Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural

María Carmen Leonardi

Director: Dr. Julio Leite
Co-director: Dr. Gustavo Rossi

Tesis presentada al Dpto. de Informática de la Universidad Nacional de La Plata como parte de los requisitos para la obtención del título de Magister en Ingeniería de Software.

La Plata, Noviembre de 2001

Facultad de Informática
Universidad Nacional de La Plata - Argentina

Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural

Resumen

En esta tesis se presenta una estrategia para la definición de un modelo conceptual de objetos a partir de modelos de requisitos basados en Lenguaje Natural. Más precisamente, se utilizan modelos pertenecientes a la *Requirements Baseline*, en particular, el *Léxico extendido del Lenguaje*(*LEL*), para modelar el lenguaje del Universo del Discurso (*UofD*), el *Modelo de Escenarios* para representar el comportamiento y un *Modelo de Reglas de Negocio* para definir las reglas de la organización. Se define un conjunto de heurísticas que permite manipular la gran cantidad de información generada por estos modelos, con el objetivo de definir un modelo de objetos. La modelización consta principalmente de dos modelos: el modelo de *CRCs* que define al *UofD* en términos de clases, responsabilidades y colaboraciones y un modelo lógico que representa los aspectos estructurales, más concretamente los métodos, atributos y asociaciones de las clases.

Las heurísticas guían la construcción de ambos modelos a partir de los modelos de la *Requirements Baseline*. La aplicación de estas heurísticas permiten definir relaciones de trace entre los modelos generadores y los generados, mejorando la *pre-traceability*. Este modelo de objetos es independiente del sistema de software que se va a construir y de la metodología de desarrollo de software orientada a objetos que se elija para las etapas posteriores.

Abstract

In this thesis, a strategy for the definition of an Object Conceptual Model from requirements models based on Natural Language is presented. More precisely, models belonging to Requirements Baseline, more specifically, the Language Extended Lexicon (LEL) to model the Language of the Universe of Discourse (*UofD*), the Scenario Model to represent behaviour and a Business Rule Model to define organisation rules, are used. A set of heuristics is defined in order to manipulate a large quantity of data generated by these models with the objective of defining an object model. Modelling has mainly two models: the CRCs model which defines *UofD* in terms of classes, responsibilities and collaborations and a logical model representing structural aspects and in a more concrete way, methods, attributes and class associations.

Heuristics guide the construction of both models from Requirement Baseline models. The application of these heuristics allows to define trace relationships among generating and generated models, thus enhancing *pre-traceability*. This object model is independent from the software system to be constructed and from the object-oriented software development methodology chosen for further stages.

Índice

Lista de Figuras

Capítulo I
Introducción

1.1 Motivación

"*Se puede considerar al problema del desarrollo de software como un problema de construir un artefacto*[1]*. Este artefacto será instalado en el mundo con el cual va a interactuar* [Jackson'95]". Esa parte del mundo en la cual los efectos del artefacto serán sentidos, evaluados y aprobados en caso de éxito, se *denomina Dominio de Aplicación* o en el contexto de esta tesis *Universo de Discurso, UofD* [Leite'95] o Macrosistema. El *UofD* es el contexto general en el cual el software será desarrollado, operado y mantenido. Incluye todas las fuentes de información y personas o sectores relacionados con la aplicación Las personas y sectores son llamados actores. El universo de discurso esta condicionado por el conjunto de objetivos establecidos por aquellos actores que demandan una solución de software para las tareas que realizan. El *UofD* es donde se originan los requisitos [Jackson'95; Jacobson'99], por lo que, sino se define apropiadamente no será posible enfocarse en los mismos. Un problema está más caracterizado por la estructura y propiedad del *UofD* al que pertenece, que por el *artefacto* que se construirá para resolverlo [Jackson'95]. Asimismo, en su guía práctica de Requisitos, Sommerville aconseja considerar los aspectos de organización en la cual estará inserto el software (reglas 3.4: Makes a business Case for the system, 4.6:Use Business concerns to Drive Requirement Elicitation [Sommerville'97]), determinando que el costo de introducción y de aplicación de estas reglas es bajo.

A partir de las necesidades mencionadas anteriormente, las metodologías orientadas objetos, surgidas desde el área de programación [Jackson'95], han incorporado técnicas de modelado conceptual en los últimos tiempos [Kotonya'98]. Sin embargo, como se indica en [Jackson'95] los conceptos de orientación a objetos, son más útiles para la etapa de programación (y también para la etapa de diseño, como por ejemplo los *design patterns* propuestos por [Gamma'94]) que para ser utilizados en forma pura en los modelos de requisitos. El concepto fundamental de cualquier modelo de objetos es obviamente la noción de objetos [Kotonya'98], donde un objeto subsume comportamiento, eventos y datos; todo es un objeto [Jackson'95]. Esta abstracción no resulta natural para el cliente quien piensa principalmente en términos de las actividades, objetivos, reglas y recursos dentro de su organización. Algunas metodologías Orientadas a Objetos incorporan otras técnicas, como son los *Use-Cases* [Jacobson'99; Cockburn'99] o técnicas similares como los *User-Stories* [Beck'00] constituyendo los primeros modelos para la etapa de modelado conceptual. Este modelo, si es utilizado como único modelo para definir las clases, puede provocar algunos inconvenientes en la modelización [Meyer'97], por lo que debiera

[1] artefacto: término con que Jackson denomina al software [Jackson'95]

ser acompañado de otros modelos que complementen la visión de los mismos. Por esta razón creemos que la incorporación de técnicas de requisitos que modelen diferentes aspectos del *UofD* puede ayudar en las primeras etapa de un desarrollo orientado a objetos. Se modela el Macrosistema en términos de modelos orientados al cliente utilizando lenguaje natural. A partir de estos modelos se pueden derivar un modelo conceptual de objetos siguiendo las heurísticas propuestas en esta tesis. Un modelo conceptual de objetos permite a los ingenieros de software visualizar el Macrosistema en términos de los conceptos que se manejarán a lo largo de todo el proceso de desarrollo, haciendo que la transición desde el modelado conceptual al diseño e implementación sea un proceso más natural.

Los modelos utilizados para representar al *UofD*, en nuestro caso particular la *Requirements Baseline extendida* [Leite'97; Leite'98], son modelos basados en Lenguaje Natural que contiene gran cantidad de información representando los conceptos, comportamiento, relaciones, estructuras y reglas que existe en una organización. Manipular esta información para obtener un modelo conceptual de objetos no es una tarea trivial, ya que debe filtrarse y modelarse en términos de los conceptos que se manejarán en la solución, en nuestro caso, objetos y relaciones. Las clases conceptuales son fundamentales ya que representan el conocimiento de los expertos del Macrosistema definiendo un modelo de negocio para el cual se va a desarrollar un sistema de software. La definición de clases es un proceso dual [Meyer'97], así como se definen clases determinando abstracciones relevantes para el sistema, es igualmente importante poder descartar clases. Por esta razón, es crucial adquirir la información del Macrosistema que se va a modelar y poder manipular esa información para poder definir y descartar las abstracciones sugeridas por los modelos. Por ejemplo, el LEL[Leite'95] acota el espacio de búsqueda para determinar las clases relevantes al *UofD* y que propiedades y comportamiento son importantes o no para el mismo.

A partir de los modelos escritos en lenguaje natural, definimos un conjunto de heurísticas basadas principalmente en la semántica de las entidades que pertenecen a la *Requirements Baseline* así como también a su estructura *(*meta-modelos). Un criterio puramente sintáctico (basarse en sustantivos, en tipos de verbo) puede hacer perder importantes abstracciones del Macrosistema, o llevarnos a definir abstracciones no útiles para el mismo. La libertad de expresión del lenguaje natural nos da la posibilidad de expresar conceptos utilizando diferentes componentes gramaticales, como por ejemplo sustantivos o frases verbales. En [Meyer'97] se menciona un claro ejemplo de este caso:

Se debe guardar un registro cada vez que un elevador se mueve de un piso a otro
Se debe guardar un registro cada vez que se produce un movimiento de un ascensor de un piso a otro.

La palabra mover aparece como una frase verbal en una sentencia y como sustantivo, en otra. Si bien en la primera aparecería registro como una posible clase, claramente es el registro de un movimiento el que interesa guardar, y esta abstracción es más correcta que la anterior. Un enfoque puramente sintáctico podría obviar esta abstracción.

Por todo lo anteriormente expuesto, en esta tesis se propone un conjunto de heurísticas que guían la construcción de un modelo de objetos haciendo uso de todo el conocimiento obtenido en la definición de los modelos. Esta heurísticas ayudan a

manipular la gran cantidad de información y relaciones de los modelos del LEL, escenarios y Reglas del Negocio.

1.2 Nuestra Propuesta

1.2.1 Objetivo

El objetivo principal de esta tesis es definir una estrategia que permita especificar un modelo conceptual de objetos a partir de herramientas de modelado de requisitos basadas en lenguaje natural. El modelo de objetos cubre aspectos estructurales y de comportamiento y refleja el Macrosistema en el cual el sistema de software estará inserto. El modelo conceptual obtenido tiene el mismo propósito que el Modelo de Dominio de la metodología *Rational Unified Process* y su lenguaje *UML* [Booch'98] , conocida como *RUP/UML* [Jacobson'99] y el modelo conceptual de Fowler [Fowler'96]. Adaptando la visión de ambos autores a nuestra definición de contexto del sistema como *UofD* podemos decir:

> *El objetivo de un modelo conceptual de objetos es contribuir al entendimiento del UofD, y de ese modo al entendimiento de los requisitos del sistema, ya que aquellos se originan en el propio UofD. En otras palabras, la modelización del UofD debe contribuir al entendimiento del problema para el cual se supone que el sistema resolverá en relación con su contexto. La forma interna en que el sistema resuelva este problema se tratará en las siguientes etapas de desarrollo del software. El modelo conceptual es la base para este desarrollo.*

Asimismo, se presenta en esta tesis, un Modelo de Reglas del Negocio que complementa los aspectos de comportamiento y vocabulario dado por el modelo de LEL y escenarios. Extiende la *Requirements Baseline* a través de un modelo orientado a los negocios, fácil de comprender y analizar con el cliente. Este modelo es utilizado junto con el modelo de escenarios y LEL para derivar el modelo conceptual de objetos.

