grabar datos y habrá nuevas exigencias sobre su funcionamiento interno (por ejemplo dejará de ser admisible que su catálogo de almacén esté desactualizado).

Que *Juul Møller* sea capaz de convencer a sus distribuidores para que colaboren en su nueva infraestructura dependerá de que sea capaz de darles ventajas adicionales sobre la situación actual que compensen los posibles nuevos inconvenientes. El comercio electrónico ha de ser una colaboración beneficiosa para ambos participantes.

7.4.1.1. Valor de Intercambio

La situación que se produce entre *Juul Møller* y sus distribuidores corresponde a una contradicción del *Valor de Intercambio* (ver sección 5.4.2). *Juul Møller* desea obtener unos servicios de los distribuidores. Sin embargo los distribuidores no se ven suficientemente compensados por la realización de estas actividades. Una instanciación en el problema concreto que nos ocupa puede verse en la Fig. 127.

La Fig. 127 muestra el problema del *Valor de Intercambio* para la situación concreta en la que *Juul Møller* está recopilando información para realizar un pedido. El *Distribuidor* es el encargado de proporcionar esta información. Aunque realizar esta *actividad* le ayuda a satisfacer su *objetivo* de *Proporcionar Soporte a los Clientes*, también le genera trabajo adicional. Este conflicto puede hacer que el Distribuidor no se muestre muy receptivo a colaborar con el SMA de *Juul Møller*.

Existen ciertas diferencias entre la Fig. 127 y el patrón de detección del *Valor de Intercambio* (ver Fig. 58). El primer contraste aparece respecto a los *objetivos* de la *actividad* en la parte superior del diagrama. En la Fig. 127 esta *actividad* afecta a dos *objetivos*. El patrón original sólo considera el *objetivo* con la relación *persigue*. El *objetivo* con la relación *contribuye negativamente* se ha añadido para resaltar que existen desventajas para el *Distribuidor* en la ejecución de la *actividad*. La segunda diferencia es el *objeto Datos Libros en Almacén*. Como ha ocurrido en otras ocasiones, en el patrón original aparece un *Artefacto*, que puede ser reemplazado por cualquier otro *rol* de la TA. En este caso ha sido sustituido por un *objeto*. La última diferencia es la inclusión de la *comunidad Juul Møller*. La relación de *cambio de rol* tiene sentido de una *comunidad* a un *sujeto*. Una *actividad* puede ser realizada por un *sujeto* social que quizás a otro nivel se descomponga en individuos explícitos. Lo mismo ocurre con la persecución de objetivos. Incorporando explícitamente la *comunidad* a la Fig. 127 se resalta el hecho de que esos objetivos y actividades afectan a *Juul Møller* en su conjunto.

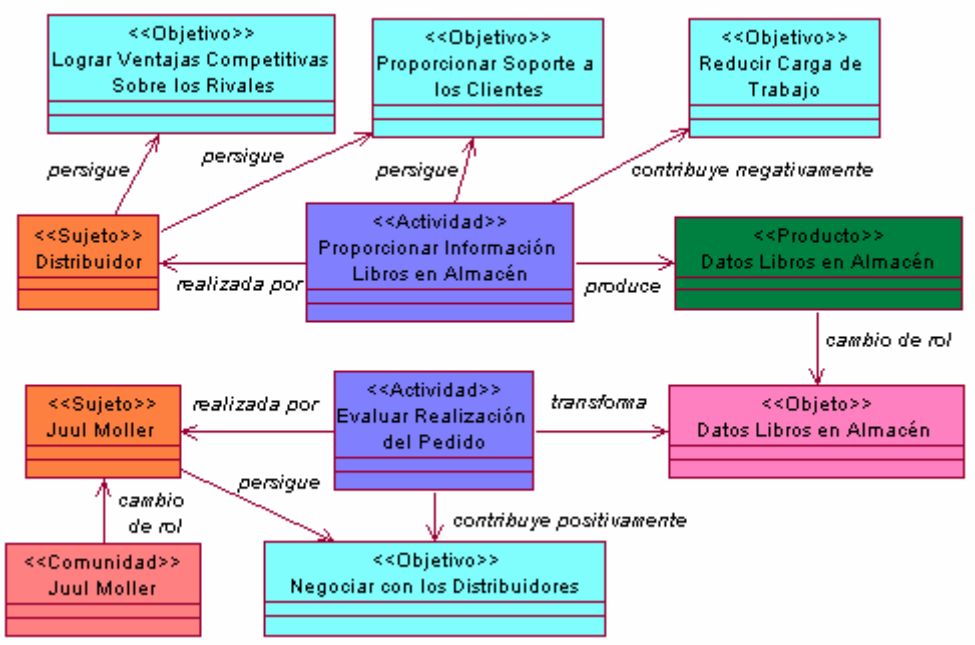
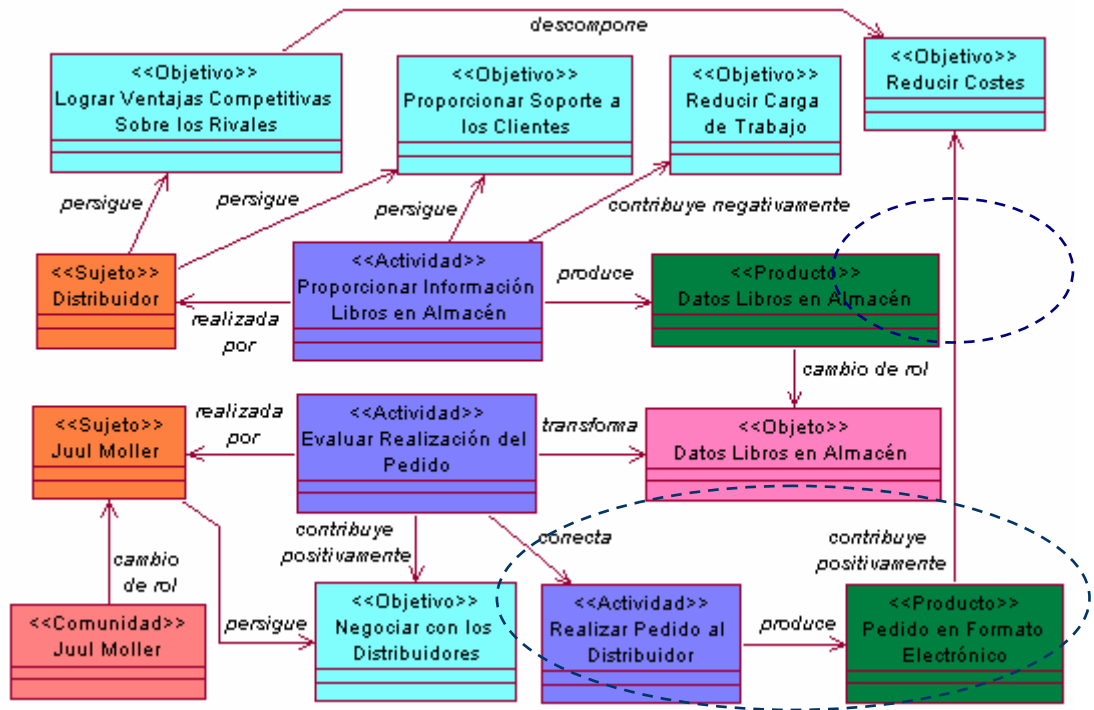


Fig. 127. Contradicción del *Valor de Intercambio* para la colaboración con los distribuidores.

7.4.1.2. Solución al Valor de Intercambio de *Juul Møller* y los distribuidores

El *Valor de Intercambio* suele resolverse incorporando algún tipo de recompensa al *sujeto* que debe prestar el servicio a la *comunidad*. De esta forma se incrementa la utilidad que tiene para el individuo realizar la *actividad* del servicio. En el caso de la Fig. 127 se trataría de que *Juul Møller* ejecutara una nueva actividad que ayudase al *Distribuidor* a satisfacer uno de sus *objetivos*.

La solución de la contradicción con la *actividad Realizar Pedido al Distribuidor* como *Actividad de Retribución* se muestra en la Fig. 130. En el marco de un sistema para comercio electrónico, *Juul Møller* y el *Distribuidor* no necesitan ya intercambiar llamadas de teléfono o faxes para hacer los pedidos. Estos llegan en formato electrónico al *Distribuidor* que puede procesarlos automáticamente con su software.



7.4.2. Promoción de pedidos anticipados

Las especificaciones anteriores (véanse principalmente las Fig. 113 y Fig. 119) dejaron abiertos ciertos conflictos sobre la planificación del curso y la obtención de sus materiales.

Se vio que la NSM (Fig. 113), dentro de su objetivo de *Formar a los Estudiantes*, deseaba que estos dispusiesen de los materiales de los cursos al menor precio (i.e. *Rebajar el Coste de los Medios para Clase*) y a tiempo para el comienzo de las clases. (i.e. *Reducir el Tiempo de Adquisición de los Medios para Clase*) Para ello era preciso que sus profesores elaboraran con la suficiente anticipación el *catálogo del curso* (i.e. *Dar Catálogo del Curso*). De esta forma *Juul Møller* podía prever los pedidos de los alumnos (i.e. *Conocer los Pedidos con Antelación*) y hacerse con los libros suficientes como para atender las peticiones iniciales. Esto redundaba en unos tiempos de entrega menores a sus clientes (i.e. *Tener los Libros para los Clientes Pronto*). Por otra parte (ver Fig. 119), saber cuándo sería necesario el material y cuánto supondría el pedido total del mismo, permitía a *Juul Møller* abordar negociaciones con los distribuidores para abaratar su coste (i.e. *Mejorar Comunicación con los Distribuidores* y *Negociar con los Distribuidores*). Si estas negociaciones tenían éxito, los alumnos podían beneficiarse de una reducción en el precio de sus libros (i.e. *Reducir Precios de los Libros*). Además, conocer los datos de pedidos con antelación permitía a *Juul Møller* una mejor organización de su entrega (i.e. *Reducir Tiempos de Entrega*) y abaratar sus costes (i.e. *Obtener Materiales al Menor Coste*).

Las ventajas anteriores tenían también sus contrapartidas en estos flujos de trabajo (ver Fig. 113). Para los profesores, anticipar el *catálogo del curso* (i.e. *Dar Catálogo del Curso*) suponía perder la flexibilidad de cambiar más tarde los libros que deseaban usar en sus clases (i.e. *Tener Flexibilidad en los Medios del Curso*). Para los alumnos, comprometerse a pedir los libros a *Juul Møller* (i.e. *Comprar en Juul Møller*) suponía que no podían cambiar y pedirlos en otra librería (i.e. *Comprar en la Mejor Tienda*) en caso de que las condiciones finales de entrega no les satisfaciesen. No obstante, este inconveniente quedaba mitigado por el hecho de que ahora el SMA de *Juul Møller* les permitía consultar los plazos previstos de entrega antes de la solicitud (ver Fig. 124), con lo que podían conocer las condiciones antes de realizar el pedido. Finalmente, para *Juul Møller*, esgrimir las ventajas de plazos de entrega o precios, podría colocarle en la difícil situación de que sus clientes le exigiesen condiciones de difícil cumplimiento.

El nexo de todos estos objetivos es el compromiso de sus participantes en el acuerdo entre *Juul Møller* y la NSM. Este acuerdo especifica que *Juul Møller* contará anticipadamente con el catálogo del curso a cambio de lo cual compensará económicamente a la NSM y a la SU. Implícitamente supone que los libros estarán disponibles antes del comienzo del curso y que los alumnos podrán adquirirlos a precios ajustados. Gracias a estos acuerdos *Juul Møller* incrementa sus ventas y se garantiza una clientela fiel. Esta información está reflejada en la red de objetivos y contribuciones entre estos de las Fig. 113 y Fig. 119.

Para que el acuerdo funcione es evidente que todas las *comunidades* implicadas han de cumplir con lo que se espera de ellas. Si demasiados *sujetos* de esas *comunidades* violan el acuerdo deja de ser interesante para todos. Por ejemplo, si pese

al acuerdo muchos profesores no proporcionan el catálogo, *Juul Møller* no tendrá los libros a tiempo y los alumnos los comprarán en sus competidores, con lo cual la librería perderá dinero. Lo mismo sucede si pese a disponer del catálogo del curso los distribuidores no proporcionan los libros a tiempo a *Juul Møller*, que éste dejará de ser una opción atractiva para los alumnos. También, si *Juul Møller* pone los libros a precios demasiado altos, los alumnos preferirán adquirirlos en otras librerías.

La clase de razonamientos que se acaban de esgrimir son fruto de un análisis cualitativo de los objetivos y las relaciones de contribución. Estas relaciones permiten saber qué objetivos se ve perjudicados o beneficiados cuando se satisfacen o fallan otros objetivos.

7.4.2.1. Objetivo Social

Las especificaciones del SMA procedentes de la captura de requisitos se pueden comprobar en busca de contradicciones. Concretamente, la contradicción descrita en la introducción de este problema corresponde a un *Objetivo Social* (ver sección 5.4.3). Su instanciación en el problema concreto que nos ocupa puede verse en la Fig. 127.