1.2.2 Descripción general de la estrategia

La estrategia consiste en definir a partir de los modelos de escenarios, *LEL* y reglas de negocio correspondientes a la *Requirements Baseline*, un modelo preliminar de objetos. El *LEL* permite obtener un conocimiento global de la entidad, es decir, cómo se comporta la entidad en todo el *UofD*. De esta forma se puede tener una visión más global de un actor o un recurso, y si se quiere detallar, se puede analizar su comportamiento en cada uno de los escenarios en donde aparece. El *LEL* permite relacionar a los escenarios y determinar el comportamiento total de una entidad dentro del sistema. Si sólo se tienen escenarios se corre el riesgo de analizarlos por separado y no tener en cuenta que un objeto puede estar en varios de ellos (sobre todos para proyectos grandes). Los escenarios ayudan a refinar las responsabilidades de las clases y sus colaboraciones. Finalmente el modelo de reglas, presentado en esta tesis, modifica el modelo de objetos para reflejar las políticas de la organización desde el punto de vista del cliente, ya sea incorporando nuevas clases o modificando las existentes.

La Figura 1.1 describe la estrategia general a través de un patrón de proceso [Fiorini'98] indicando el capítulo en el cual se desarrolla cada etapa.

Identificación: Definición de Modelo de Objetos dirigida por Modelos basados en Lenguaje Natural.
Clasificación: patrón de procesos, desarrollo de software e ingeniería de requisitos
Objetivo: guiar a los ingenieros de software durante las primeras etapas de un desarrollo de software Orientado a Objetos. Provee un método que permite la definición de un modelo lógico utilizando la descripción del Macrosistema.
Problema (¿qué?):

- La forma de pensar de los clientes difiere del paradigma de orientación a objetos. Generalmente los clientes piensan en las acciones realizadas para cumplir con sus objetivos manipulando recursos y comunicándose con otros actores. La orientación a Objetos es una abstracción no fácilmente comprensible por los clientes ya que los objetos encapsulan datos y comportamiento en un mismo nivel. Cualquier concepto del mundo real se transforma en objetos y sus relaciones [Jackson'95].
- Las técnicas tradicionales de Orientación a Objetos no cubren el proceso de adquisición y modelado de Requisitos en forma detallada.
- Contexto:

¿Donde? Se realiza durante la etapa de modelado de requisitos en un desarrollo de software orientado a objetos

¿Por qué? Porque es difícil entender el problema
Por el conocimiento tácito [Zave'97]
Porque es difícil encontrar un equilibrio entre la representación que el cliente fácilmente comprende y la representación que puede ser analizada y manipulada por los ingenieros de software

¿Cuando? En las primeras etapas del desarrollo del software, al definir los objetos

¿Quien? Los clientes y los ingenieros de software.

¿Como? Definiendo el modelo LEL para el Macrosistema (cap. 2)
Definiendo los escenarios para el Macrosistema (cap. 2)
Definiendo las reglas de negocio (cap. 3)
Definiendo las CRCs (cap. 4)
Definiendo las clases primarias
Definiendo las clases secundarias
Refinando las colaboraciones
Definiendo el modelo lógico (cap. 5)
Definiendo el diagrama de objetos
Definiendo el diagrama de interacción
Determinando los límites del sistema de software (cap. 6)
Determinando relaciones de trace (cap. 7)

Figura 1.1: patrón de descripción de Estrategia

Si bien a lo largo de todo el trabajo nos referimos al término clase, se debe tener en cuenta que al no tener establecido los límites de software y modelar todo el Macrosistema del cual el sistema de software será una parte, se está modelando clases del Macrosistema, o *componentes* como se denominan en [Cockburn'99]. Las clases del Macrosistema o *componentes* están orientadas a los negocios, no al sistema de software y pueden ser un sistema, una organización, una persona, o una clase de objetos en un sistema orientado a objetos. Las clases del Macrosistema tienen un conjunto de responsabilidades, que son los servicios que se ejecutan en el mismo.

1.2.3 Evaluación de la propuesta

Basándose en la clasificación presentada en [Zelkowitz'98], la evaluación se hizo utilizando las estrategias de validación histórica y observacional. En cuanto a la estrategia de validación histórica se usaron los métodos de búsqueda literaria (cuyos resultados se describen en el capítulo 9 en Comparación con otras estrategias) y lecciones aprendidas a partir de los casos de Estudio (capítulo 8 sección 8.3), así como también en los casos en que ha sido utilizada como estrategia complementaria [Neto'00]. En cuanto a la validación observacional, se utilizó los métodos de caso de estudio y estudio de campo, utilizando proyectos de envergadura media, cuyos resultados se describen en el capítulo 8.

1.2.4 Publicaciones y trabajos relacionados

1.2.4.1 Publicaciones

Los siguientes artículos publicados son algunos de los resultados obtenidos respecto al tema de esta tesis:

Trabajos relacionados a la Incorporación de un modelo de reglas (cap. 3 de esta tesis)

-*Business rules as organizational Policies*
Leite J.C.S.P, Leonardi Ma. Carmen
IEEE Ninth International Workshop on Software Specification and Design
IEEE Computer Society Press, 1998, pp. 68-76.

- *Estrategias para la identificación de Reglas de Negocio*
Leonardi Carmen, Leonardi Carmen, Leite J.C.S.P, Rossi G
Anais de Sbes98 "Simposio Brasilero de Engenharia de Software" Sociedade Brasilera de Computacao, Maringa, Brasil, 14-16 de Octubre de 1998, pp. 53-67.

Trabajos relacionados a la Estrategia de modelización de objetos a partir de los modelos de LEL, escenarios y reglas. Aspectos de Traceability. (cap. 4-7 de esta tesis)

- *Una Estrategia de Análisis Orientada a Objetos Basada en Escenarios*
Leonardi Carmen, Maiorana Vanesa, Balaguer Federico
Actas de II Jornadas de Ingeniería de Software, JIS97, Dpto. de Informática, Universidad del país Vasco, San Sebastián, España, 1997, pp. 87-100.

- *Integración de un Modelo de Reglas a la definición de Requisitos*
Leonardi Carmen, Leite J.C.S.P, Petersen Laura
Actas de la Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software, Universidad Federal do Río Grande do Sul, Instituto de Informática, 1998, pp. 273-285.

- *Un modelo de hipertexto para la especificación de Requisitos*
Leonardi Carmen, Rossi G, Leite J.C.S.P
Anais de WER98: Workshop de Engenerhia de Requisitos. Departamento de Informática. Puc Rio 1998, pp. 119-128.

- *Una Estrategia de Análisis Orientada a Objetos basada en Escenarios: Aplicación en un Caso Real*
Rivero L, Doorn J, del Fresno M, Mauco V, Ridao M, Leonardi C. Anais Workshop de Engenerhia de Requisitos, Departamento de Informática. Puc-Rio 1998, pp. 79-88.

- *Un modelo orientado a objetos para el rastreo de requisitos*
Leonardi Carmen, Leite J.C.S.P Rossi G, Actas de las Terceras Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes Software - 2000 Resumen, pp. 492-493.

1.2.4.2 Trabajos relacionados

Las siguientes son Tesis de grado de la Carrera de Ingeniería de Sistemas de la UNICEN co-dirigidas con relación al tema de esta tesis.

- Codirectora del seminario "Escenarios del proceso de construcción de escenarios: Auto-aplicación de la metodología" Alumnos: García Omar, Gentile Claudio. Nota final: 10. Año 2000

Esta tesis aplica la estrategia de derivación de objetos utilizando como caso de estudio la misma estrategia, es decir la estrategia de derivación de un modelo de objetos a partir de los modelos de la *Requirements Baseline*.

- Codirectora del seminario "*HEAR*: Una Herramienta para la Adquisición de Requisitos" Alumnos: Stella Tornabene, Laura Petersen. Año 1999. Nota final: 10

Esta tesis define un meta-modelo para la autoría y navegación de documentos textuales de requisitos. La navegación se hace a partir de contextos navegacionales, como se propone en el capítulo 7 de esta tesis. En particular fue instanciado con los modelos de LEL y de escenarios. Sin embargo, el meta-modelo puede ser instanciado con cualquier documento textual, por ejemplo Reglas de Negocio y *CRCs*.

1.3 Propuestas relacionadas

Existen varias propuestas de modelización conceptual orientadas a objetos que utilizan *Use-Cases* o sus variantes como principal herramienta para la modelización del sistema y su entorno. Entre las más conocidas podemos mencionar *RUP/UML*, *Responsibility-based Modeling*, *RBM* [Cockburn'99], *Responsibility-Driven Design RDD* [Wirfs'90; Wirfs'95], *Object Behavior Analysis OBA* [Rubin'92] y en una versión simplificada de *Use-Cases* llamadas *User-Stories*, en *Extreme Programming XP* [Beck'00]. Todas estas estrategias utilizan a los *Use-Cases* o sus variantes para modelar la interacción del usuario con el sistema y a partir de ello descubrir los objetos. Sin embargo, no presentan heurísticas explícitas para la modelización de los objetos. De todas estas metodologías, *RUP/UML* es la única que propone un modelo de negocios como un paso anterior a la definición de requisitos, necesario para comprender el contexto del sistema. En el capítulo 9 se detalla una comparación de estas metodologías y otras estrategias similares con respeto a nuestra propuesta. Resumiendo, nuestra estrategia difiere principalmente en dos aspectos:

El primer y principal aspecto está basado en el uso del LEL y el Modelo de Reglas junto con los Escenarios para la definición del modelo de objetos. La utilización de los *Use-Cases* como el único modelo para encontrar los objetos tiene varios riesgos [Meyer'97]: a) los *Use-Cases* enfatizan el orden en el tiempo lo cual es incompatible con la filosofía Orientada a Objetos. b) modelan lo que el usuario ve como las operaciones del sistema el cual todavía no existe. c) los *Use-Cases* favorecen un paradigma funcional basado en procesos. Creemos que el uso de modelos declarativos y de alto nivel de abstracción como el LEL y las Reglas pueden mitigar este problema, sin afectar los beneficios dado por los escenarios. El LEL define, a través de los impactos, el

comportamiento global de una entidad dentro del Macrosistema, permitiéndonos definir responsabilidades con un grado de abstracción más alto que los usados en la descripción de los escenarios. Por el otro lado, actúa como filtro para concentrarse solamente en los términos relevantes del Macrosistema. En cuanto al modelo de reglas, permite modelar de manera explícita e independiente las políticas de la organización y adaptar el modelo de objetos para reflejarlas, pudiendo en algunos casos generar nuevas clases que reflejen a las reglas, con una visión similar a la presentada en [Diaz'97] ya que en determinadas circunstancias la naturaleza global de las reglas impide que se puedan incorporar en una determinada clase [Hoydalsvik'93].

El segundo aspecto está relacionado con el alcance de los escenarios. *OBA* [Rubin'92], *RBM* [Cockburn'99], *RDD* [Wirfs'90; Wirfs'95], consideran a los *Use-Cases* o sus variantes como descripciones de la comunicación del usuario con el sistema. Por consiguiente ya consideran los límites entre el software y su entorno. Como ya se explicó anteriormente, en nuestra propuesta el alcance de los escenarios se extiende a las descripciones del Macrosistema. La clase del conocimiento que se maneja es la que Rolland et al. clasificaron como contexto organizacional [Rolland'98], que define *"la amplia descripción de como se lleva a cabo el trabajo"*. Nuestra hipótesis, confirmada por la literatura [Rolland'98; Zorman'95; Potts'95] y la realización de varios casos de estudio(capítulo 8), es que los escenarios proveen una atractivo medio de comunicación entre los actores del *UofD* manteniendo información de tal forma que puede ser reconocida por los mismos. Esta visión también se aplica al modelo de objetos, definiendo un modelo conceptual de los objetos del Macrosistema, sin tener en cuenta en una primer etapa cuáles se transformarán en objetos del sistema de software. En el caso de *RUP/UML*, si bien se propone, previo al modelo de requisitos, un Modelo de Negocios a partir de los *Business Use-Cases*, ya en este nivel comienza a considerarse aspectos que pueden ser automatizados, mientras que en nuestra estrategia, esta actividad se hace en una etapa posterior. Asimismo, *RUP/UML* no define heurísticas precisas para determinar las clases del Modelo de Negocios.

1.4 Organización de la tesis

Este trabajo esta organizado de la siguiente manera: en el capítulo 2 se describe brevemente los modelos de la *Requirements Baseline* y su proceso de construcción. El capítulo 3 presenta la incorporación del Modelo de Reglas de Negocio a la *Requirements Baseline*, la taxonomía propuesta y heurísticas para identificarlas y modelarlas. Los capítulos 4,5 y 6 describen la estrategia presentando cada una de las actividades definidas en la Figura 1.1 En el capítulo 4 se describe la construcción y validación de las CRCs. El capítulo 5 presenta las heurísticas para la construcción de un modelo lógico que refleje los aspectos estructurales desde el punto de vista de orientación a objetos. En el capítulo 6 se describen las heurísticas para definir los limites de software y formular un primer modelo de diseño que será la base para continuar con un desarrollo orientado a objetos siguiendo cualquier metodología. Durante estos capítulos se ejemplificará cuestiones puntuales utilizando ejemplos del caso de estudio desarrollado en el capítulo 8, o en caso de que no exista un ejemplo del punto específico que se desea ejemplificar, se tomará algunos de casos de estudios mencionados también en el capítulo 8. En el capítulo 7 se describen y ejemplifican las relaciones de traceability entre los modelos generados por las heurísticas de esta estrategia. El capítulo 8 muestra las aplicaciones de la estrategia, y se describe en detalle el desarrollo del "caso de estudio del Círculo de Ahorro para la compra de un

automóvil." Se discuten ventajas y problemas encontrados al aplicar esta estrategia. Finalmente en el capítulo 9 se compara esta estrategia con metodologías que utilizan modelos o técnicas similares. Este capítulo finaliza con las contribuciones de esta tesis y futuros trabajos. En el anexo I se presenta el modelo de LEL y escenarios del caso de Estudio del Círculo de Ahorro para la compra de un automóvil.

2.1 Introducción

La *Requirements Baseline* [Leite'95; Leite'97] es un meta-modelo que contiene descripciones sobre el *UofD* y el artefacto de software que ha de ser construido dentro de él. Estas descripciones, relacionadas entre sí, son escritas en lenguaje natural siguiendo patrones determinados. El uso de lenguaje natural posibilita la intervención de los stakeholders[2] a la hora de validar el conjunto de especificaciones obtenidas. La idea básica detrás de la *Baseline* es que es perenne: se genera durante el proceso de ingeniería de requisitos, evoluciona junto al proceso de desarrollo del software y acompaña las tareas de mantenimiento. La Fig. 2.1, extraída de [Leite'95] da una idea del paralelismo entre la evolución de la *Baseline* y la evolución del proceso de software. La *Requirements Baseline* es independiente del modelo de proceso usado en el desarrollo. En la figura, se usa un clásico modelo de cascada.

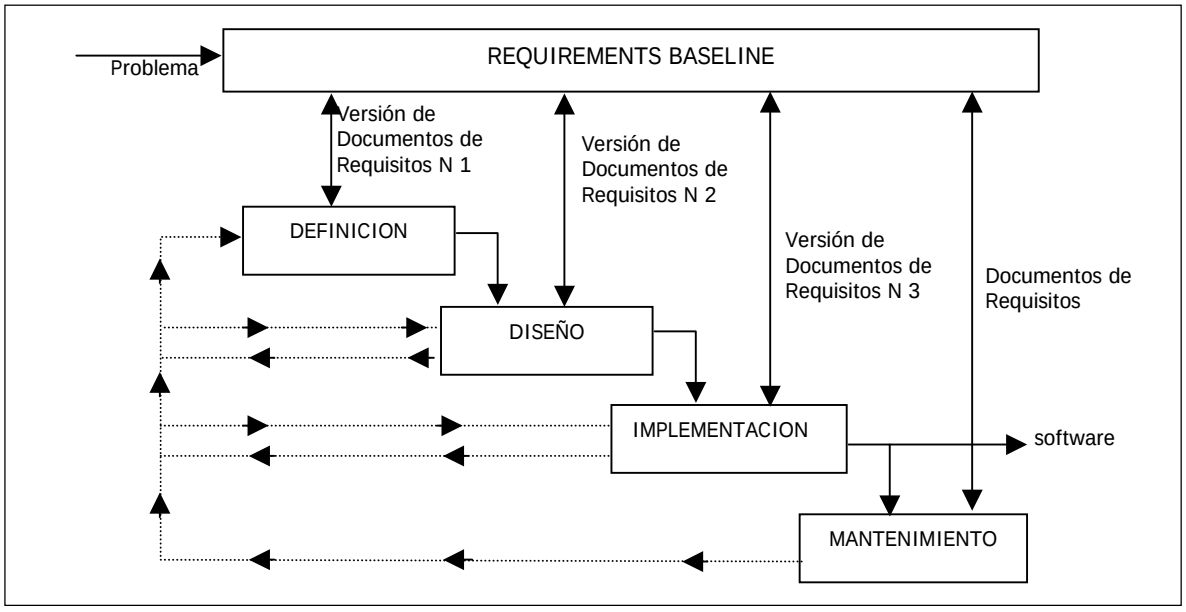


Fig. 2.1 Relación entre *Requirements Baseline* y el proceso de desarrollo de software [Leite'95]

La *Requirements Baseline* está compuesta por cinco vistas:

- LEL, vista del modelo léxico

[2] Un stakeholder es cualquier individuo que puede intercambiar información sobre el sistema, sus restricciones de implementación o del dominio del problema[Potts'94]. En esta tesis se usa este término en lugar de usuario, ya que tiene un mayor alcance.

- SMV, vista del modelo de escenarios
- BMV, vista del modelo básico
- HV, vista de hipertexto
- CV, vista de configuración

La vista léxica es un meta-modelo diseñado para ayudar a la elicitación del lenguaje usado en el Macrosistema. La vista básica está soportada por diagramas de entidad-relación que representan los requisitos externos propuestos por el cliente del Macrosistema. La vista de escenarios describe situaciones del comportamiento del Macrosistema. Es importante resaltar que las vistas de Hipertexto y de Configuración son ortogonales a las otras tres vistas. Son un soporte de servicios indispensable para garantizar el acceso a la información almacenada (HV) y el seguimiento de los modelos y sus revisiones (CV). A continuación describiremos en detalle cada una de las vistas, poniendo mayor énfasis en las vistas LEL y SMV.