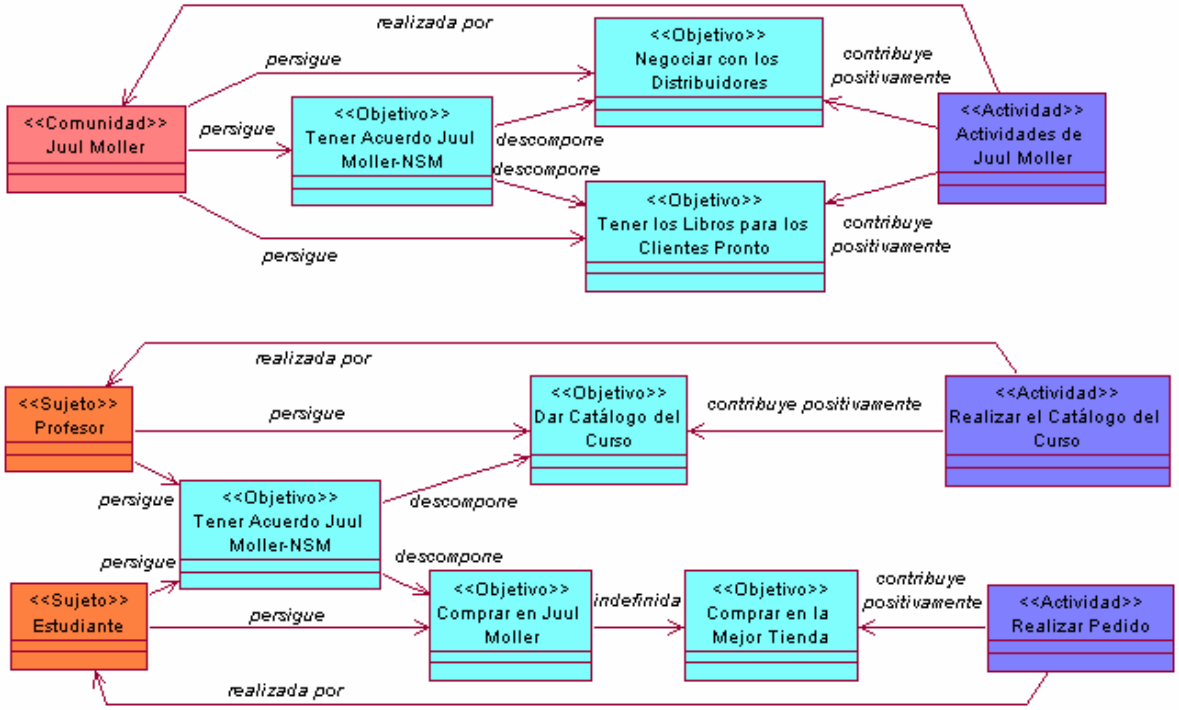


Fig. 129. Contradicción del *Objetivo Social* para la petición anticipada de libros.

La Fig. 127 muestra una generalización a varios *sujetos* de la presentada en la Fig. 60. En lugar de trabajar con sólo dos *sujetos* como ocurría en la original, ésta versión se ha generalizado para incluir tres. El *objetivo* social considerado es *Tener Acuerdo Juul Møller-NSM*. Este objetivo es construido colaborativamente por tres actores: *Juul Møller*, los profesores y los estudiantes, donde estos dos últimos son instancias de

sujetos dentro de la *comunidad* NSM. *Juul Møller* es una *comunidad* pero mediante un cambio de rol (omitido en la figura) puede desempeñar el papel de *sujeto*. Cada uno de estos actores ha de llevar a cabo ciertos *objetivos* para que el acuerdo funcione globalmente. De *Juul Møller* se espera que tenga los libros antes del comienzo del curso y que, negociando con los distribuidores, obtenga rebajas en los precios finales de los libros. Obsérvese que en este caso el *sujeto Juul Møller* está cumpliendo dos objetivos para el objetivo social común. Una vez más esta descomposición es una generalización que no afecta al significado de la contradicción. Los profesores han de *Dar el Catálogo del Curso* con anticipación para que se puedan iniciar en la librería los trámites para conseguir el material. Finalmente, los beneficios del acuerdo han de hacer que adquirir los libros en *Juul Møller* sea la mejor opción de compra para los estudiantes. El *objetivo* inicial de estos estudiantes es sólo *Comprar en la Mejor Tienda* y *Comprar en Juul Møller* es una opción más (contribución *indefinida* en la Fig. 113). Sin embargo, bajo las condiciones del acuerdo, el *objetivo Comprar en Juul Møller* les garantiza que esa será su mejor opción[9].

La contradicción radica en que ninguno de los involucrados tiene información fehaciente del beneficio global de *Tener Acuerdo Juul Møller-NSM* y algunos ni siquiera conocen su papel en dicho acuerdo. Por ejemplo, los profesores saben que han de proporcionar el *catálogo del curso* pero en caso de que se retrasen, *Juul Møller* no les informa de cómo se ha visto afectada la entrega del material por ello. Por otra parte, los alumnos ni siquiera saben la influencia que tienen sus pedidos individuales en los plazos de entrega o en los beneficios de la SU. Así, los profesores pueden pensar que los alumnos obtendrían igual los libros aunque ellos fijasen el material más tarde; los alumnos podrían concluir que pueden obtener su material en otras librerías sin que cambie nada entre la NSM y *Juul Møller*; *Juul Møller* puede pensar que su presencia en el campus le permitirá seguir teniendo clientes sin necesidad del acuerdo. Es decir, los involucrados desconocen el efecto de sus acciones individuales sobre los beneficios de la *comunidad* del acuerdo.

7.4.2.2. Solución al Objetivo Social de *Juul Møller* y la NSM

La solución a la contradicción de *Objetivo Social* pasa por hacer partícipes a todos los *sujetos* de los beneficios logrados en la satisfacción del objetivo global. En este caso se trata de comunicar a las partes implicadas cómo van mejorando o empeorando sus expectativas y las del resto de los participantes en el acuerdo según los diferentes actores van realizando o no las *actividades* que tienes encomendadas.

En el caso propuesto para el acuerdo entre *Juul Møller* y la NSM, el patrón de solución de la contradicción podía plasmarse de diferentes formas ya que hay varios *sujetos* implicados. La solución consiste en informar a los participantes de aquellos resultados que les interesen en relación con el objetivo social. Por tanto se admiten diferentes combinaciones acerca de la información intercambiada, su origen y sus destinatarios. Dados los requisitos de la sección anterior, se determinó que la información relevante para los profesores era cuándo estaban disponibles los materiales para su adquisición en la librería. A partir de ese momento podían considerar que los alumnos los tendrían para sus clases. Esta información debía ser

[9] La contribución *indefinida* puede reemplazar en los razonamientos sobre contribuciones a cualquier relación de contribución como se vio en la sección 3.4.1.

suministrada por *Juul Møller*. Los alumnos estaban interesados tanto en los plazos de entrega como en el precio final de los libros para su compra. Esta información también la debía proporcionar también *Juul Møller*. Para potenciar los pedidos anticipados, *Juul Møller* podría incluir con la información de los precios las posibles reducciones al incrementarse el número de pedidos. De esta manera los propios alumnos estarían interesados en que sus compañeros incluyesen sus pedidos en el SMA. Por último, para *Juul Møller*, los datos relevantes son el catálogo del curso, que han de proporcionar los profesores, y los pedidos de los alumnos. La Fig. 128 ilustra esta solución. Para simplificar los diagramas se ha omitido el cambio de rol de los *sujetos* cuando se convierten en el *objeto* de las *actividades de intercambio* que aparecen en la Fig. 61. Estas *actividades* que transforman *sujetos* representan las tareas que les transmiten la información.

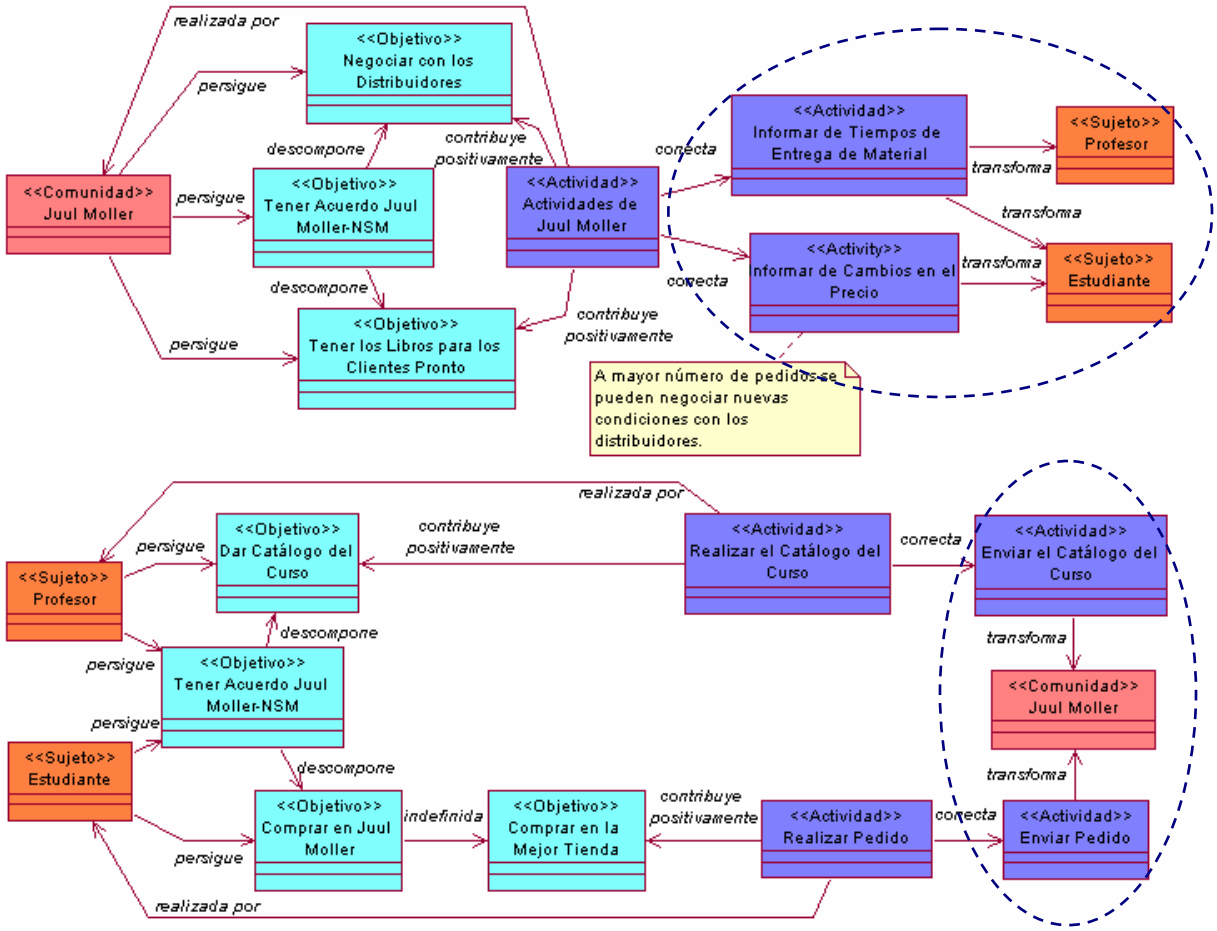


Fig. 130. Solución al *Objetivo Social* sobre el acuerdo *Juul Møller*-NSM.

7.4.3. Revisión de la interfaz con las librerías en Internet

La contradicción de este apartado no está relacionada con la interacción entre el SMA y su contexto social, sino con su propio funcionamiento como software basado en el paradigma de agentes. Se trata de su adaptabilidad ante un entono cambiante y su capacidad de aprendizaje. Buena parte de la literatura sobre agentes [Maes 1994, Sykara 1998] hace hincapié en la flexibilidad y adaptabilidad de los SMAs. Los agentes han de reaccionar de acuerdo con una agenda propia sujeta a cambios y un entorno en permanente evolución. Puesto que una de las características que diferencian a un agente en relación con otros tipos de software es su autonomía, la interacción con el usuario para solventar la adaptación ha de ser mínima. Por ello estas características de flexibilidad, adaptabilidad y autonomía están asociadas frecuentemente con capacidades de aprendizaje de los agentes.

Al tratar los requisitos se vio que uno de los servicios que presta el SMA de *Juul Møller* es el de consultar otras librerías en Internet. Esta consulta forma parte del sistema para recoger información sobre los clientes. Permite averiguar cuáles son los competidores de *Juul Møller* en el mercado de librerías en Internet, las características de sus servicios más apreciadas e incluso las compras que les realizan los alumnos. Sin embargo, prestar estos servicios desde el SMA requiere conocer las interfaces de dichas librerías. Estas interfaces se hallan sujetas a cambios sin previo aviso. En muchos casos, estos cambios se realizan periódicamente para evitar la actuación sobre los sitios *web* de software. Si se desea que el SMA reaccione por sí mismo a estos cambios es necesario contemplar capacidades de adaptación y aprendizaje a ellos.

Una de estas librerías en Internet es Amazon.com, que ya apareció en la introducción de este caso de estudio. La siguiente URL corresponde a una consulta sobre el libro "*The Russian Theory of Activity*" [Bednyi & Meister 1997]:

http://www.amazon.com/exec/obidos/search-handle-url/
index%3Dstripbooks%26field-keywords%3D
The%252520Russian%252520Theory%252520of%252520Activity%26
store-name%3Dbooks/102-5356946-5539310

La anterior URL puede ser construida automáticamente por un agente. Se necesita saber los diferentes tipos de consultas existentes y para cada tipo sus constituyentes. En el ejemplo de la consulta de un libro los elementos son:

- *Prefijo al elemento consultado*. http://www.amazon.com/exec/obidos/search-handle-url/index%3Dstripbooks%26field-keywords%3D
- *Separadores entre los componentes del elemento consultado*. %252520
- *Sufijo al elemento consultado*. %26store-name%3Dbooks/102-5356946-5539310

Con la anterior información, un agente puede ser capaz de automatizar el acceso a los sitios *web* para las consultas de los usuarios, ya que sabe cómo se construyen dichas consultas. El aprendizaje consiste aquí en automatizar la recolección y tratamiento de esta información sobre composición de URLs. Hay que conocer el formato de las páginas y cuándo se producen cambios.

Esta funcionalidad de acceso y seguimiento se introdujo en los requisitos como la *actividad Consultar y Comparar Librerías* (ver Fig. 124). Su descomposición sólo

contemplaba la interacción entre el agente *Representante Alumno* y el *Representante de Librería de Internet* para realizar la petición de la información, la recolección de la misma de los sitios *web* de las librerías y finalmente su recepción y composición. La Fig. 131 muestra este flujo de trabajo.