2.2 LEL, la vista del modelo léxico

El Léxico extendido del Lenguaje (LEL) es una representación de los símbolos en el lenguaje del dominio del problema, que intenta capturar el vocabulario de una aplicación y su entorno. Su objetivo principal es que el ingeniero de software entienda el lenguaje que habla el stakeholder, entendiendo los términos que utiliza, sin preocuparse por entender el problema. Cada símbolo tiene uno o más nombres que lo identifican, llamados sinónimos. El significado de los símbolos se representa mediante dos tipos de descripciones: nociones e impactos. La noción indica que es, describiendo la denotación de la palabra o frase. El impacto describe como repercute en el sistema, es decir su connotación. Estas descripciones se construyen siguiendo dos principios importantes: maximizar el uso de términos del LEL utilizados (principio de circularidad) y minimizar el uso de símbolos externos al dominio destino (principio de mínimo vocabulario). Mediante estos dos principios se logra un conjunto autocontenido de símbolos, altamente vinculado, y por lo tanto plausible de ser representado como un documento hipertextual. La Fig. 2.2, extraída de [Leite'00] muestra el patrón de descripción del LEL.

LEL = representación de los símbolos en el lenguaje del dominio de aplicación
sintaxis: $\{Símbolo\}_1^N$
Símbolo = entrada del LEL con un significado especial en el dominio de aplicación
sintaxis: $\{Nombre\}_1^N + \{Noción\}_1^N + \{impacto\}_1^N$
Nombre = identificador del símbolo. Más de uno representa sinónimos
sintaxis: Palabra / Frase
Noción = denotación del símbolo. Debe estar expresado usando referencias a otros Símbolos y usando Vocabulario Mínimo
sintaxis: Sentencia
Impacto = connotación del Símbolo. Debe estar expresado usando referencias a otros Símbolos y usando Vocabulario Mínimo
sintaxis: Sentencia
donde Sentencia es compuesta por Signos y No-Signos, estos últimos pertenecientes al Vocabulario Mínimo

Fig. 2.2: meta-modelo del LEL [Leite'00]

Los símbolos del LEL, definen objetos (entidades pasivas), sujetos (entidades activas), verbos y estados. En [Hadad '97] se expresa la utilidad de adaptar esta clasificación al Universo del Discurso de la aplicación en estudio o la de adoptar una procedente de este. Estas nuevas categorías pueden corresponder tanto a categorías de especialización (refinamiento de alguna de las categorías básicas) como a categorías ortogonales a las básicas. Ambas clasificaciones ayudan a homogeneizar las descripciones,

especialmente cuando el equipo de trabajo es numeroso. Según sea el tipo de símbolo, sus nociones e impactos tienen una semántica diferente, como se indica en la figura 2.3, basada en las heurísticas de construcción del LEL [Leite'97b]:

Sujeto	*Nociones:* describen quien es el sujeto.
	Impactos: registran acciones ejecutadas por el sujeto
Objeto	*Nociones:* definen al objeto e identifica a otros términos con los cuales el objeto tiene algún tipo de relación.
	Impactos: describen las acciones que pueden ser aplicadas al objeto.
Verbo	*Nociones:* describen quien ejecuta la acción, cuando ocurre, y cuales son los procedimientos involucrados.
	Impactos: describen las restricciones sobre la acción, cuáles son las acciones desencadenadas en el ambiente y las nuevas situaciones que aparecen como resultado de la acción.
Estado	*Nociones:* describen que significa y que acciones pueden desencadenarse como consecuencia de ese estado.
	Impactos : describen otras situaciones y acciones relacionadas

Fig. 2.3 Heurísticas para la definición de los símbolos del modelo léxico.

Las siguientes figuras muestran ejemplos, extraídas del caso de estudio del Círculo de Ahorro (Anexo I).

Adjudicatario
Nociones:
- Es el adherente que ganó el sorteo y aceptó el bien tipo, teniendo las cuotas al día, ó ganó la licitación.

Impacto:
- El adjudicatario paga el derecho de adjudicación
- El adjudicatario puede cambiar el bien tipo adjudicado, pagando la diferencia si el nuevo bien tipo es de mayor valor que el bien adjudicado, ó computándose la diferencia para cancelar las últimas cuotas del plan, en caso de ser un bien de menor valor. La administradora puede negarse a realizar el cambio.
- Elegir aseguradora para el seguro de bien tipo.

Fig. 2.4: Término del LEL correspondiente a un sujeto

Bien tipo | Bien
Nociones:
- Vehículo que desean obtener los adherentes de un determinado plan de ahorro.

Impacto:
- Adjudicado a un adherente
- La administradora lo entrega a un adjudicatario.

Fig. 2.5: Término del LEL correspondiente a un objeto

2.3 SMV, vista del modelo de escenarios

El modelo de escenarios es una combinación de las ideas presentadas en [Zorman'95; Jacobson'92; Rubin'92; Potts'95] con el agregado de nuevos conceptos:

- Un escenario describe situaciones del Macrosistema.
- Un escenario evoluciona durante el proceso de desarrollo del software siguiendo la filosofía de la *Requirements Baseline* a la cual pertenece.
- Los escenarios están naturalmente conectados al LEL.
- Un escenario describe situaciones, poniendo énfasis en el comportamiento del

Macrosistema. Usa lenguaje natural para su representación.

Cada escenario tiene la siguiente estructura:

- Título: identifica al escenario, puede ser uno o varios.
- Objetivo: meta a lograr en el Macrosistema.
- Contexto: describe la ubicación geográfica y temporal del escenario, así como un estado inicial o precondición.
- Recursos: son los medios de soporte, dispositivos que se necesita estén disponibles en el escenario.
- Actores: son las personas o estructuras de organización que tienen un rol en el escenario.
- Episodios: son una serie ordenada de sentencias escritas de manera simple, que posibilitan la descripción de comportamiento. Pueden ser opcionales, condicionales o simples, y estar agrupados según su forma de ocurrencia en grupos secuenciales o no secuenciales. Para cada escenario se prevé la descripción de excepciones: causas y soluciones a situaciones que discontinúan su evolución natural, e impiden el cumplimiento del objetivo. Estas generalmente reflejan la falta o mal funcionamiento de un recurso. El tratamiento de la excepción puede estar dado por un escenario.

Para algunos componentes del escenario pueden describirse restricciones, es decir requerimientos de ámbito o calidad referidos al componente. Las restricciones permiten especificar aspectos referidos a requisitos no funcionales. Un aspecto importante es la posibilidad de que un episodio pueda ser un escenario en sí mismo. Entonces, se tienen relaciones de jerarquía entre escenarios y subescenarios, y relaciones de integración entre escenarios. Esto induce una estructura jerárquica en la vista de escenarios que facilita su comprensión y especificación. La Fig. 2.6, extraída de [Leite'97] muestra el diagrama de entidad-relación para el modelo de escenarios, y la Fig. 2.7 [Hadad'99] describe cada entidad del diagrama anterior, así como su sintaxis.

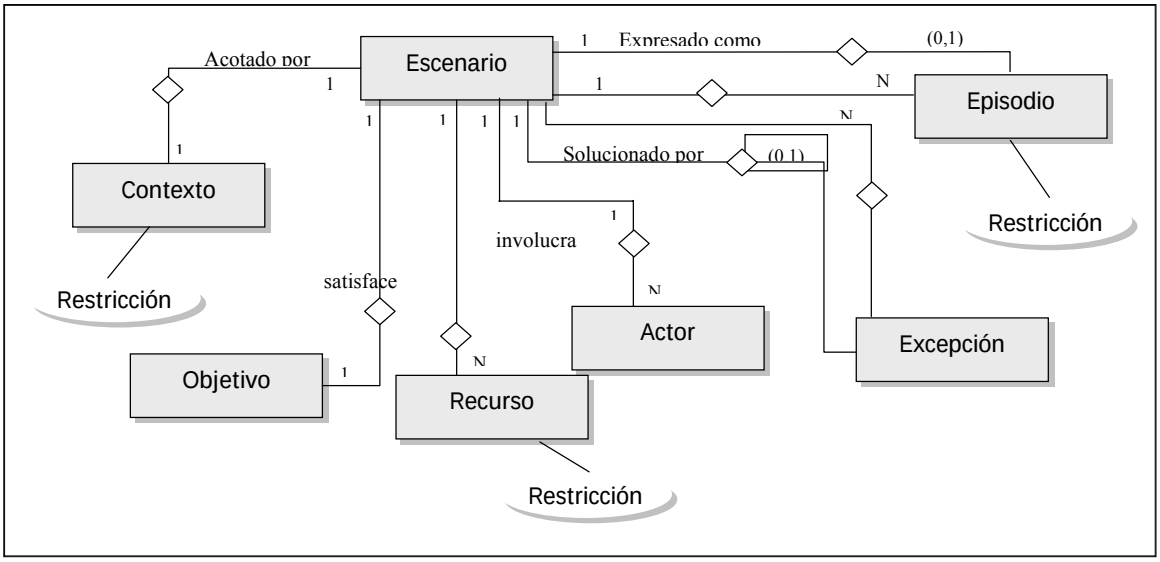


Fig. 2.6: Diagrama de entidad-relación para el modelo de escenarios [Leite'97]