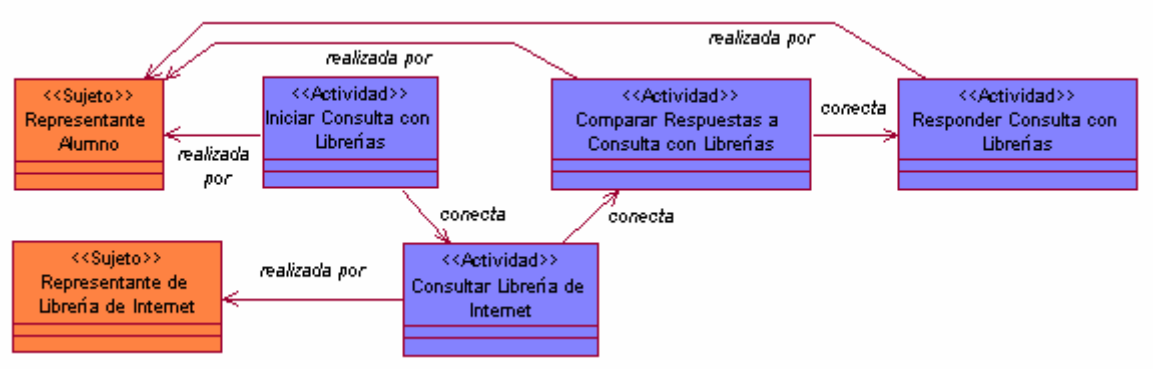


Fig. 131. Búsqueda y comparación de libros en Internet.

7.4.3.1. Interrupción de Uso

La situación ilustrada en la introducción de la *actividad Consultar y Comparar Librerías* corresponde a una contradicción de *Interrupción de uso* (ver sección 5.4.6). Esta contradicción surge cuando se produce un fallo en la *actividad* del *sujeto*, en este caso que el *Representante de Librería de Internet* no puede acceder correctamente a su sitio *web*. Este fallo en la *actividad* debe originar un ciclo de reflexión. En esta reflexión, el *sujeto* estudiará el fallo, determinará sus causas y si es necesario adaptará su comportamiento. Si el *sujeto* carece de la información necesaria para este aprendizaje se considera que existe una *Interrupción de Uso*.

La *Interrupción de Uso* de la *actividad Consulta Librería de Internet* queda reflejada en la Fig. 132. El fallo de la *actividad* al usar la *herramienta Interfaz con Librería* no lleva aparejada la *actividad* que procesa el fallo. El aprendizaje y la adaptación no son por tanto posibles.

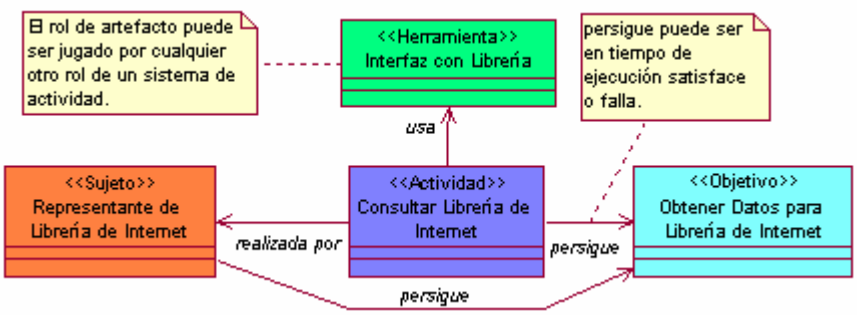


Fig. 132. Contradicción de *Interrupción de Uso* para el acceso a las librerías de Internet.

La Fig. 132 contiene algunas sustituciones respeto al patrón de detección original de la *Interrupción de uso* (ver Fig. 66) que hay que comentar. La primera es la sustitución del *artefacto Elemento* original por la *herramienta Interfaz con la Librería*. Aquí se debe recordar que un *artefacto* es un comodín que puede ser reemplazado por cualquier otro *rol* de la TA. El segundo elemento a considerar es la relación *falla* que ha sido sustituida por *persigue*. *persigue* es una relación genérica entre *actividades* y *objetivos* que en ejecución puede ser sustituida por *satisface* o *falla* dependiendo del resultado final de la *actividad*.

7.4.3.2. Solución a la Interrupción de Uso para librerías de Internet

La solución de la *Interrupción de Uso* consiste en introducir la *Actividad de Reflexión*. Esta *actividad* recoge la información sobre el fallo del *objetivo Obtener Datos para Librería de Internet* y hace uso de la *herramienta* de acceso. Con la información que obtiene de ambos generará el diagnóstico que servirá de base para la adaptación del acceso. Esta solución queda reflejada en la Fig. 133.

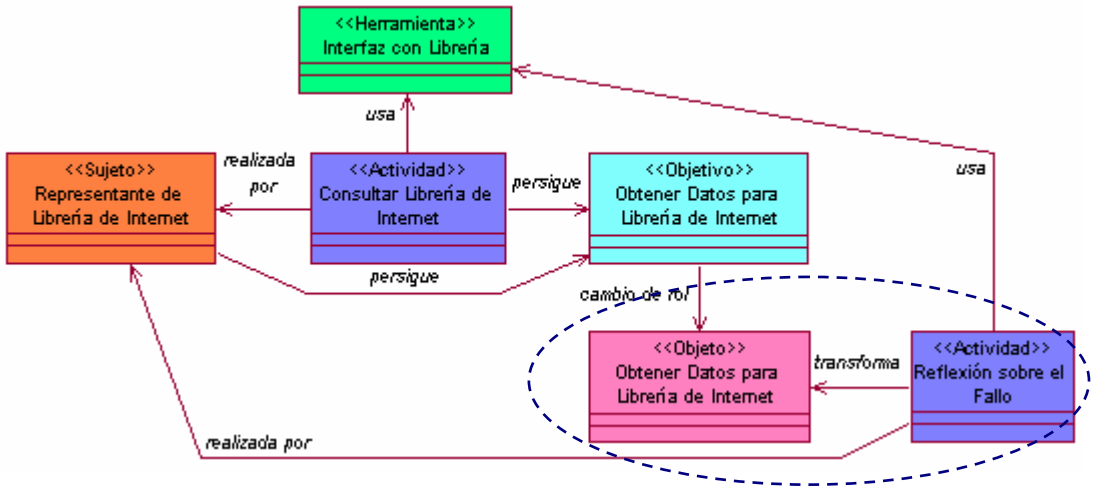


Fig. 133. Solución de la *Interrupción de Uso* para el acceso a las librerías de Internet.

La nueva *actividad Reflexión sobre el Fallo* sólo es el primer elemento necesario para el proceso de diagnóstico y aprendizaje. En caso de que se deseen incluir estas capacidades, se tendrá que refinar la *actividad*, determinar sus *objetos* y *herramientas*, quizás establecer interacciones con otros *sujetos*... Esta contradicción, al igual que las restantes, da posibles soluciones de una forma simple, pero son los desarrolladores los que han de concretarlas al problema y necesidades concretos.

7.4.4. Cambios en el modelo laboral de ***Juul Møller***

Los trabajadores de *Juul Møller* se cuentan entre los actores afectados por la creación del SMA. En la Fig. 115 se reflejaba su preocupación porque la llegada del sistema ocasionase perdida de puestos de trabajo. La llegada del SMA supondrá la automatización de buena parte de las tareas "administrativas" sobre los libros, tales como la introducción de los inventarios o algunas consultas a los distribuidores. Esta automatización haría que fuesen necesarios menos trabajadores en estas tareas. El posterior desarrollo del contexto del SMA con la GCR quedó reflejado en la Fig. 120. Se mostró que la empresa pretendía conservar los puestos de trabajo aunque reconvirtiéndolos a otras funciones. Los trabajadores dejarían de hacer las tareas más repetitivas y se centrarían en el trato con clientes y distribuidores. Esta reconversión de trabajo ilustra una situación en la que la *actividad* de un agente externo, el SMA en este caso, conduce a una situación de cambio en las *actividades* de los actores para continuar satisfaciendo sus necesidades.

7.4.4.1. Estado de Necesidad

El cambio en las ocupaciones de los trabajadores corresponde a una contradicción de *Estado de Necesidad*. Existen trabajadores que realizaban ciertas labores en *Juul Møller* que se convertirán en funciones del SMA. Estas labores tenían una utilidad para la empresa lo que les permitía conservar su trabajo. Con la llegada del SMA, la utilidad de esta labor humana se reduce y deja de garantizar el trabajo. Esta situación se refleja en la Fig. 134.

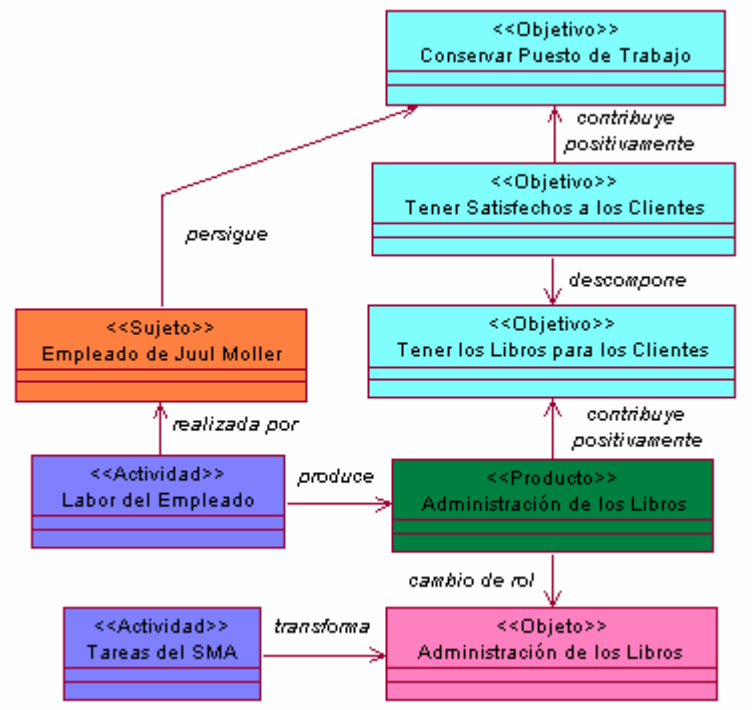


Fig. 134. Contradicción del *Estado de Necesidad* para los trabajadores de *Juul Møller*.

La identificación de la Fig. 134 con el patrón de detección de la Fig. 70 es posible gracias al razonamiento cualitativo sobre la cadena de *objetivos*[10]. En el patrón original el *sujeto persigue* directamente el *objetivo* al que *contribuye positivamente* el *producto*. La Fig. 134 responde a la misma situación. El *producto Administración de los Libros* ayuda a satisfacer el *objetivo Tener los Libros para los Clientes*. Gracias a la relación *descompone*, este objetivo es una forma de *Tener Satisfechos a los Clientes*. A su vez, cumplir este último objetivo contribuye positivamente al objetivo del *sujeto* que es *Conservar Puesto de Trabajo*. La relación *descompone* entre los objetivos aparece en la Fig. 111 y la de contribución positiva a *Conservar Puesto de Trabajo* en la Fig. 120. De esta manera, el *producto* tiene una relación *contribuye positivamente* deducida con el *objetivo* perseguido por el *sujeto*.

En el diagrama, la contradicción surge de que *producto Administración de los Libros* ya no es exclusivo de la *actividad* del *Empleado de Juul Møller*, lo que le resta utilidad para el objetivo de conservar el trabajo.

7.4.4.2. Solución al Estado de Necesidad de los trabajadores

La solución propuesta a una situación de *Estado de Necesidad* (ver Fig. 71) pasa por añadir nuevas actividades que constituyan alternativas para alcanzar el *objetivo*

[10] Estos razonamientos se basan en la definición del lenguaje UML-TA y las equivalencias entre sus primitivas introducidos en la sección 3.4.

que la antigua *actividad* ya no garantiza. Esta solución sería la reflejada en la Fig. 135.

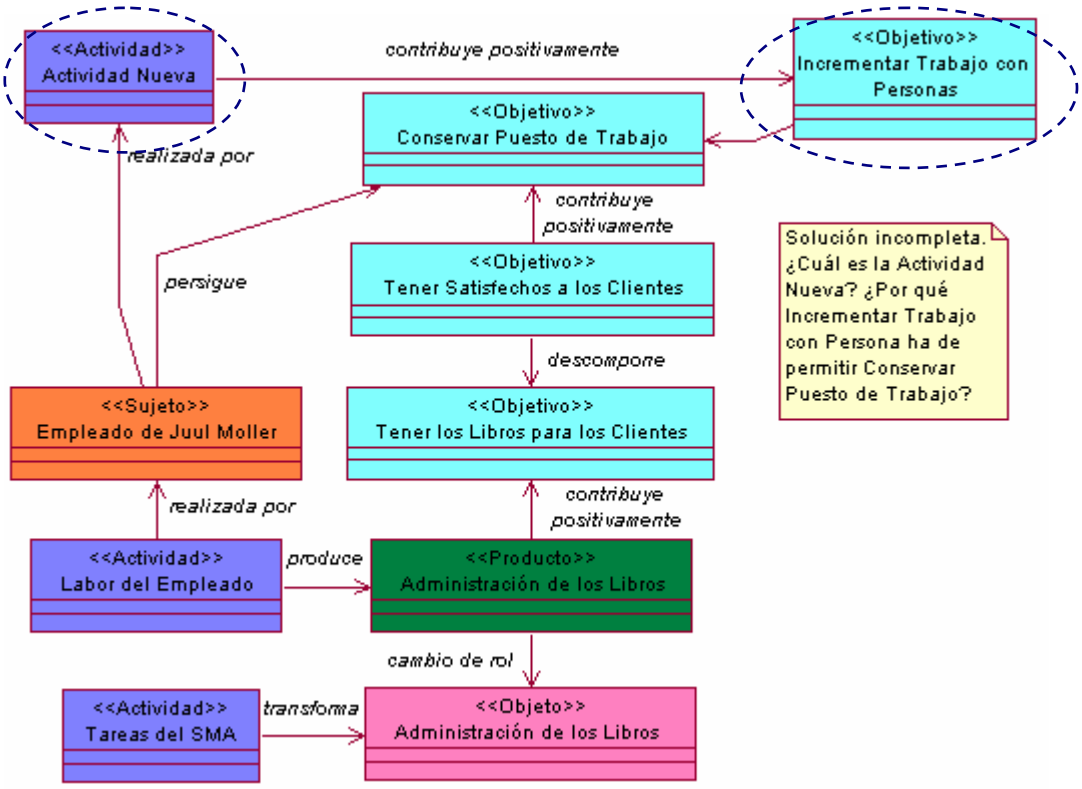


Fig. 135. Solución del *Estado de Necesidad* para los trabajadores de *Juul Møller*.