Escenarios: Título + Objetivo + Contexto + {Recursos}$_1^N$ + {Actores}$_1^N$ + {Episodios}$_2^N$ + {Excepciones}
Título: Frase | ([Actor | Recurso] + Verbo + Predicado)
Objetivo: [Sujeto] + Verbo + Predicado
Contexto: {Ubicación Geográfica} + {Ubicación Temporal} + {Precondición}
donde Ubicación Geográfica es: Frase + {Restricción}
donde Ubicación Temporal es: Frase + {Restricción}
donde Precondición es: [Sujeto | Actor | Recurso] + Verbo + Predicado + {Restricción}
Recursos: Nombre + {Restricción}
Actores: Nombre
Episodios:

<episodios>::=<serie grupo> | <serie episodios>
<serie grupo>::= <grupo> <grupo> | <grupo no secuencial> | <serie grupo> <grupo>
<grupo>::= <grupo secuencial> | <grupo no secuencial>
<grupo secuencial>::= <sentencia básica> | <grupo secuencial> <sentencia básica>
<grupo no secuencial>::= **#**<serie episodios> **#**
<serie episodios>::=<sentencia básica><sentencia básica>|<serie episodios><sentencia básica>
<sentencia básica>::= <sentencia simple>| <sentencia condicional> | <sentencia optativa>
<sentencia simple>::= <sentencia episodio> **CR**
<sentencia condicional>::= **Si** <condición> <conector> <sentencia episodio> **CR**
<conector>::= **Entonces** | **,**
<sentencia optativa>::= **[** <sentencia episodio> **] CR**

donde <sentencia episodio> tiene la estructura:
(([Actor | Recurso] + Verbo + Predicado) | ([Actor | Recurso] + [Verbo] + Título)) + {Restricción}
Excepciones: (Causa**)** [Solución]
donde Causa es: Frase | ([Sujeto | Actor | Recurso] + Verbo + Predicado)
donde Solución es: Título
Atributo:
Restricción: ([Sujeto | Actor | Recurso] + [**No**] **Debe** + Verbo + Predicado) | Frase

Fig. 2.7 Sintaxis de las Entidades del modelo de Escenarios [Hadad'99].

La Fig. 2.8 presenta un ejemplo de un escenario, extraído del caso de estudio Círculo de Ahorro (Anexo I), las palabras subrayadas corresponden a símbolos del LEL.

Armar un grupo
Objetivo: Formar un nuevo grupo de adherentes a un plan de ahorro.
Contexto: Se tienen tantas solicitudes de adhesión como miembros requiere el grupo.
Actores: Administradora
Recursos:

- Planillas de adhesión**.**

Episodios:

1. Se arma la lista de los miembros del nuevo grupo incluyendo sus datos y los números de orden asignados a sus planillas de adhesión.
2. Se registra al nuevo grupo para participar en la próxima adjudicación.
3. Se comunica fehacientemente la aceptación a los nuevos adherentes.
4. Se envían los cupones de pago correspondientes al plan de ahorro a todos los adherentes.

Fig. 2.8: Ejemplo de escenario

2.4 BMV, la vista del modelo básico

Esta vista apunta a registrar los requisitos (los cuales no consideran restricciones de software o hardware) tal y como son expresados por los clientes del macrosistema, del cual es software será una de las partes. Se encarga de registrar información concerniente a las acciones de los clientes. Estas acciones son sus actividades diarias, que pueden expresarse mediante un verbo en infinitivo. Los eventos externos expresan las necesidades que originan el problema y que deben ser solucionadas por el sistema, así como las interacciones percibidas entre el sistema y el Universo del Discurso. El lenguaje de representación de esta vista se basa en el framework de entidad-relación de Elmasri [Elmasri'89]. Cada diagrama de la vista muestra una acción concreta que toma un cliente, las sub-acciones que la componen y los eventos externos que las disparan. Cada evento debe estar asociado a una acción existente. También permite especificar qué sucesos estimulan a estos eventos, así como la información que el sistema produce y consume. Los atributos más importantes de las entidades del modelo son las Restricciones (requerimientos de ámbito o calidad) y los Diagnósticos (observaciones que marcan problemas). Todas las descripciones son escritas en lenguaje natural, utilizando términos del *LEL*. Los eventos externos tienen un ordenamiento temporal, que se hace explícito mediante un diagrama de estructuras de Jackson [Jackson 95]. La Fig. 2.9 muestra un diagrama de entidad-relación que describe la vista del modelo básico, y la Fig. 2.10 describe cada entidad en más detalle, así como su sintaxis.

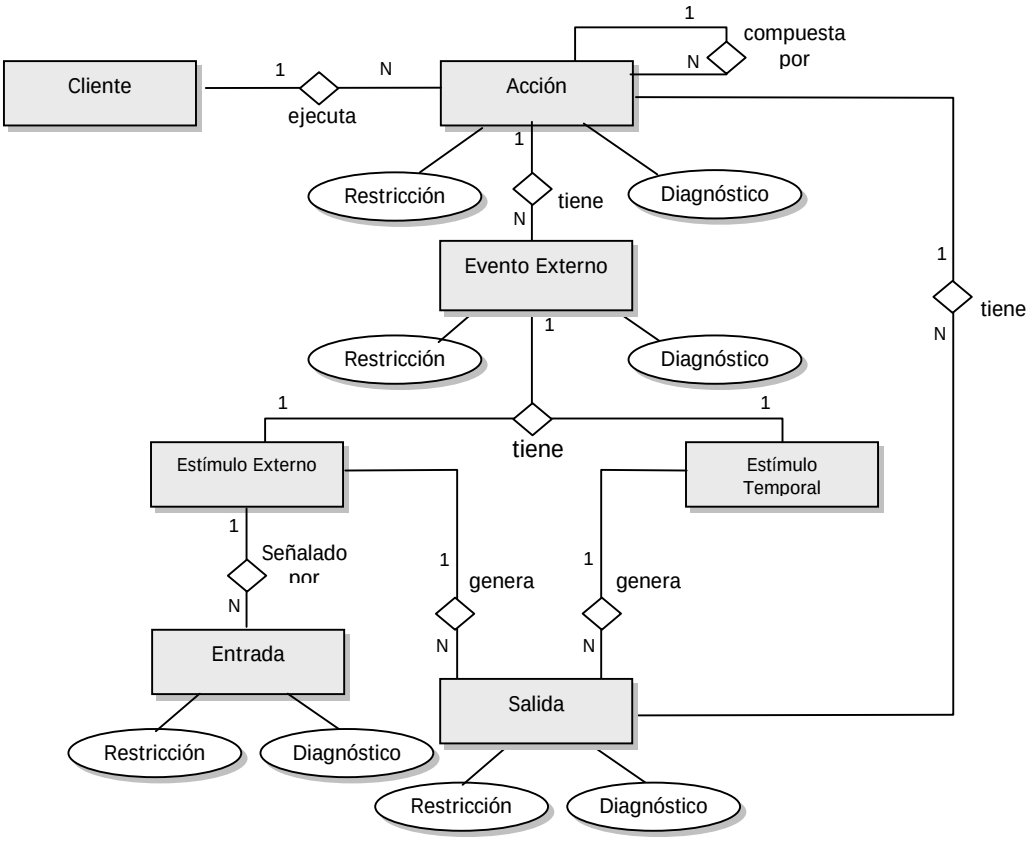


Fig. 2.9: ERD para el Modelo de la Vista Básica[Leite'95]

Entidades:
Cliente: persona o grupo de personas responsable de una Acción.
Sintaxis: Nombre + Ubicación
Acción: es la acción concreta que toma el cliente en su trabajo. Puede ser descompuesta en subacciones con la misma estructura.
Sintaxis: Verbo Infinitivo + Predicado + [Id Acción Padre] + {Restricción} + {Diagnóstico}
Evento externo: ocurre en el Universo del Discurso y debería producir una respuesta del sistema. Cada evento debe estar asociado a una Acción existente.
Sintaxis: [Id Estímulo Temporal | Id Estímulo Externo] + Id Acción + [Evento externo precedente] + {Restricción} + {Diagnóstico}
Estímulo Temporal: instantes de tiempo en los cuales el sistema deberá reaccionar independientemente de una Entrada.
Sintaxis: Referencia Temporal + Sujeto + Verbo + Predicado
Estímulo Externo: suceso en el Universo del Discurso que requerirá una reacción del sistema.
Sintaxis: Sujeto + Verbo + Predicado
Salida: estructura de información que será producida por el sistema.
Sintaxis: Nombre + (Descripción) + {Restricción} + {Diagnóstico}
Entrada: estructura de información consumida por el sistema.
Sintaxis: Nombre + (Descripción) + {Restricción} + {Diagnóstico}
Atributos:
Restricción: requerimiento de ámbito o calidad referido a una entidad.
Sintaxis: Debe + Verbo + Predicado
Diagnóstico: observación que describe un problema relacionado a una entidad.
Sintaxis: sentencia corta, sin estructura fija.

Fig. 2.10 Entidades y Atributos del Modelo Básico y su sintaxis[Petersen'00].

2.5. HV, la Vista de Hipertexto

Esta vista es ortogonal a la BMV, LEL y SMV. Trabaja como un integrador de las mismas, habilitando la definición de vínculos dentro de la misma vista y entre las tres. Estos vínculos están determinados por las relaciones naturales entre las vistas y por el uso de un vocabulario controlado. En esta red hipertexto, los nodos corresponden a cada componente de las distintas vistas antes mencionadas: símbolos del LEL, escenarios en la SMV y cada una de las entidades en la BMV.

En el caso del LEL, que es una red hipertextual en sí misma, los vínculos están asegurados por la utilización del principio de circularidad. Entonces, cada referencia a un símbolo ya existente constituye un vínculo a la definición de ese símbolo. La vinculación en la BMV está dada por el uso de símbolos del LEL cuando se especifican las entidades de cada diagrama, y por las relaciones entre las entidades del diagrama.

Los vínculos en la SMV permiten explorar las relaciones entre escenarios, y entre los componentes de cada escenario y otros componentes de la *Baseline*. Los vínculos entre los componentes de cada escenario son derivados de 3 formas: a partir de relaciones estructurales existentes, a partir del uso de información en el LEL, y a partir de un pedido del ingeniero de aplicación. Como la cantidad de nodos y relaciones puede ser de un tamaño considerable, el modelo de hipertexto está organizado en contextos navegacionales, esto es, en conjuntos de nodos que están muy relacionados y que típicamente se recorrerán juntos. Estos contextos pueden estar constituidos por los nodos conectados por los tres tipos de vínculos antes mencionados, o bien porque los nodos tienen características comunes (utilizan un mismo recurso, involucran a un mismo actor, o son distintas versiones de un mismo escenario). Los escenarios también están vinculados a nodos que representan entradas del LEL o de la BMV.

La Fig. 2.11, extraída de [Petersen'99] muestra una parte de la red hipertextual correspondiente al mismo caso de estudio.

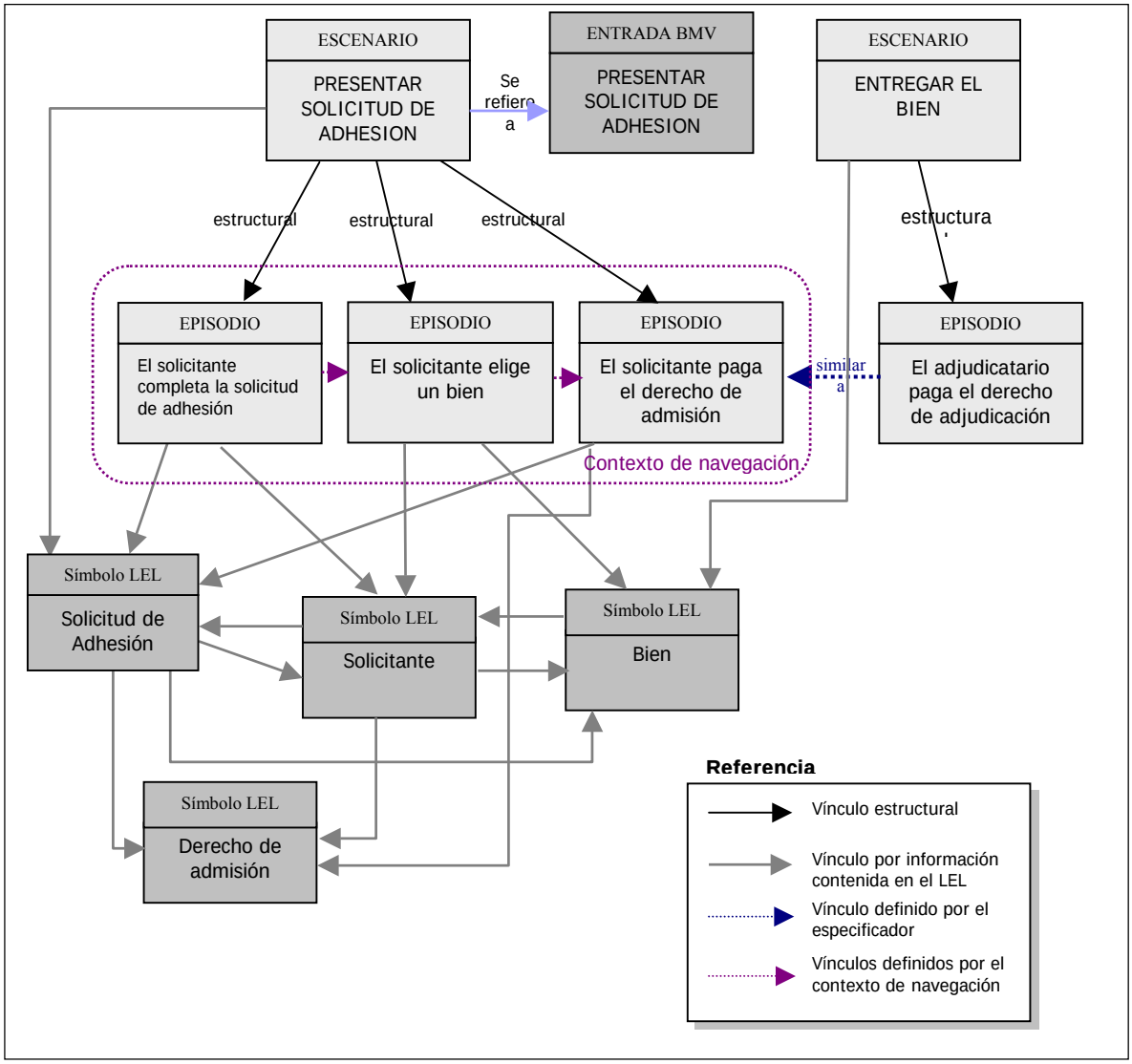


Fig. 2.11: Ejemplo Parcial de la Vista de Hipertexto [Petersen'99].

2.6 CV, la Vista de Configuración

Esta vista es el sistema de versionado de la *Requirements Baseline*. Es esencial para mantener la traceability de los modelos y sus revisiones. El LEL, la BMV y la SMV están sujetos a un control de configuración y versión. Su consistencia se garantiza mediante restricciones determinadas por cada modelo. Entonces, cada operación de cambio causa la ejecución de un proceso que controla la consistencia de una configuración dada. Si, por ejemplo, se excluye un episodio de un escenario, y ese episodio estaba descrito por un subescenario, este proceso controla que el subescenario sea excluido también.
Este versionado mantiene la historia de los cambios, almacenando para cada uno la siguiente información: fecha, hora, persona que realiza el cambio, razones que motivan el cambio (motivo, fecha y autorización) y tipo de cambio (inclusión, modificación o exclusión). La Fig. 2.12, extraída de [Leite'95] muestra el esquema de una ficha de versionado que se completan para cada versión de cada entidad de las vistas, y la Fig. 2.13 [Petersen'99] la primera versión de un escenario, tomado del caso de estudio.

VERSIÓN N
Fecha:
Hora:
Usuario: Persona que realiza el cambio.
Causa del cambio: razón del cambio.
Fecha de causa:
Autorización: Persona que autoriza el cambio.
Tipo de cambio: Inclusión | Modificación | Exclusión

Fig. 2.12: Ficha de Versionado de la Vista de Configuración [Leite'95].

La vista de configuración permite mantener la evolución de cada entidad de las vistas. En [Breitman'98], se profundiza el estudio de este aspecto evolutivo de los escenarios y la utilidad de la vista de configuración mediante su aplicación a un caso de estudio. La Figura 2.14 [Petersen'99] muestra dos versiones de un mismo escenario.

Fecha: 16/11/1997 VERSIÓN 1
Hora: 16:25
Usuario: Carmen Leonardi
Causa del cambio: Caso de Estudio Círculo de Ahorro
Fecha de causa: 11/11/1997
Autorización: Laura Rivero
Tipo de cambio: Inclusión

Armar un grupo
Objetivo: Formar un nuevo grupo de adherentes a un plan de ahorro.
Contexto: Se tienen tantas solicitudes de adhesión como miembros requiere el grupo.
Actores: Administradora
Recursos:

- Planillas de adhesión**.**

Episodios:

1. Se arma la lista de los miembros del nuevo grupo incluyendo sus datos y los números de orden asignados a sus planillas de adhesión.
2. Se registra al nuevo grupo para participar en la próxima adjudicación.
3. Se comunica fehacientemente la aceptación a los nuevos adherentes.
4. Se envían los cupones de pago correspondientes al plan de ahorro a todos los adherentes.

Fig. 2.13: Ejemplo de la Vista de Configuración [Petersen'99].

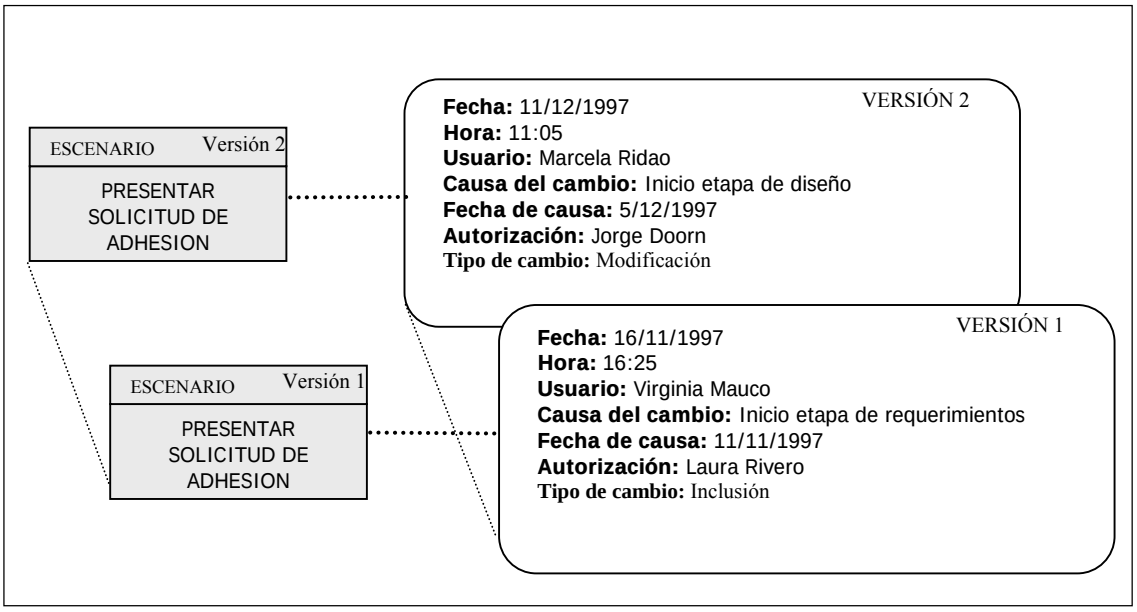


Fig. 2.14: Evolución de un escenario, desde la vista de configuración [Petersen'99].