La solución propuesta no es sin embargo aplicable. La empresa propuso que los empleados cuya labor pasase a ser hecha por el SMA realizarían tareas que supondrían *Incrementar Trabajo con Personas*. Aunque también está sin perfilar cuál será la *Actividad Nueva*, el problema radica en que no se sabe cómo el *objetivo Incrementar Trabajo con Personas* contribuirá a *Conservar Puesto Trabajo*. Éste parecer ser, no tanto un error del dominio real, como una omisión en la captura de requisitos. Para obtener esta información podría usarse el *aspecto* de refinamiento de objetivos 1.7 en la GCR. El resultado se muestra en la Fig. 136.

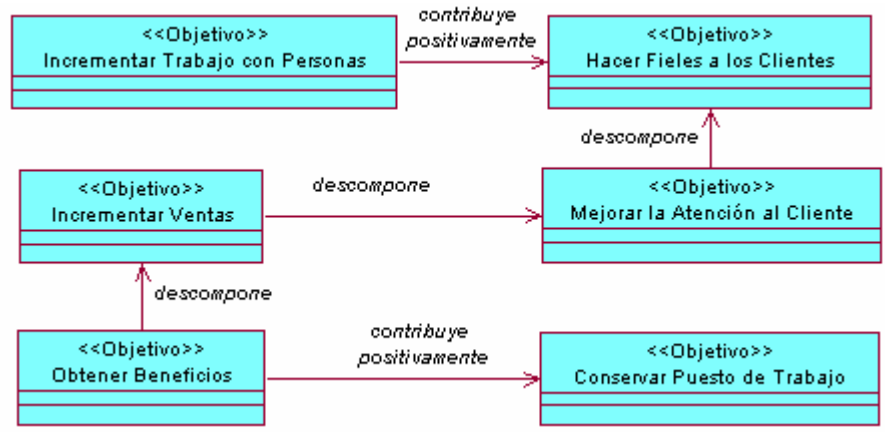


Fig. 136. Resolución de la omisión en los requisitos descubierta con el *Estado de Necesidad*.

La Fig. 136 refleja un nuevo elemento en las especificaciones previas, la relación *contribuye positivamente* entre *Incrementar Trabajo con Personas* y *Hacer Fieles a los Clientes*. Gracias a esta nueva relación se puede llevar a cabo un razonamiento sobre los *objetivos* como el ya realizado para el patrón de detección de esta contradicción (ver sección 7.4.4.1) Con esta información el patrón de solución de la Fig. 135 sería aplicable ya que *Incrementar Trabajo con Personas* sí ayudaría a los trabajadores a *Conservar Puesto de Trabajo*.

Para terminar con el *Estado de Necesidad*, señalar que éste es un caso en el que las contradicciones de la TA señalan una omisión en el análisis. Resolver esta omisión permite representar correctamente el razonamiento original del cliente sobre el trabajo de sus empleados.

7.4.5. Ayuda en la elaboración del catálogo del curso

A lo largo de la sección 7.3 de captura de requisitos (ver Fig. 113 y Fig. 119) se ha visto la influencia crítica que tiene en el buen funcionamiento del acuerdo *Juul Møller*-NSM contar a tiempo con los materiales necesarios para los cursos. Como se recordará de la introducción (ver sección 7.2) y los requisitos (ver explicación Fig. 116), contar con los libros para las clases requería que los profesores elaborasen el catálogo y que *Juul Møller* adquiriese los textos indicados en él. El catálogo es una lista de los libros que se emplearán en las distintas asignaturas. En virtud del acuerdo, *Juul Møller* recibe este catálogo unos meses antes del principio del curso, lo que le permite anticipar los pedidos. El problema radica en que el catálogo suele llegar tarde y está sujeto a cambios de última hora por los profesores. Estos cambios perjudican enormemente la estrategia de ventas de *Juul Møller*. Si se arriesga a pedir los libros con las primeras versiones del catálogo es posible que éste sufra cambios posteriores. Entonces *Juul Møller* debería devolver los libros si le es posible o bien quedarse con lotes de difícil venta, ya que no tendrían salida en la NSM. En todo caso, si retrasa sus pedidos tiene una mayor seguridad de que serán los libros utilizados en el curso. Sin

embargo, los distribuidores pueden tardar hasta dos meses en servir un pedido. Si estos plazos suceden cerca del comienzo de las clases los alumnos pueden recurrir a la competencia y *Juul Møller* perder esas ventas.

Anteriormente se trató el problema del catálogo desde el punto de vista del *Objetivo Social* del acuerdo entre *Juul Møller* y la NSM en la sección 7.4.2. Allí se incidía en hacer ver a los profesores cómo afectaban sus cambios en el catálogo a los plazos y precios de los pedidos. Se trataba de animarles a elaborar un catálogo definitivo cuánto antes y a minimizar los cambios. Esta solución ayuda a que los profesores tomen conciencia de la importancia del catálogo para contar con los materiales de los cursos a tiempo, pero no les proporciona medios para reducir esos cambios.

La Fig. 113 mostró que el principal motivo de los profesores para retrasar la salida del catálogo era *Tener Flexibilidad en los Medios del Curso*. Los profesores reciben a lo largo del año ejemplares de las novedades editoriales, algunas proporcionadas por la propia *Juul Møller*. Entre estas novedades pueden surgir libros más adecuados que los incluidos en el catálogo para las discusiones del curso. Esta situación puede ser el origen de cambios en el catálogo.

En la situación actual los profesores son los encargados de generar el catálogo y éste es usado por *Juul Møller* para adquirir los textos. Se trata de dos *sujetos* que trabajan sobre partes distintas de un mismo *objeto* (i.e. los materiales del curso), construyendo cada uno su porción sin colaborar en la construcción global. Sin embargo, la adquisición de los materiales para los cursos debiera ser una labor colaborativa, donde *Juul Møller* ayuda a los profesores a descubrir nuevo material y estos en el mismo proceso le anticipan los cambios en el catálogo.

7.4.5.1. Información Contradictoria

La situación anterior puede ser reflejada en una *Información Contradictoria*. Los *Materiales para el Curso* son un *objeto* que construyen colaborativamente los profesores y *Juul Møller*. Los primeros se ocupan de *Realizar el Catálogo del Curso* generar el catálogo del curso y *Juul Møller* de *Obtener Libros*. La creación del catálogo es un proceso de aprendizaje continúo sobre cuáles son las mejores referencias para el curso. Por otra parte, *Juul Møller* puede ir adquiriendo experiencia acerca de la clase de títulos que suelen incorporarse al catálogo en función de las novedades editoriales o de los libros que los profesores han encontrado interesantes. La situación de *Información Contradictoria*, incluyendo los *objetivos* de los *sujetos*, se refleja en la Fig. 137.

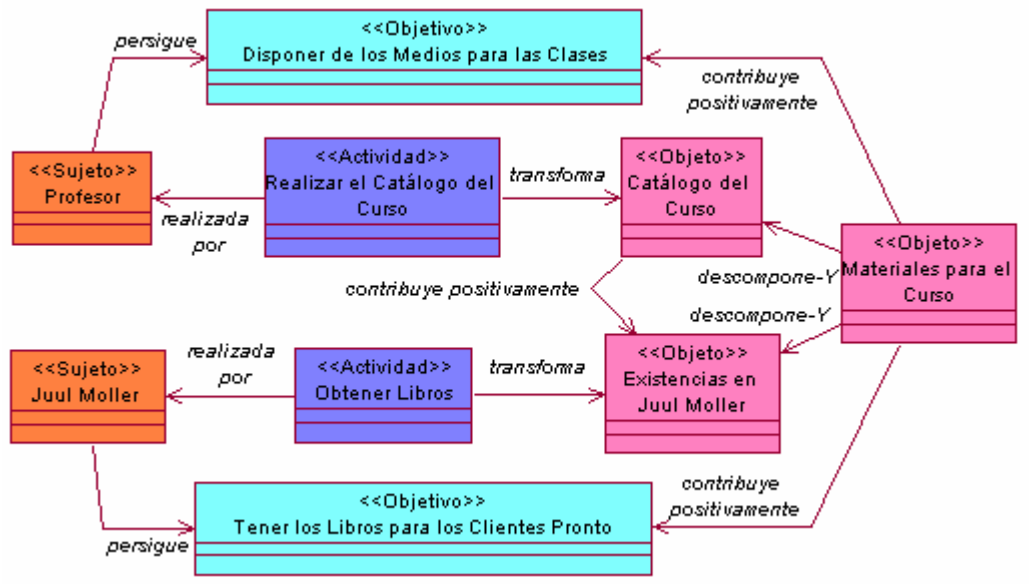


Fig. 137. *Información Contradictoria* en la elaboración de materiales para el curso.

Respeto al patrón de detección original de esta contradicción (ver Fig. 75), sólo indicar que la relación *indefinida* entre las dos partes del *objeto* construido colaborativamente ha sido instanciada por un *contribuye positivamente*. Esta sustitución no afecta al sentido de la contradicción, ya que el problema radica en la independencia de las *actividades* de construcción llevadas a cabo por cada *sujeto*.

7.4.5.2. Solución a la Información Contradictoria en el catálogo

Una opción para fomentar la construcción colaborativa de los *Materiales para el Curso* es que *Juul Møller* participe en el proceso de modificación de la bibliografía del curso. Como se ha dicho, la principal fuente de cambios es la aparición de novedades editoriales. *Juul Møller* tiene conocimiento de estas apariciones a través de sus distribuidores y puede indicárselas a los profesores. De este modo sustituye la iniciativa de los profesores por una conjunta en la que ella es parte integrante. Esta nueva forma de trabajo se representa en la Fig. 138. Ahora *Discutir Novedades Editoriales*, realizada conjuntamente por ambos *sujetos*, es la forma de anticipar cambios en el catálogo para *Juul Møller* y de descubrir nuevas referencias para los profesores.

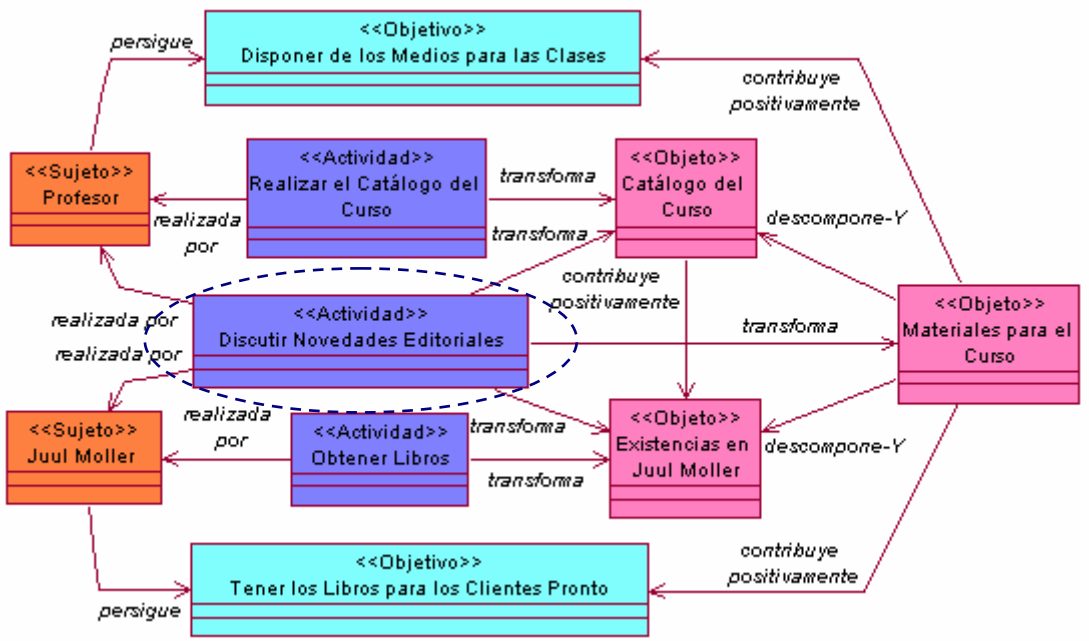


Fig. 138. Solución de la *Información Contradictoria* en la elaboración de materiales para el curso.

La nueva *actividad Reflexión sobre el Fallo* sólo es el primer elemento necesario para el proceso de diagnóstico y aprendizaje. En caso de que se deseen incluir estas capacidades, se tendrá que refinar la *actividad*, determinar sus *objetos* y *herramientas*, quizás establecer interacciones con otros *sujetos*... Esta contradicción, al igual que las restantes, da posibles soluciones de una forma simple, pero son los desarrolladores los que han de concretarlas al problema y necesidades concretos.

7.5. Evaluación del caso de estudio

El caso de estudio mostrado en las secciones previas de este capítulo pretende ilustrar cómo puede ser un desarrollo software de SMAs usando las herramientas de la TA. Aunque el desarrollo completo no se recoge debido a su extensión, esta sección pretende resumir algunos datos acerca del proceso total.

La Tabla 20 recoge estadísticas acerca de las especificaciones del SMA para *Juul Møller*. Como se señaló en la introducción de este capítulo, el SMA ha sido construido con la metodología INGENIAS [Pavón & Gómez-Sanz 2003]. INGENIAS adapta el USDP [Jacobson *et al.* 1999] a SMAs aunque no contempla la fase de captura de requisitos que delega en otras técnicas. Por tanto, los datos recogidos acerca de la captura de requisitos corresponden a las técnicas basadas en la TA. No obstante, cuando se consideró necesario en esta fase, se usó la traducción de parte de las especificaciones a INGENIAS para aclarar aspectos de las especificaciones del

SMA. Al finalizar la captura de requisitos, se realizó la traducción de las especificaciones a INGENIAS. Esta traducción sirvió como punto de partida del resto del desarrollo. Por supuesto, en el resto del proceso se siguió usando la GCR pero estos diagramas ya no están contabilizados en estas estadísticas.

El proceso de desarrollo seguido es el de INGENIAS con la guía de las técnicas de la TA. Al tratarse de las especificaciones del sistema sólo se contemplan las fases de *Inicio* y *Elaboración* de cada flujo de trabajo. En cada uno de estos flujos no se contabiliza el tamaño global de las especificaciones sino sólo los nuevos elementos introducidos en las mismas. Además sólo se cuentan las entidades y relaciones diferentes. Es decir, múltiples apariciones de un mismo elemento en varios diagramas sólo cuentan una vez.

Flujos de Trabajo	Fase	Diagramas	Entidades	Relaciones
Requisitos	Inicio	13	43	54
	Elaboración	15	28	43
Análisis	Inicio	15	16	28
	Elaboración	16	16	32
Diseño	Inicio	8	26	29
	Elaboración	21	31	38
Total		88	160	224

Tabla 20. Cifras de la especificación de *Juul Møller Bokhandel A/S.*

La Tabla 20 arroja algunos datos llamativos. Los resultados en la fase de requisitos presentan un elevado número de componentes, superior en muchos casos a los de fases posteriores. La explicación a este fenómeno es doble: el estudio del entorno en los requisitos; la fragmentación de las respuestas a las *preguntas* de la GCR. Por un lado los requisitos prestan mucha atención al contexto del futuro SMA. En ellos se contemplan la organización que rodea al sistema y sus reglas. A medida que avanza el proceso de desarrollo, estos elementos van perdiendo visibilidad, ya que quedan subsumidos implícitamente en el SMA. El propio SMA incorpora en sus componentes y estructura la influencia de la organización externa. El segundo factor que determina el elevado número de componentes es la fragmentación de la GCR. Ésta se compone de un gran número de *preguntas*, muchas de las cuales recogen sólo pequeñas porciones de información. Un desarrollador genera normalmente diagramas de mayor tamaño que corresponden a la fusión de las respuestas a varias *preguntas*. También sucede que en el análisis de los requisitos se responden *preguntas* que luego no resultan relevantes en el resto del desarrollo. Esto también origina elementos que aumentan las cifras de esta fase y sin embargo no tienen reflejo en el análisis y diseño.

Otro aspecto a considerar acerca del caso de estudio es qué componentes de la GCR han sido utilizados y en qué medida. La GCR cubre un amplio espectro de *áreas* de introspección que a su vez incluyen numerosos *aspectos*. La versión para SMAs surge de la experiencia en el desarrollo de SMAs y el trabajo con la TA. Su formulación deberá evolucionar a medida que se vaya obteniendo nueva experiencia. Por otra parte, los diferentes proyectos inciden en características distintas, por lo que se centran en diferentes partes de la GCR. El estudio de este uso podrá permitir en el futuro establecer versiones específicas de la GCR para ciertos dominios.

El caso de estudio ha permitido sacar algunas conclusiones acerca del uso de la GCR y de los resultados de su uso.

En primer lugar se observa que no todos los *aspectos* tienen el mismo uso. En algunos casos, especialmente los de las *áreas* de *Aprendizaje/cognición/articulación* y *Desarrollo*, el motivo de este menor uso puede estar en la clase de desarrollo realizado. Se trata de un SMA nuevo donde no se ha abordado con profundidad la evolución futura del sistema. Hay otros aspectos que sin embargo son usados por encima de los demás. Estos aspectos hacen referencia a características centrales del paradigma de agentes, como los *aspectos* 1.2, 1.7 y 1.8, o de la TA, por ejemplo los aspectos 1.13 y 1.16.

Un segundo punto de atención es el resultado de responder las *preguntas*. En general se logra rellenar la forma UML-TA en un elevado número de ocasiones. No obstante, la información recogida no es relevante para el desarrollo en ciertos casos, quizás por corresponder a conocimiento sobre el entorno que no es utilizada después o a una opción de diseño que no es elegida.

La otra técnica de la TA que se ha considerado en este caso de estudio es el uso de contradicciones. También aquí se han extraído varias conclusiones.

La primera es que los patrones más simples se detectan un número de veces alto, dando lugar a "falsos positivos". Éste es el caso de las contradicciones de *Significado Dual*, *Interrupción de Uso* y *Paradoja de Planificación*. Se trata de patrones que incluyen pocos elementos y que pueden identificarse con numerosas configuraciones dentro de las especificaciones. Ello conduce a que muchas de las detecciones carezcan de significado como contradicciones reales en las especificaciones. Mantener la formulación actual de las contradicciones exige proporcionar filtros que permitan al usuario descartar *a priori* buena parte de las identificaciones no relevantes. Entre estos filtros se encuentran la posibilidad de incluir determinadas entidades y relaciones en el patrón de detección de la contradicción o que la explicación de la contradicción sobre los modelos incluya un número mínimo de elementos. No obstante, es de esperar que un mayor refinamiento en los patrones de detección y en el vocabulario empleado para definirlos pueda proporcionar identificaciones más precisas.

La segunda conclusión apunta que se trata de una técnica que ha permitido detectar un elevado número de conflictos en las especificaciones. Las contradicciones tenían su origen sobre todo en dos tipos de situaciones: las múltiples apariciones de un elemento en diferentes diagramas, con cadenas de asociaciones inadvertidas; configuraciones del sistema que no contemplaban situaciones a las que podían dar lugar y que debían ser especificadas. Sobre las identificaciones hay que decir también que éstas han de ser valoradas por el usuario de la técnica. Sólo una persona con conocimiento sobre el dominio real del problema puede determinar qué situaciones constituyen verdaderas contradicciones.

Por último decir que para las identificaciones de contradicciones relevantes, la propuesta de solución asociada a la contradicción sirvió en un porcentaje muy elevado (superior a un tercio) para eliminar el problema de las especificaciones. De esta forma el desarrollador puede solucionar fácilmente los problemas de menor complejidad y dedicar su atención a aquellos que requieren en verdad una solución innovadora. En todo caso, un mayor número de posibles soluciones por patrón detección contribuiría a aumentar este ratio de éxito.

7.6. Conclusiones

A lo largo de este capítulo se ha expuesto un caso práctico de cómo llevar a cabo la especificación de SMAs con el soporte de técnicas basadas en la TA. Este caso de estudio ha mostrado algunas de las diferencias clave en cuanto a las aportaciones de la TA como soporte de la ISOA respecto a otras técnicas.

En el aspecto positivo cabe citar que el uso de las técnicas de la TA permite a los desarrolladores mejorar la toma de decisiones sobre la evolución de las especificaciones y localizar contradicciones sociales.

La GCR permite recoger de forma sistemática los aspectos del SMA que han de ser tratados. Las *preguntas* engarzan y cruzan datos de tal modo que se obtiene nuevos elementos y vistas diferentes de los ya contemplados. En este sentido, la GCR constituye un recordatorio de aspectos a tratar, suministra posibles alternativas en las especificaciones y dirige la introspección a nuevos puntos del sistema. Además, su formulación mediante estructuras de *propiedades sociales* indica también cómo modelar la información capturada.

Las técnicas basadas en la TA y su experiencia investigadora también permiten aprovechar las componentes sociales e intencionales de los SMAs para encontrar puntos conflictivos en las especificaciones, elaborar explicaciones sobre el conflicto y proponer soluciones. Estas tareas se realizan mediante la aplicación de las contradicciones al estudio de las especificaciones. Las contradicciones han indicado aspectos de las especificaciones que necesitaban mayor elaboración y han sugerido la forma de llevar a cabo su modificación.

Además, frente a otras propuestas que confían la existencia de este conocimiento sobre captura de información y contradicciones sociales a la experiencia de los desarrolladores, las técnicas basadas en la TA se sirven de los casos de estudio ya generados en las ciencias sociales como conocimiento experto. De esta forma los desarrolladores disponen de librerías de conocimiento listas para el uso e independientes de su propio conocimiento.

La propuesta también ha mostrado algunas dificultades en el caso de uso: excesiva fragmentación de los diagramas; sobrecarga de información; falta de conocimiento sobre el significado en el dominio real de los elementos modelados.

La fragmentación de los diagramas afecta principalmente a los resultados de aplicar las *preguntas* de la GCR. Su formulación en UML-TA consiste generalmente en diagramas con muy pocos elementos. Al contestar las *preguntas* para los distintos aspectos del SMA se generan muchos diagramas de tamaño reducido: la comprensión de la información se vería facilitada si varios de estos diagramas se fusionasen en otros de mayor tamaño. Ésta ha sido la solución adoptada a lo largo de este caso de estudio. Por ejemplo, muchas de las figuras donde aparecían relacionados los objetivos del sistema y su entorno eran el resultado de sintetizar las respuestas de múltiples *preguntas*.

El segundo inconveniente es la gran cantidad de información generada al proceder con estos métodos.

En el caso de la GCR existe una tendencia a proceder exhaustivamente con las *preguntas* para todos los elementos del SMA donde sean aplicables. Aunque este proceder permite una visión muy completa del sistema y su entorno, ha de quedar limitado por la utilidad de la información recogida. Esta utilidad depende de su uso en el posterior desarrollo.

Para las contradicciones el problema está en que hay demasiadas identificaciones de los patrones de detección en las especificaciones. Incluso los patrones más complejos pueden presentarse varias decenas de veces en unos modelos. Por supuesto, las identificaciones no siempre son relevantes en un desarrollo, ya que su significado en el dominio real no depende sólo de la estructura del patrón sino también de la semántica de los elementos del modelo involucrados. En la herramienta software implementada para estas técnicas, se ha mejorado el tratamiento de las detecciones mediante una serie de filtros que minimizan los resultados no significativos según los intereses del usuario.

La última limitación emerge del vocabulario manejado. Este trabajo ha optado por mantener un conjunto reducido de primitivas a fin de facilitar el uso de estos métodos y los procesos automáticos. La contrapartida es que los conceptos de la TA son muy genéricos y en ocasiones se hace difícil expresar ciertos matices con ellos. Esto genera una carencia de conocimiento sobre el significado real de algunos elementos modelados. Una mayor precisión en el vocabulario permitiría describir con más detalle los requisitos o reducir el número de falsas identificaciones de contradicciones. Como en muchos casos en la Ingeniería del Software, se trata de alcanzar un equilibrio adecuado a los intereses del desarrollo entre capacidad expresiva, de análisis y facilidad de uso.

Para terminar hay que señalar que otra dificultad es la preparación de las correspondencias para la traducción entre los conceptos de TA y los del vocabulario para SMAs. Como ya se señaló al presentar estas correspondencias, se trata de una labor que requiere conocimientos de los conceptos de la TA y SMAs tal y como se manejan en el problema concreto. La definición de estos conceptos admite algunas interpretaciones por lo que pueden requerir cierta adaptación con el uso. Este problema no se ha visto en este ejemplo porque se han usado las correspondencias de un capítulo anterior,

A modo de conclusión se puede afirmar que las herramientas basadas en la TA son útiles para la guía del desarrollo y la mejora de las especificaciones en relación con los aspectos sociales e intencionales de los SMAs. Su principal inconveniente es la enorme cantidad de información que puede generar su uso indiscriminado.

Capítulo 8. Conclusiones finales

En este capítulo final se recogen las principales conclusiones de la memoria: las aportaciones originales de este trabajo, las próximas líneas de investigación y el trabajo futuro.

8.1. Aportaciones

La aportación principal de esta tesis ha sido un conjunto de métodos basados en la TA para analizar los aspectos sociales de los SMAs. Estos métodos y la infraestructura para su uso incluyen un lenguaje de modelado para la TA (i.e. UML-TA), estructuras para la representación de las *propiedades sociales* de los SMAs, un método de traducción de información entre la TA y las metodologías de SMAs, una Guía de Captura de Requisitos sociales en SMAs (i.e. la GCR), un proceso de guía del desarrollo basado en contradicciones, librerías de *propiedades sociales* sobre requisitos y contradicciones fundadas en la experiencia de la TA y un asistente software de la TA para aplicar estos métodos (i.e. el ATA).

La hipótesis de este trabajo fue que el tratamiento de lo que se han denominado *propiedades sociales* (i.e. aspectos tales como organización, evolución, confianza, motivación o aprendizaje) de los SMAs estaba perjudicado por el uso de soluciones fragmentadas. La mayor parte de las propuestas de la ISOA no contemplaban todas estas propiedades o lo hacían aisladamente, y en todo caso existía muy poco soporte específico para su tratamiento. Esto hacía que la captura de información sobre estas propiedades, su desarrollo y comprobación fueran muy complejas en la ISOA y dependieran en buena medida del conocimiento de los desarrolladores. Se necesitaban por tanto técnicas que permitieran abordar el trabajo de las *propiedades sociales* de una forma global, con conocimiento de soporte y que fueran aplicables desde una perspectiva propia de la Ingeniería del Software.

Este razonamiento condujo a considerar las teorías de Sociología y Psicología como una posible base para el análisis de las características sociales de los SMAs. La justificación de esta adecuación se basa en las características sociales e intencionales comunes a los objetos de estudio de las ciencias sociales y la ISOA, respectivamente las organizaciones humanas y los SMAs. Entre las teorías sociales se seleccionó la TA. Su insistencia en un análisis sistémico de la sociedad, en contemplar la perspectiva social, cultural y temporal, y su estudio de las contradicciones como parte integrante de la evolución del sistema la hacían una candidata adecuada para ayudar en un proceso software de la ISOA.

El desarrollo de los procesos para aplicar la TA en la ISOA llevó a las aportaciones vistas inicialmente y que se repasan a continuación.

El **lenguaje UML-TA** se emplea para el modelado gráfico de las *propiedades sociales*. Se siguió un ciclo de interpretación, síntesis y descripción de los principales conceptos de la TA en UML. El resultado fue la especificación del lenguaje UML-TA. UML-TA permite transmitir las ideas de la TA en una forma menos ambigua que el lenguaje natural usado en las disciplinas sociales y la integra naturalmente en los procesos de desarrollo software. Además, este lenguaje sentó las bases de la conexión genérica de la TA con las metodologías de agentes.