2.7 Proceso de Construcción de LEL y Escenarios

La construcción del LEL y Escenarios se facilita cuando se sigue un proceso estructurado. Los trabajos de [Hadad'97; Doorn' 98; Hadad'99; Leite'00] sugieren organizar este proceso en dos grandes etapas:

1. Construcción del LEL,
2. Construcción de los Escenarios a partir del LEL.

Estas etapas del proceso se explican brevemente en las siguientes secciones.

2.6.1 Proceso de Construcción del LEL

El modelo léxico se construye en 6 etapas interdependientes, en algunos casos simultáneas, como se muestra en la Fig. 2.15 [Hadad'97]. Estas son:

a. Entrevistas
b. Generación de la lista de símbolos
c. Clasificación de los símbolos
d. Descripción de los símbolos
e. Validación con los clientes
f. Control del LEL

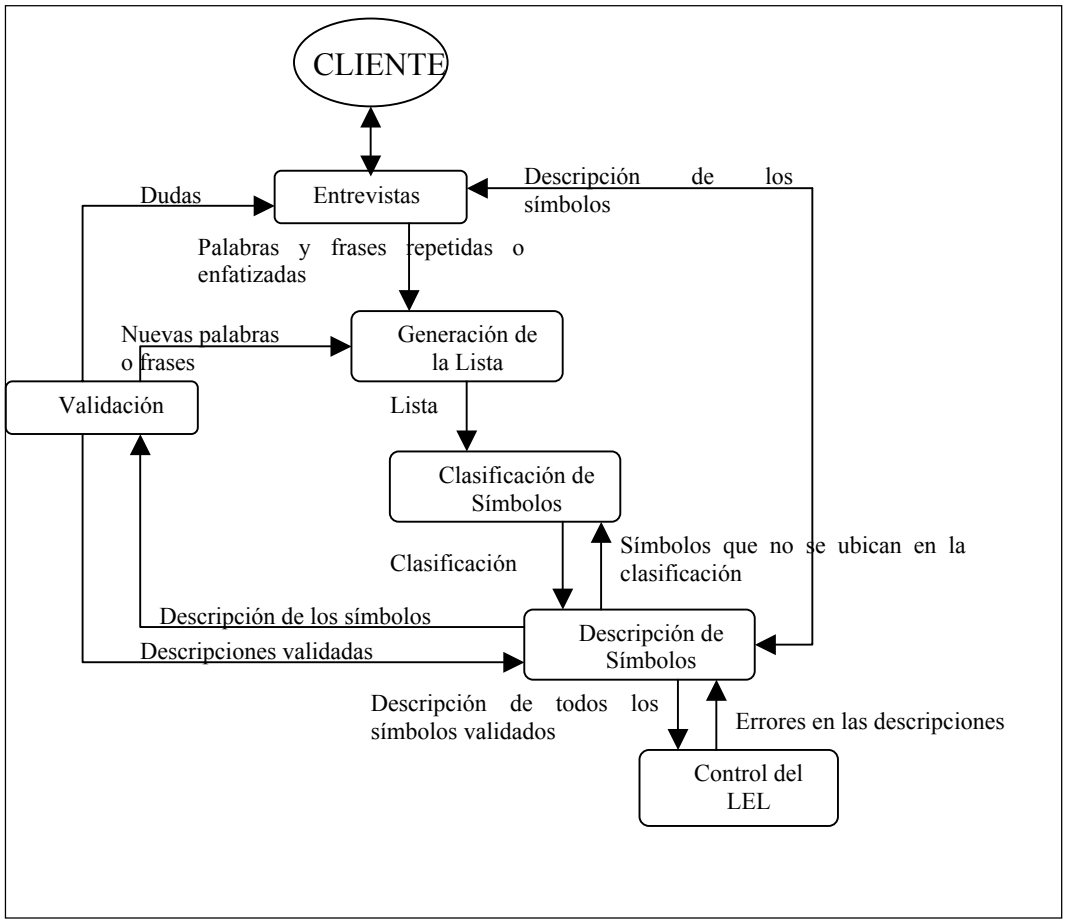


Fig. 2.15: Etapas para la construcción del LEL [Hadad'97].

El proceso de construcción del LEL se lleva a cabo mediante el desarrollo de varias entrevistas con el stakeholder (a). En las primeras se deja hablar libremente al stakeholder; a medida que avanza el proceso, éstas adquieren mayor estructura y se dedican a validar los símbolos ya existentes, además de reconocer a nuevos. Luego de la primera entrevista se genera una lista de símbolos candidatos (b), con las palabras que el stakeholder utiliza más frecuentemente. Esta lista de símbolos candidatos también puede construirse mediante la lectura de documentos. A continuación, estos símbolos son clasificados (c) según la clasificación general definida anteriormente o bien una propia. La etapa siguiente (d), de descripción de los símbolos, consiste en determinar su noción e impacto, que luego se validarán con los clientes. Como resultado de las entrevistas de validación (e), la lista de símbolos se transforma en definitiva, y se ratifica y rectifica el conocimiento adquirido durante entrevistas anteriores. También permiten detectar información faltante. Finalmente, en la etapa de control (f) se termina de unificar la sintaxis. Esta etapa es la que asegura un LEL consistente y homogéneo.

2.7.2 Proceso de Construcción de Escenarios

El proceso de construcción de escenarios, según se explica en [Hadad'99], puede basarse en tres enfoques diferentes: bottom-up, top-down o middle-out. El enfoque bottom-up comienza identificando acciones de granularidad fina, episodios en nuestro caso, para luego armar subescenarios, y finalmente integrarlos en escenarios. El enfoque top-down comienza armando uno o muy pocos escenarios que representen todo el sistema, y luego los refina para tener una sucesión de escenarios con un creciente nivel de detalle. El enfoque middle-out(es el utilizado actualmente en esta estrategia) combina los dos

enfoques anteriores. Comienza construyendo escenarios a partir del LEL, proceso durante el cual se factorizan algunas partes comunes para crear subescenarios. Los subescenarios también se generan cuando uno o más episodios de un escenario merecen un tratamiento independiente. Esta constituye la parte top-down del proceso. En este punto, los ingenieros necesitan tener una visión global de los escenarios, entonces construyen escenarios integradores a partir de los escenarios existentes. Esta es la parte bottom-up del proceso.

Entre ambas etapas del proceso de construcción de escenarios se lleva a cabo una tercera etapa: la de inspección de los escenarios obtenidos de la derivación. Esto implica la verificación de consistencia para los mismos y la validación con los clientes. Esta etapa permite detectar errores y omisiones en los escenarios, y corregirlos. A continuación se detallan el conjunto de heurísticas que guían ambas partes del proceso, así como las de inspección de escenarios.

2.7.2.1 Derivación de escenarios a partir del LEL

En [Hadad'97; Leite'00] se propone la construcción de los escenarios a partir de la información contenida en el LEL, mediante el uso de una serie de heurísticas. Este proceso produce un primer conjunto de escenarios, que se completarán luego de nuevas entrevistas con los clientes.
El proceso de construcción, ilustrado en la Fig.2.16, consta de las siguientes etapas:

a. Identificación de los actores de la aplicación.
b. Generación de la lista de escenarios candidatos, a partir de los actores principales.
c. Descripción de los escenarios candidatos, provenientes de los actores principales.
d. Ampliación de la lista de escenarios candidatos, a partir de los actores secundarios.
e. Descripción de los escenarios candidatos, provenientes de actores secundarios.
f. Revisión de los escenarios.
g. Validación de escenarios.

El proceso de derivación de escenarios comienza con la identificación de los símbolos del *LEL* (a) que representan a los actores del Universo del Discurso. Todos ellos pertenecen a la clasificación SUJETO en el *LEL*. Se identifica cuales son actores principales y cuales secundarios, según interactúen directamente con la aplicación o sólo reciban o brinden información, sin ejecutar acciones directas sobre la aplicación. A partir de cada impacto del *LEL* de cada actor principal se genera un escenario candidato (b), cuyo nombre será la acción del impacto con su verbo en infinitivo. De la lista resultante deberán eliminarse los escenarios candidatos repetidos. A continuación, deberá describirse cada escenario de la lista candidata (c), según el esquema propuesto en la Fig. 2.16 [Hadad'97]. Para esto, se utiliza la información existente en el LEL para los símbolos utilizados para describir el impacto. En este paso podrán surgir dudas, que deberán registrarse para ser eliminadas en una futura entrevista con el stakeholder.