Las **estructuras de representación de las *propiedades sociales*** son los contenedores de propiedades individuales de la TA. Esta representación incluye una descripción textual de cada propiedad, patrones de detección para localizarla en especificaciones descritas con la TA y patrones de solución que describen posibles mejoras en los modelos cuando se detecta la propiedad. Cada patrón incluye dos representaciones: una textual y otra con UML-TA. La textual sirve de explicación sobre el patrón. La forma UML-TA es un marco con ranuras editables que recoge la información de una propiedad concreta para el desarrollo y que sirve de base para los procesos semi-automáticos posteriores.

El **proceso de traducción** se emplea para permitir la portabilidad de las técnicas de la TA sobre diferentes metodologías de la ISOA. El propósito planteado inicialmente en esta investigación era proporcionar un conjunto de métodos aplicables a las metodologías existentes. En este punto se resolvió que la conexión se haría estableciendo correspondencias entre los estructuras de la TA y de las metodologías de SMAs. Ello permitiría la traducción bidireccional de especificaciones entre ambos lenguajes. De esta manera, tanto los desarrolladores como los usuarios podrían trabajar con el lenguaje que prefiriesen y usar las técnicas de ayuda de la TA.

La **GCR** es el método basado en la TA para la captura de requisitos. Se estructura en una serie de *áreas*, *aspectos* y *preguntas* predefinidas que inciden en los aspectos sociales del SMA a diseñar. Las *áreas* y *aspectos* describen textualmente su contenido mientras que las *preguntas* se representan con las estructuras para *propiedades sociales*. La representación UML-TA de las *preguntas* actúa como una plantilla para capturar los datos en las respuestas de los usuarios. Esta información era la base de los requisitos del sistema.

La GCR para SMAs cubre un hueco en la captura de los requisitos sociales para agentes. Establece una guía para la captura de requisitos con un lenguaje social, una doble representación para él (textual y UML) y reglas para la traducción entre ambos. De esta forma clientes y desarrolladores pueden trabajar en lenguaje natural con un vocabulario común cuya traducción al lenguaje de modelado de software es orientada por la propia técnica. Así se facilita la comunicación y se reducen los errores de interpretación en esta fase.

Las **contradicciones de la TA** son el método de análisis de las especificaciones. Las contradicciones también se representan con la estructura para *propiedades sociales*. Estas contradicciones se extraen de los estudios de la TA y constituyen una librería de situaciones conflictivas en las sociedades. Esta clase de contradicciones sociales suelen ser difíciles de determinar ya que no surgen por propiedades inherentes a los artefactos de modelado, sino por el significado de estas abstracciones en sus organizaciones y las relaciones entre ellas.

La detección de contradicciones indica los puntos que pueden necesitar mayor desarrollo en las especificaciones. El proceso indica el posible problema social,

explica porque puede serlo y suministra alternativas para cambiar los modelos. Los desarrolladores cuentan de esta manera con una herramienta precisa de ayuda a la toma de decisiones que puede guiar las iteraciones sobre sus modelos.

Finalmente, esta tesis ha generado una herramienta, **el asistente de TA o ATA**, que implementa los métodos previos. Se ha creado una arquitectura genérica, configurable con ficheros XML, que verifica propiedades definidas como patrones estructurales en UML-TA sobre cualquier especificación para la que disponga de correspondencias de traducción. Además cuenta con interfaces para conectarlo a herramientas de modelado existentes y para aplicar filtros y métricas de evaluación sobre los resultados que obtiene.

8.2. Líneas de investigación abiertas y trabajo futuro

La principal dificultad de las técnicas expuestas es la interpretación y traducción del conocimiento de la TA. Esta dificultad se plasma en el modelado de la información obtenida en términos del lenguaje UML-TA y la traducción de este lenguaje al vocabulario de una metodología SMAs dada. Este trabajo ha establecido pautas y procesos para realizar ambos pasos, aunque dependen en cierta medida de la interpretación personal.

Otra mejora que es necesario abordar es la cobertura de la GCR. Su primera versión contempla características relevantes de los requisitos de un SMA para los cuales se han creado *preguntas*. Sin embargo es necesario completar su definición actual y combinarla con otras herramientas de la Ingeniería de Requisitos. Los requisitos de un SMA son demasiado ricos y complejos como para suponer que son cubiertos con un solo método.

Respecto al vocabulario manejado en la actualidad para modelar los conceptos de la TA, ha sido definido atendiendo a las necesidades que han surgido en el presente trabajo. El vocabulario es un reflejo de los conceptos y relaciones que han sido usados para modelar la GCR y las contradicciones. La futura experimentación en estas áreas ampliará las necesidades. Entonces será necesario realizar modificaciones en el lenguaje. Estas modificaciones posiblemente impliquen una revisión de las correspondencias actuales para hacer más precisos algunos significados.

A propósito del vocabulario de modelado y las traducciones hay que validar el uso del metalenguaje GOPRR. En la implementación de la herramienta se optó por el uso de GOPRR para especificar UML-TA, los lenguajes de las metodologías de la ISOA y las correspondencias. En ese momento se argumentó que GOPRR tenía la suficiente capacidad expresiva para especificar la mayor parte de los lenguajes gráficos y formalismos. Queda pendiente comprobar la extensión real de esta capacidad expresiva y encontrar los posibles lenguajes que no puedieran ser representados con GOPRR.

Además hay que ampliar la librería de contradicciones. Esta tesis ha trabajado con diez contradicciones de la TA. La literatura sobre el tema contiene no obstante más patrones con situaciones que sería interesante comprobar en un SMA. Esta investigación no tiene por qué limitarse a la TA. La experiencia ganada con este

estudio hace que se pueda considerar la ampliación de la aproximación hacia nuevas teorías sociales que cubran aspectos no considerados de los SMAs.

Otro punto es la adaptación de la GCR a dominios específicos de aplicación. Un SMA contempla muchos aspectos que deben ser modelados. Crear una única guía que pretenda cubrirlos todos llevaría posiblemente a una versión poco operativa de la misma por su tamaño. En su lugar se plantea disponer de versiones específicas para aspectos concretos de un SMA que permitan abordarlos con mayor profundidad. De esta manera, la actual GCR sería reemplazada por una librería de GCRs. Un desarrollo concreto usaría únicamente aquellas que fueran relevantes para su aplicación.

Finalmente, estas propuestas han sido validadas con experimentos en el seno de nuestro grupo de investigación. Queda pendiente una experimentación más amplia con otros proyectos y dominios de aplicación. En este sentido, la integración con una herramienta de modelado como el *INGENIAS Development Kit* da una amplia base de usuarios potenciales para realizar estos experimentos.

Por último, y como línea de investigación abierta a más largo plazo, se plantea el estudio de si estas técnicas, convenientemente modificadas, son exportables a otros paradigmas, por ejemplo a la OO. El problema de considerar el entorno social no es exclusivo de la ISOA, sino que aparece en cualquier desarrollo software. Aunque la conexión con la TA no sería tan clara en otros paradigmas, los beneficios de su uso serían similares a los que aporta a la ISOA y también se trata de una aproximación inexplorada.

8.3. Publicaciones relacionadas

A lo largo de la investigación que ha culminado con esta tesis y fruto de los trabajos que durante este periodo se han ido realizando, se han producido una serie de publicaciones que se detallan a continuación en orden cronológico:

1. **Lenguaje UML-TA, correspondecias y contradicciones**. [Fuentes *et al.* 2003a] R. Fuentes, J.J. Gómez-Sanz, J. Pavón: *Activity Theory for the Analysis and Design of Multi-Agent Systems*. In Proceedings of the 4th International Workshop on Agent Oriented Software Engineering (AOSE 2003), Melbourne, Australia, July 2003. Volume 2935 of Lecture Notes in Computer Science, pages 110–122. Springer Verlag. 2003.
2. **Lenguaje UML-TA, correspondecias y contradicciones**. [Fuentes *et al.* 2003b] R. Fuentes, J.J. Gómez-Sanz, J. Pavón: *Social Analysis of Multi-Agent Systems with Activity Theory*. In Proceedings of CAEPIA 2003, San Sebastian, Spain, November 2003. Volume 3040 of Lecture Notes in Artificial Intelligence pages 526-535. Springer Verlag. 2004.
3. **Guía de Captura de Requisitos**. [Fuentes *et al.* 2004a] R. Fuentes, J.J. Gómez-Sanz, J. Pavón: *Towards Requirements Elicitation in Multi-Agent Systems*. In Proceedings of the 4th International Symposium From Agent Theory to Agent Implementation, AT2AI 2004, pages 582-587, Vienna, Austria, April 2004.
4. **Guía de Captura de Requisitos**. [Fuentes *et al.* 2004b] Rubén Fuentes, Jorge Gómez-Sanz, Juan Pavón, Lorna Uden: *Activity Theory Applied to Requirements Elicitation of Multi-Agent Systems*. In Proceedings of the 1st International Workshop on Activity Theory Based Practical Methods for IT Design, ATIT 2004, Copenhagen, Denmark, September 2004. 12 pages. To appear.
5. **Casos de estudio**. [Fuentes *et al.* 2004c] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Captura del Entorno Social de Sistemas Multiagente*. In Proceedings of the 3rd International Workshop on Practical Applications of Agents and Multi-Agent Systems, IWPAAMS 2004, Burgos, Spain, October 2004. 10 pages. To appear.
6. **Estado del arte**. [Fuentes *et al.* 2004d] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Técnicas de Verificación y Validación para Sistemas Multi-Agente*. Novatica 170, 5 pages, Julio-Agosto 2004.
7. **Contradicciones**. [Fuentes *et al.* 2004e] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Managing Conflicts between Individuals and Societies in Multi-Agent Systems*. In Proceedings of the 5th International Workshop Engineering Societies in the Agents World, ESAW'04, Toulouse, France, October 2004. To appear in Lecture Notes in Artificial Intelligence, 12 pages. Springer Verlag.
8. **Propiedades sociales**. [Fuentes *et al.* 2004f] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *A Sociological Framework for Multi-Agent Systems Verification and Validation*. In Proceedings of the International Workshop on COnceptual MOdelling for Agents, CoMoA 2004, Shanghai, China, November 2004. To appear in Volume 3289 of Lecture Notes in Computer Science, pages 458-469. Springer Verlag.
9. **Contradicciones**. [Fuentes *et al.* 2004g] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Checking Social Properties of Multi-Agent Systems with Activity Theory*. In Proceedings of the 9th Ibero-American Conference on Artificial Intelligence, Iberamia 2004, Puebla, Mexico, November 2004. To appear in Volume 3315 of Lecture Notes in Artificial Intelligence, pages 1–11. Springer Verlag

Bibliografía

[Barab *et al.* 2002] Sasha A. Barab, Michael Barnett, Lisa Yamagata-Lynch, Kurt Squire, Thomas Keating: *Using Activity Theory to Understand the Systemic Tension Characterizing a Technology-Rich Introductory Astronomy Course*. Mind, Culture, and Activity, 9 (2), pp. 76-107, 2002.

[Bardram 1997] Bardram, J. E.: *Plans as Situated Action: An Activity Approach to Workflow Systems*. Hughes et al. (eds.): In Proceedings of the Fifth European Conference on Computer-Supported Cooperative Work, ECSCW'97, Kluwer, pp. 17-32. 1997.

[Barley 1986] Barley, S.R.: *Technology as an occasion for structuring: Evidence from observations of CT Scanners and the social order of radiology departments*. Administrative Science Quarterly, 31, pages 78-108. 1986.

[Barros & Verdejo 2000] B. Barros, M.F. Verdejo: *DEGREE: un sistema para la realización y evaluación de experiencias de aprendizaje colaborativo en enseñanza a distancia*. Revista Iberoamericana de Inteligencia Artificial Vol 9 ISSN 1137-3601. Pages 27-37. 2000.

[Bateson 1972] Bateson, G.: *Steps to an ecology of mind*. New York: Ballantine Books. 1972.

[Bauer *et al.* 2001] Bauer, B., Müller, J. P., Odell, J.: *Agent UML: A Formalism for Specifying Multiagent Interaction*. International Journal of Software Engineering and Knowledge Engineering (IJSEKE), vol. 11, no. 3, 2001.

[Bednyi & Meister 1997] Bednyi, G. Z., Meister, D.: *The Russian Theory of Activity: Current Application to Design and Learning*. Lawrence Erlbaum Associates. 1997.

[Bellifemine *et al.* 2001] Bellifemine, F., Poggi, A., Rimassa, G.: *JADE: a FIPA2000 compliant agent development environment*. In Proceedings of the 5th international conference on Autonomous agents (pp. 216--217): ACM Press. 2001.

[Bérard *et al.* 2001] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen: *Systems and Software Verification. Model-Checking Techniques and Tools*. Springer. 2001.

[Bernon *et al.* 2002] Carole Bernon, Marie-Pierre Gleizes, Sylvain Peyruqueou Gauthier Picard: *Adelfe, a methodology for adaptive multi-agent systems engineering*. In Proceedings of the "Engineering Societies in the Agents World" workshop. 2002.

[Bertelsen 1996] Olav W. Bertelsen: *Elements of a Theory of Design Artefacts, a contribution to critical systems development research*. Ph. D. Thesis. Department of Information and Media Science. Aarhus University. 1996.

[Bertrand *et al.* 1998] P. Bertrand, R. Darimont, E. Delor, P. Massonet, A. van Lamsweerde: *GRAIL/KAOS: an environment for goal drivent requirements engineering*. In Proceedings ICSE'98 - 20th International Conference on Software Engineering, IEEE-ACM, Kyoto, April 1998.

[Bisgaard *et al.* 1989] Bisgaard, O., Mogensen, P., Nørby, M., Thomsen, M.: *Systemudvikling som lærevirksomhed, konflikter som basis for organisationel udvikling* [Systems development as a learning activity, conflicts as the origin of organizational development]. DAIMI IR-88. Århus: University of Aarhus. 1989.

[Bødker & Grønbæk 1996] Bødker, S., Grønbæk, K.: *Users and Designers in Mutual Activity: An Analysis of Cooperative Activities in System Design*. In Y. Engeström and D. Middleton, eds., Cognition and Communication at Work, 130-158. Cambridge, England: Cambridge University Press. 1996.

[Bødker *et al.* 1998] Bødker, S., Knudsen, J., Kyng, M., Ehn, P., Madsen, K.H.: *Computer Support For Cooperative Design*. In Proceedings of ACM CSCW'88 Conference on Computer-Supported Cooperative Work (Portland, OR). 1988.

[Booch 1994] Grady Booch: *Object-Oriented Analysis and Design with Applications*. Addison-Wesley. 1994.

[Bratman 1987] Bratman, M. E.: *Intentions, Plans, and Practical Reason*. Harvard University Press. 1987.

[Bratus & Lishin 1983] Bratus, B. S., Lishin, O. V.: *Laws of the development of activity and problems in the psychological and pedagogical shaping of the personality*. Soviet Psychology XXI, 38-50. 1983.

[Brazier *et al.* 1994] Brazier, F. M. T., van Langen, P., Treur, J., Wijngaards, N., Willems, M.: *Modelling a design task in DESIRE: the VT example*. Technical Report IR-377, Universiteit Amsterdam, Department of Mathematics and Computer Science, Vrije, Amsterdam, the Netherlands. 1994.

[Brazier *et al.* 1997] Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R., and Treur, J.: *DESIRE:Modelling Multi-Agent Systems in a Compositional Formal Framework*. International Journal of Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, 1997.

[Brooks 1987] F.P. Brooks: *No Silver Bullet: Essence and Accidents of Software Engineering*. IEEE Computer, Vol. 20 No. 4, April 1987, pp. 10-19.

[Brooks 1975] F.P. Brooks: *The Mythical Man-Month*. Addison-Wesley, Reading, MA, 1975.

[Caire *et al.* 2001] Caire, G., Leal, F., Chainho, P., Evans, R., Garijo, F., Gomez-Sanz, J. J., Pavon, J., Kerney, P., Stark, J. y Massonet, P.: *Agent Oriented Analysis using MESSAGE/UML*. Actas de conferencia. Springer Verlag. LNCS 2222. pp. 119-135. 2001.

[Castelfranchi 2000] Cristiano Castelfranchi: *Engineering Social Order*. In Proceedings of ESAW 2000, Berlin, Germany, August 2000. In Omicini A., Tolksdorf R., Zambonelli F., *Engeneering Societies in the Agents World* (pp. 1-18). Springer.

[Castro *et al.* 2001] Jaelson Castro, Manuel Kolp, John Mylopoulos: *A Requirements-Driven Development Methodology*. In Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), Interlaken, Switzerland, June 4-8, 2001.

[Cimatti *et al.* 2000] A. Cimatti, E. M. Clarke, F. Giunchiglia, M. Roveri: *NuSMV: A New Symbolic Model Checker*. International Journal on Software Tools for Technology Transfer, 2(4):410–425. 2000.

[Constantine 1995] L. L. Constantine: *Constantine on Peopleware*. Englewood Cliffs, NJ: Yourdon Press, 1995.

[Dardenne *et al.* 1993] Anne Dardenne, Axel van Lamsweerde, Stephen Fickas: *Goal-directed Requirements Acquisition*. Science of Computer Programming, Vol. 20, 1993, p. 3-50.

[Davydov 1988] Davydov, V. V.: *Problems of developmental teaching: The experience of theoretical and empirical psychological research*. Soviet Education, Part I: 30 (8), 15-97; Part II: 30 (9), 3-38; Part III: 30 (10), 3-77. 1988.

[DeLoach 2001] DeLoach, S.: *Analysis and Design using MaSE and agentTool*. In proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conferece (MAICS). 2001.

[Demazeau 1995] Demazeau, Y.: *From cognitive interactions to collective behaviour in agent-based systems*. In Proceedings of the 1st European Conference on Cognitive Science, pp. 117-132, Saint-Malo. April 1995.

[Dubois *et al.* 1994a] Dubois, E. Du Bois, P. Dubru, F. and Petit, M.: *Agent-oriented requirements engineering: A case study using the albert language*. In Proceedings of the 4th International Working Conference on Dynamic Modelling and Information Systems (DYNMOD'94), pages 205–238. 1994.

[Dubois *et al.* 1994b] Eric Dubois, Philippe Du Bois, Frédéric Dubru: *Animating formal requirements specifications of cooperative information systems*. In Proceedings of the 2nd International Conference on Cooperative Information Systems (CoopIS-94), pp. 101–112, Toronto (Canada), May 17-20, 1994.

[El'konin 1977] El'konin, D. B.: *Toward the problem of stages in the mental development of the child*. In M. Cole (ed.) Soviet developmental psychology. White Plains, N.Y.: Sharpe (pp. 538-563). 1977.

[Ellen 1984] Ellen, R. F. ,ed.: *Ethnographic Research: A Guide to General Conduct*. Academic Press, London. 1984.

[Elmasri & Navathe 1994] Ramez Elmasri, Shamkant B. Navathe: *Fundamentals of database systems*. Addison-Wesley. 1994.

[Engeström 1987] Yuri Engeström: *Learning by expanding*. Orientakonsultit, Helsinki. 1987.

[Engeström 1990] Yuri Engeström: *Constructing the object in the work activity of primary care physicians*. In Engeström. Y. *Learning, working and imagining. Twelve studies in activity theory*. Orienta-Konsultit OY. Helsinki, pp. 106-129. 1990.

[Engeström & Middleton 1998] Yuri Engeström, David Middleton: *Cognition and communication at work*. Yuri Engeström and David Middleton editors. Cambridge University Press. 1998.

[Espen 2001] Espen Andersen: *Juul Møller Bokhandel A/S*. Case Study available at http://www.espen.com/. 2001.

[Fensel 1995] D. Fensel: *Formal Specification Languages in Knowledge and Software Engineering*. The Knowledge Eng. Rev., vol. 10, no. 4, pp. 361-404. 1995.

[Ferber 1999] Ferber, J.: *Multi-Agent Systems*. Addison-Wesley. 1999.

[Fjeld *et al.* 2002] M. Fjeld, K. Lauche, M. Bichsel, F. Voorhorst, H. Krueger, M. Rauterberg: *Physical and Virtual Tools: Activity Theory Applied to the Design of Groupware*. In B. A. Nardi & D. F. Redmiles (eds.) A Special Issue of *Computer Supported Cooperative Work (CSCW)*: Activity Theory and the Practice of Design, Volume 11 (1-2), pp. 153-180. 2002.

[Fuentes *et al.* 2003a] R. Fuentes, J.J. Gómez-Sanz, J. Pavón: *Activity Theory for the Analysis and Design of Multi-Agent Systems*. In Proceedings of the 4th International Workshop on Agent Oriented Software Engineering (AOSE 2003), Melbourne, Australia, July 2003. Volume 2935 of Lecture Notes in Computer Science, pages 110–122. Springer Verlag. 2003.

[Fuentes *et al.* 2003b] R. Fuentes, J.J. Gómez-Sanz, J. Pavón: *Social Analysis of Multi-Agent Systems with Activity Theory*. In Proceedings of CAEPIA 2003, San Sebastian, Spain, November 2003. Volume 3040 of Lecture Notes in Artificial Intelligence, pages 526-535. Springer Verlag. 2004.

[Fuentes *et al.* 2004a] R. Fuentes, J.J. Gómez-Sanz, J. Pavón: *Towards Requirements Elicitation in Multi-Agent Systems*. In Proceedings of the 4th International Symposium From Agent Theory to Agent Implementation, AT2AI 2004, pages 582-587, Vienna, Austria, April 2004.

[Fuentes *et al.* 2004b] Rubén Fuentes, Jorge Gómez-Sanz, Juan Pavón, Lorna Uden: *Activity Theory Applied to Requirements Elicitation of Multi-Agent Systems*. In Proceedings of the 1st International Workshop on Activity Theory Based Practical Methods for IT Design, ATIT 2004, Copenhagen, Denmark, September 2004. 12 pages. To appear.

[Fuentes *et al.* 2004c] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Captura del Entorno Social de Sistemas Multiagente*. In Proceedings of the 3rd International Workshop on Practical Applications of Agents and Multi-Agent Systems, IWPAAMS 2004, Burgos, Spain, October 2004. 10 pages. To appear.

[Fuentes *et al.* 2004d] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Técnicas de Verificación y Validación para Sistemas Multi-Agente*. Novatica 170, 5 pages, Julio-Agosto 2004.

[Fuentes *et al.* 2004e] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Managing Conflicts between Individuals and Societies in Multi-Agent Systems*. In Proceedings of the 5th International Workshop Engineering Societies in the Agents World, ESAW'04, Toulouse, France, October 2004. To appear in Lecture Notes in Artificial Intelligence, 12 pages. Springer Verlag.

[Fuentes *et al.* 2004f] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *A Sociological Framework for Multi-Agent Systems Verification and Validation*. In Proceedings of the International Workshop on COnceptual MOdelling for Agents, CoMoA 2004, Shanghai, China, November 2004. To appear in Volume 3289 of Lecture Notes in Computer Science, pages 458-469. Springer Verlag.

[Fuentes *et al.* 2004g] Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón: *Checking Social Properties of Multi-Agent Systems with Activity Theory*. In Proceedings of the 9th Ibero-American Conference on Artificial Intelligence, Iberamia 2004, Puebla, Mexico, November 2004. To appear in Volume 3315 of Lecture Notes in Artificial Intelligence, pages 1–11. Springer Verlag.

[Fuxman *et al.* 2001] Ariel Fuxman, Marco Pistore, John Mylopoulos, Paolo Traverso: *Model Checking Early Requirements Specifications in Tropos*. In Proceedings 5th IEEE International Symposium on Requirements Engineering (RE01). 2001.

[Gamma *et al.* 1995] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Professional Computing Series. 1995.

[Genesereth & Ketchpel 1994] Genesereth, M. R., Ketchpel, S. P.: *Software Agents*. Communications of the Association for Computing Machinery, July 1994, pp 48-53.

[Goguen 1992] Goguen J. A.: *The Dry and the Wet*. Information System Concepts, pp. 1-17. Elsevier North-Holland, 1992.

[Goguen & Linde 1993] Goguen, J. A., Linde, C.: *Techniques for requirements elicitation*. In Proceedings of the 1st IEEE International Symposium on Requirements Engineering (RE'93). IEEE Computer Society Press, Los Alamitos, CA., 152-164. 1993.

[Gómez-Sanz 2003] Jorge J. Gómez-Sanz: *Metodologías para el diseño de sistemas multi-agente*. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 18: 51-64, 2003.

[Gómez-Sanz *et al.* 2003] J. Gómez-Sanz, J. Pavón, A. Díaz-Carrasco: *The PSI3 Agent Recommender System*. In Proceedings of the International Conference on Web Engineering, ICWE 2003. Lecture Notes in Computer Science 2722, pp. 30-39. 2003.

[Gómez-Sanz & Pavón 2004] Jorge Gómez-Sanz, Juan Pavón: *Methodologies for Developing Multi-Agent Systems*. In Journal of Universal Computer Science, 10 (4), pp. 359-374. 2004.

[Gould *et al.* 2000] Gould, E., Verenikina, I., Hasan H.: Activity Theory as a Basis for the Design of Web Based System of Inquiry for World War 1 Data. In Proceedings of the 23rd IRIS Conference, pp. 761-770, Lingatan, Sweden, August, 2000.

[Gurevich 1984] Gurevich, Yuri: *Toward logic tailored for computational complexity*. Computation and Proof Theory, 1104:175–216. 1984.

[Hall *et al.* 1995] Hall, E. P., Gott, S. P., Pokorny, R. A.: *A Procedural Guide to Cognitive Task Analysis: The PARI Methodology*. Armstrong Research Laboratory, Brooks AFB, Texas. 1995.

[Hasu & Engeström 2000] Mervi Hasu, Yrjö Engeström: *Measurement in Action: An Activity-Theoretical Perspective on Producer-User Interaction*. International Journal of Human-Computer Studies, special issue on Understanding Work and Designing Artefacts, vol. 53(1), pp. 61-69, July 2000.

[Hedestig & Kaptelinin 2002] Hedestig, U., Kaptelinin, V.: *Re-contextualization of teaching and learning in videoconference-based environments: An empirical study*. In Stahl, G (ed) Proceedings of CSCL 2002, Computer Support for Collaborative Learning: Foundations for a CSCL Community, Lawrence Erlbaum Associates Inc, Hillsdale New Jersey, USA. 2002.

[Holtzblatt & Beyer 1993] Holtzblatt, K., Beyer, H.: *Making customer-centered design work for teams*. Communications of the ACM 36, 10. 1993.

[Hughes *et al.* 1994] Hughes, J., King, V., Rodden, T., Andersen, H.: *Moving out from the control room: ethnography in system design*. In Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work - CSCW'94 (Chapel Hill, NC, 1994) ACM Press, pp. 429-439.

[Hutchins & Klausen 1992] Hutchins, E., Klausen, T.: *Distributed cognition in an airline cockpit*. In D. Middleton & Y. Engestrom (Eds.), Communication and Cognition at Work. Beverly Hills, CA.: Sage Books. 1992.

[IBM 2002] IBM alphaWorks: *Robocode. 2002*. http://robocode.alphaworks.ibm.com

[Iglesias *et al.* 1998] Iglesias, C. A., Garijo, M., Gonzalez, J. C., and Velasco, J. R.: *Analysis and design of multiagent systems using MAS-CommnonKADS*. In Intelligent Agents IV. LNAI Volume 1365 ed. Springer Verlag: Berlin, 1998.

[Iglesias *et al.* 1999] Iglesias, C. A:, Garijo, M., and Gonzalez, J. C.: *A Survey on Agent-Oriented Methodologies*. In Intelligent Agents V. LNAI Volume 1555, pp. 317-330 ed. Springer Verlag: Berlin, July 1999.

[Ilyenkov 1982] E. V. Ilyenkov: *The dialectics of the abstract and the concrete in Marx's Capital*. Moscow: Progress. 1982.

[Jacobson *et al.* 1999] Jacobson, I., Rumbaugh, J., Booch, G.: *The Unified Software Development Process*. Addison-Wesley, 1999.

[Jennings & Wooldridge 2000] N. R. Jennings, M. Wooldridge: *Agent-Oriented Software Engineering*. In Handbook of Agent Technology, (ed. J. Bradshaw) AAAI/MIT Press, 2000.

[Johnson & Johnson 1975] Johnson, D. W., Johnson, R.: *Learning together and alone: Cooperative, competitive, and individualistic learning*. Boston: Allyn & Bacon. First edition, 1975.

[Kaptelinin *et al.* 1999] V. Kaptelinin, B. A. Nardi, C. Macaulay: *The Activity Checklist: A tool for representing the "space" of context*. Interactions, 6 (4), p27-39. 1999.

[Kinny *et al.* 1996] Kinny, D., Georgeff, M., Rao, A.: *A methodology and modelling technique for systems of BDI agents*. In van der Velde, W. and Perram, J. editors, Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-96), Lecture Notes in Artificial Intelligence Vol. 1038, pages 56–71. Springer-Verlag. 1996.

[Kinny *et al.* 1997] Kinny, D., Georgeff, M., Rao, A.: *A Methodology and Modelling Technique for Systems of BDI Agents*. Informe. 1997.

[Kuutti 1996] K. Kuutti: *Activity Theory as a potential framework for Human-computer interaction research*. In B.A. Nardi, (ed.), context and consciousness: Activity Theory and Human-Computer Interaction. Cambridge, MA: MIT press. 1996.

[Lamsweerde 2000] A. van Lamsweerde: *Requirements Engineering in the Year 00: A Research Perspective*. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE-2000)*, Limerick, Ireland, June 2000. ACM Press.

[Lamsweerde & Willemet 1998] A. van Lamsweerde, L. Willemet: *Inferring Declarative Requirements Specifications from Operational Scenarios*. IEEE Transactions on Software Engineering, Special Issue on Scenario Management, December 1998.

[Leontiev 1974] Aleksie Nikolaevich Leontiev: *The problem of activity in psychology*. In Soviet Psychology, 13(2), 4–33. 1974.

[Leontiev 1978] Aleksie Nikolaevich Leontiev: *Activity, Consciousness, and Personality*. Prentice-Hall. 1978.

[Leontiev 1981] Aleksie Nikolaevich Leontiev: *Problems of the development of the mind*. Moscow, Progress. 1981.

[Leontiev 1989] Aleksie Nikolaevich Leontiev: *The problem of activity in the history of Soviet psychology*. In Soviet Psychology, 27(1), 22–39. 1989.

[Livingstone 1988] Livingstone, E.: *Making Sense of Ethnomethodology*. London: Routledge, Kegan and Paul. 1988.

[Luria 1976] Luria, A. R.: *Cognitive development: Its cultural and social foundations*. Cambridge, Mass.: Harvard University Press. 1976.

[Luria 1979] Aleksandr Romanovich Luria: *The Making of Mind: A Personal Account of Soviet Psychology*. Harvard University Press. 1979.

[Lyytinen & Rossi 1999] Lyytinen, K. S., Rossi, M.: *METAEDIT+ --- A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment*. In Proceedings of CAISE-96. Springer-Verlag. LNCS 1080. 1999.

[Maes 1994] Pattie Maes: *Modeling Adaptative Autonomous Agents*. Artificial Life Journal, C. Langton, ed., Vol. 1, No. 1 & 2, MIT Press, 1994.

[Maes 1998] Pattie Maes: *Agents that Reduce Work and Information Overload*. Readings in Intelligent User Interfaces, Morgan Kauffman Publishers, 1998.

[Malsch 2001] Thomas Malsch: *Naming the Unnamable: Socionics or the Sociological Turn of/to Distributed Artificial Intelligence*. Autonomous Agents and Multi-Agent Systems 4 (3): 155-186. 2001.

[Marx & Engels 1844] Marx, K., Engels, F.: *Works*. Vol. 3. Progress Publishers of the Soviet Union. 1975.

[Marx 1909] Marx, K.: *Capital*. Vol. 1. London, William Glaisher. 1909.

[Marx 1973] Marx, K.: *Grundrisse: Foundations of the critique of political economy (rough draft)*. Harmondsworth: Penguin Books. 1973.

[McGrath & Uden 2000] G. Michael McGrath, Lorna Uden: *Modelling Softer Aspects of the Software Development Process: An Activity Theory based approach*. 33rd Hawaii International Conference on System Sciences. (HICSS-33) Wailea, Maui, Hawaii, USA - Software Process Improvement. IEEE Computer Society Press. January 2000.

[Miao *et al*. 2000] Yongwu Miao, Shirley Holst, Torsten Holmer, Jutta Maria Fleschutz, Peter Zentel: *An Activity-Oriented Approach to Visually Structured Knowledge Representation for Problem-Based Learning in Virtual Learning Environments*. In Proceedings of the 5th International Conference on the Design of Cooperative Systems (COOP'2000). May 2000, Sophia Antipolis, France, pp. 303-318, Frontiers in Artificial Intelligence and Applications. Vol. 58, ISSN 0922-6389, Amsterdam, the Netherlands, IOS Press, 2000, ISBN 1-58603-042-6

[Militello & Hutton 1998] L. G. Militello, R. J. B. Hutton: *Applied Cognitive Task Analysis (ACTA): a practioner's toolkit for understanding cognitive task demands*. Ergonomics, 4 (11), 1618-1641. 1998.

[Mwanza 2000] Daisy Mwanza: *Mind the Gap: Activity Theory and Design*. KMI Technical Reports, KMI-TR-95, http://kmi.open.ac.uk/publications/techreports.html, Knowledge Media Institute, The Open University, Milton Keynes, UK. 2000.

[Newell 1982] A. Newell: *The knowledge level*. Artificial Intelligence, vol. 18 pp. 87-127, 1982.

[Nuseibeh & Easterbrook 2000] Nuseibeh, B.A., Easterbrook, S.M.: *Requirements engineering: A roadmap*. In Proceedings of the 22nd International Conference on Software Engineering (ICSE'00), pages 35–46. 2000.

[OME 2000] University of Toronto, Canada: *OME: Organization Modelling Environment. Version 3*. 2000. http://www.cs.toronto.edu/km/ome/

[OMG 2002] OMG: *MOF. Meta Object Facility. Specification. Version 1.4*. 2002. http://www.omg.org

[OMG 2003] OMG: *Unified Modeling Language Specification. Version 1.5*. 2003. http://www.omg.org

[Pavón & Gómez-Sanz 2003] Pavón J., Gómez-Sanz, J.: *Agent Oriented Software Engineering with INGENIAS*. In V. Marík, J. Müller, and M. Pechoucek, editors, Multi-Agent Systems

and Applications III, volume 2691 of LNCS, pages 394-403. 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003), Springer Verlag.

[Perini *et al.* 2001] A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, and J. Mylopoulos: *A Knowledge Level Software Engineering Methodology for Agent Oriented Programming*. In Proceedings of the 5th International Conference on Autonomous Agents, Montreal CA, 28 May - 1 June 2001.

[Pressman 2000] Pressman, R. S.: *Software Engineering: A Practitioner's Approach*. *5th edition*. McGraw-Hill Series in Software Engineering and Technology. McGraw-Hill, Inc. 2000.

[Probert 1999] Probert, S. K.: *Requirements engineering, soft system methodology and workforce empowerment*. Requirements Engineering, 4 (1999), 85-91, Springer-Verlag, London. 1999.

[Raccoon 1995] L. B. S. Raccoon: *The Chaos Strategy*. Software Engineering Notes, Volume 20, Number 5, Pages 40 to 47, ACM Press. 1995.

[Rao & Georgeff 1991] A. S. Rao, M. P. Georgeff: *Modeling rational agents within a BDI-architecture*. In J. Allen,R. Fikes, and E. Sandewall, editors, Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, 473–484. Morgan Kaufmann. 1991.

[Rao & Georgeff 1995] A. S. Rao, M. P. Georgeff: *BDI Agents: From Theory to Practice.* In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95), 312-319. 1995.

[Ricci *et al.* 2002] A. Ricci, A. Omicini, E. Denti: *Activity Theory as a Framework for MAS Coordination*. In Proceedings of the Engineering Societies in the Agents World III: Third International Workshop, ESAW 2002, Madrid, Spain, September 16-17, 2002. Lecture Notes in Artificial Intelligence, Springer (Berlin, D), Vol. 2577, March 2003.

[Ross 1977] D. T. Ross: *Structured Analysis (SA): A Language for Communicating Ideas*. IEEE Transactions on Software Engineering, Vol. 3, No. 1, 1977, 16-34.

[Ruohonen 1996] Mikko Ruohonen: *Information Technology Mediated Activities in Organizational Contexts – A case of Strategic Information System Planning*. In TUCS Technical Reports 3, ISBN 951-650-730-1, April 1996.

[Russell & Norvig 1995] Russell, S., P. Norvig: *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey. 1995.

[Saam & Schmidt 2002] Nicole J. Saam, Bernd Schmidt: *Cooperative Agents: Applications in the Social Sciences*. Nicole J. Saam and Bernd Schmidt editors. In Theory and Decision Library A: Philosophy and Methodology of the Social Sciences, vol. 32, Kluwer Academic Publishers. 2002.

[Sannicolo' *et al.* 2002] Sannicolo' F., Perini A., Giunchiglia F.: The Tropos Modeling Language. A user guide. ITC-irst Centro per la Ricerca Scientifica e Tecnologica., Technical Report # 0204-13. April 2002.

[Shoham 1993] Shoham, Y.: *Agent Oriented Programming*. Artificial Intelligence, vol. 60 pp. 51-92, 1993.

[Sommerville 2001] Ian Sommerville: *Software Engineering*. Addison-Wesley International Computer Science Series. Addison-Wesley 2001, 6th edition.

[Suchman 1987] Suchman, L.: *Plans and situated actions. The problem of human-machine communication*. Cambridge: Cambridge University Press. 1987.

[Sykara 1998] Katia P. Sykara: *Multiagent systems*. AI Magazine 19(2). 1998.

[Tuikka 2002] Tuomo Tuikka: *Towards Computational Instruments for Collaborating Product Concept Designers*. Ph. D. Thesis. Department of Information Processing Science. Oulu University. 2002.

[Uden *et al.* 2001] Lorna Uden, Kecheng Liu, Gary Shank: *Linking Radical Constructivism and Semiotics to Design a Constructivist Learning Environment*. Journal of Computing in

Higher Education, Spring 2001, Volume 12, Number 2, Amherst, MA: Norris Publishers, pp 34-51.

[Vygotsky 1978] L. S. Vygotsky: *Mind and Society*. Cambridge MA, Harvard University. 1978.

[Werner & Shoepfle 1987] Werner, O., Shoepfle, G. M.: *Systematic Fieldwork: Foundations of Ethnography and Interviewing*. Sage Publications, Newbury Park. 1987.

[Winograd & Flores 1986] Winograd, T., Flores, C.F.: *Understanding Computers and Cognition: A new foundation for design*. Norwood, NJ: Ablex. 1986.

[Wooldridge & Jennings 1998] M. Wooldridge, N. R. Jennings: *Pitfalls of agent-oriented development*. In Proceedings of the Second International Conference on Autonomous Agents (Agents 98), pages 385–391, Minneapolis/St Paul, MN, May 1998.

[Wooldridge & Ciancarini 2000] Wooldridge, M., Ciancarini, P.: *Agent-Oriented Software Engineering: The State of the Art*. In Ciancarini, P. and Wooldridge, M., editors, First International Workshop on Agent-Oriented Software Engineering, volume 1957, pages 1–28. Springer-Verlag, Berlin. 2000.

[Wooldridge *et al.* 2000] Wooldridge, M., Jennings, N. R., and Kinny, D.: *The Gaia Methodology for Agent Oriented Analysis and Design*. In Journal of Autonomous Agents and Multi-Agent Systems, vol. 15. 2000.

[Yu & Mylopoulos 1994] E. Yu, J. Mylopoulos: *Towards Modelling Strategic Actor Relationships for Information Systems Development - With Examples from Business Process Reengineering*. In Proceedings of the 4th Workshop on Information Technologies and Systems, Vancouver, B.C., Canada, December 17-18, 1994. pp. 21-28.

[Yu 1999] Eric Yu: *Strategic Modelling for Enterprise Integration*. 14th World Congress. International Federation of Automatic Control (IFAC'99). July 5-9, 1999 Beijing, China.

[Zave & Jackson 1997] Pamela Zave, Michael Jackson: *Four dark corners of requirements engineering*. ACM Transactions on Software Engineering and Methodology, 6(1):1-30, January 1997.

[Zave 1999] Pamela Zave: *Formal description of telecommunication services in Promela and Z*. In Manfred Broy and Ralf Steinbruggen, editors, *Calculational System Design* in Proceedings of the Nineteenth International NATO Summer School, pages 395-420, IOS Press. 1999.

[Zinchenko 1985] Zinchenko, V. P.: *Vygotsky's ideas about units for the analysis of mind*. In J. V. Wertsch (ed.) *Culture, communication, and cognition: Vygotskian perspectives*. Cambridge: Cambridge University Press (pp. 94-118). 1985.

Glosario

API	*Application Programming Interface*
ATA	Asistente de la Teoría de Actividad
BDI	*Beliefs-Desires-Intentions*
CTA	Contradicciones de la Teoría de Actividad
CSCW	*Computer Supported Cooperative Work*
GA	Guía de Actividades
GCR	Guía de Captura de Requisitos
GOPRR	*Graph*, *Object*, *Property*, *Relationship*, *and Role*
GRASIA	GRupo de Agentes Software: Ingeniería y Aplicaciones. Departamento de Sistemas Informáticos y Programación de la Universidad Complutense de Madrid
HCI	*Human-Computer Interaction*
IDK	*INGENIAS Development Kit*
ISOA	Ingeniería del Software Orientada a Agentes
MOF	*Meta Objects Facilities*
OCL	*Object Constraint Language*
OMG	*Object Managemet Group*
OO	Orientado a Objetos, Orientación a Objetos
SMA	Sistema Multi-Agente
TA	Teoría de Actividad
UML	*Unified Modelling Language*
UML-TA	Representación del vocabulario de la TA con UML
USDP	*Unified Software Development Process*
XML	*eXtended Modelling Language*