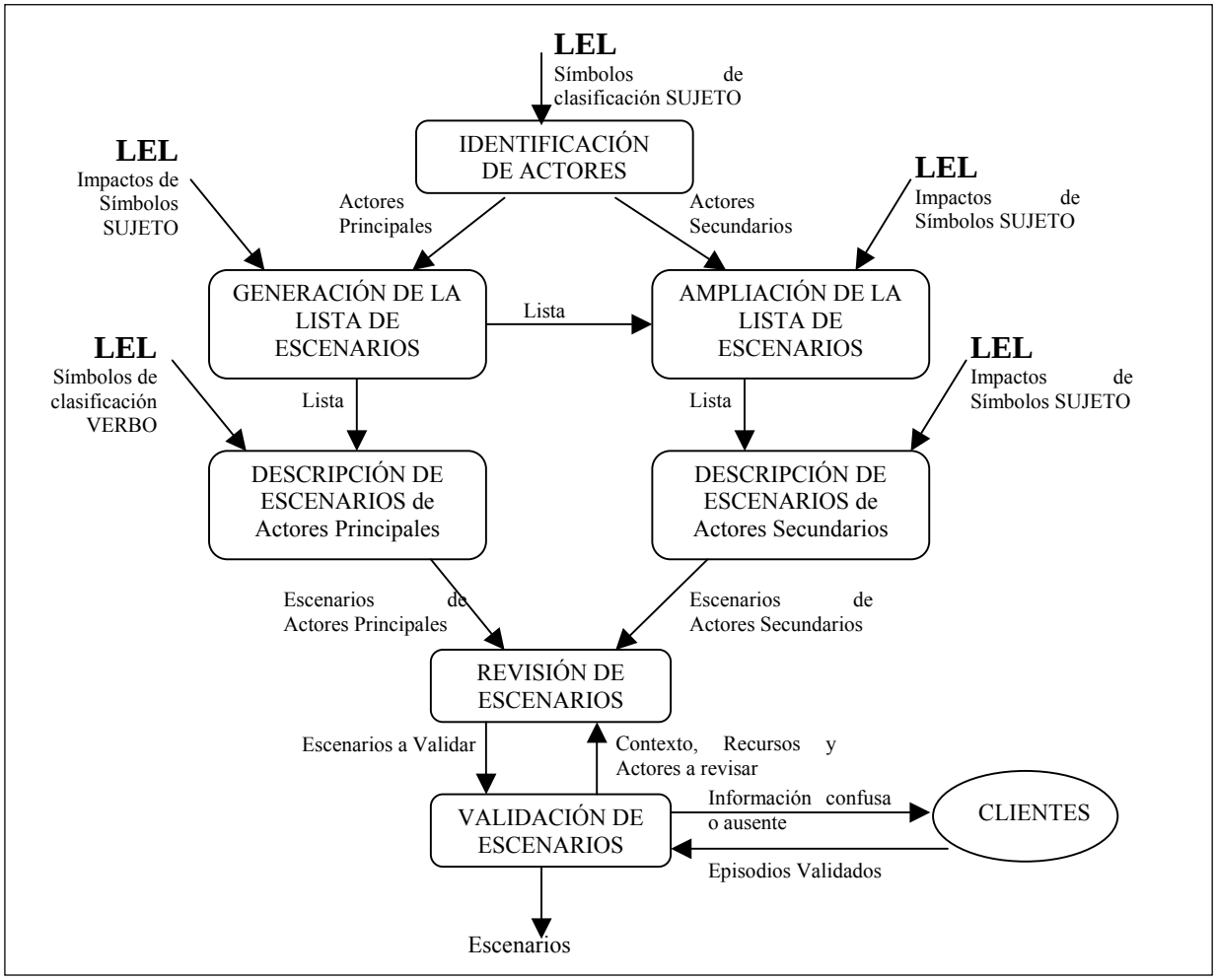


Fig. 2.16: Etapas para la construcción de escenarios a partir del LEL [Hadad'97].

Los siguientes pasos consisten en aumentar la lista de escenarios candidatos con los escenarios derivados a partir de los impactos de cada actor secundario (d) y describirlos (e). Una vez que se han descrito todos los escenarios de la lista de escenarios candidatos, se deberán revisar los escenarios candidatos (f) para detectar subescenarios que no hayan sido definidos, o conjuntos de episodios que correspondan a un escenario de un actor secundario. También puede ser necesario unificar escenarios con episodios u objetivo comunes, definir restricciones a los episodios o excepciones a los escenarios. Luego de esta validación, se obtiene la lista de escenarios a validar con el cliente. La validación de los escenarios con los clientes (g) permite detectar errores u omisiones, o ampliar información en los episodios. Esta etapa no sólo valida episodios, sino que requiere revisar también el contexto, los actores y los recursos de cada escenario. Como resultado de ella pueden surgir correcciones, que requieran nuevas validaciones. A partir de este punto se obtiene la lista definitiva de escenarios y sus descripciones.

2.7.2.2. Proceso de Construcción de Escenarios Integradores

Este proceso aumenta el conjunto de escenarios con uno o varios escenarios integradores. Se recomienda que los escenarios y subescenarios con los que se inicia el proceso hayan sido derivados a partir del LEL, inspeccionados con las heurísticas anteriores y validados por los clientes. El proceso de construcción de escenarios

integradores comienza con la construcción de jerarquías de escenarios, considerando los escenarios y subescenarios existentes. Luego, se arman grupos con los escenarios pertenecientes al nivel más alto de la jerarquía. Para esto se utiliza la información del contexto y del objetivo, y se genera una lista candidata de grupos de escenarios. A continuación, se debe analizar cada grupo de escenarios para generar la lista final de grupos que representarán, cada uno, un potencial escenario integrador. Cabe destacar que un mismo escenario puede pertenecer a más de un grupo. Para cada grupo, se establecen las relaciones de orden (tanto secuenciales como de paralelismo) entre los escenarios involucrados, teniendo en cuenta las precondiciones de cada escenario. Los escenarios se convierten entonces en episodios del escenario integrador que resultará de cada grupo. Luego se debe completar el escenario integrador: se deberá definir un nombre significativo para el título, el objetivo analizando los objetivos de cada escenario del grupo y el contexto combinando los contextos de los escenarios en el grupo. Los actores y recursos se obtienen a partir de la unión de actores y recursos de los escenarios del grupo. Las restricciones de cada escenario pasan a ser restricciones de los episodios del integrador, y las excepciones de cada escenario se trasladan al escenario integrador, mencionando los episodios que pueden dispararlas. Finalmente, se buscan posibles escenarios aislados no utilizados en ningún escenario integrador, y en el caso que los hubiera, deberán analizarse para detectar si corresponde a una excepción que no se había encontrado o si está fuera del dominio de aplicación.

Los escenarios integradores contribuyen al proceso de inspección, ya que permiten fácilmente detectar superposiciones y omisiones entre escenarios. Como son el último paso del proceso de construcción de escenarios, aprovechan el conocimiento del Macrosistema adquirido por el ingeniero de software en los pasos anteriores, y requieren menor mantenimiento.

2.7.3. Herramientas automatizadas que permiten la construcción de escenarios y LEL

Actualmente existen dos herramientas que automatizan la construcción de los modelos de LEL y escenarios: *HeAR* [Petersen'00] es una herramienta que permite la autoría y navegación de los modelos de escenarios y LEL siguiendo algunas de las heurísticas aquí presentadas, permitiendo además algunos aspectos de inspección y validación. *HeAR* es una instanciación del *MenDOR* [Petersen'99] un meta-modelo que permite la autoría y navegación de modelos de requisitos textuales escritos en lenguaje natural. *BaselineMentor* [Antonelli'99] es otra herramienta que permite la construcción y navegación del modelo de LEL y escenarios, asimismo incorpora algunas de las heurísticas de derivación de clases presentadas en esta tesis. En el capítulo 7 se detalla mas información.

Capítulo 3
Extensiones a la Baseline de Requerimientos: Incorporación de un modelo de Reglas de Negocio

3.1 Introducción

Las reglas de negocios constituyen un concepto importante en el proceso de definición de los requisitos de los sistemas de información computarizados, siendo consideradas como "parte del corazón" de cualquier organización [Gottesdiener'97]. Hasta el momento han sido utilizadas mayormente en el área de Base de Datos [Guide'96; Diaz'97; Ross'97], existiendo pocos trabajos en el área de Ingeniaría de Requisitos [Rosca'97].

Definimos a las Reglas de Negocios como sentencias sobre la forma en que la empresa realiza negocios. Reflejan las políticas de negocio, cuyas finalidades son: satisfacer los objetivos del negocio, satisfacer los clientes, hacer un buen uso de los recursos, y respetar las leyes o convenciones de la empresa. Las reglas de negocios pueden ser implementadas en un sistema de software como un requisito de ese sistema [Rosca'97]. Una regla de negocio pertenece a la organización y no al sistema de información de la organización o a la aplicación de software asociada. Por lo tanto las reglas pertenecen al *Universo de Discurso*.

Uno de los objetivos de esta tesis fue la incorporación de un modelo de reglas a la *Requirements Baseline*, cuya importancia se explica en la Sección 3.4. La identificación e inclusión del modelo de reglas en la *Baseline*, permite reflejar otros aspectos de la organización, las políticas, ayudando a la identificación de los requisitos de software. Por ejemplo, los escenarios pueden considerarse como posibles formas de llevar a cabo una determinada política. En el contexto de orientación a objetos, las Reglas del Negocio permiten mejorar el modelo de objetos, reflejando las políticas de la organización [Leonardi'98[a]].

3.2 Vista de Reglas de negocio

A partir de la definición presentada en la introducción y basándonos en los principales trabajos de reglas [Rosca'97; Guide'96; Herbst'96] que se aparecen en la literatura, se define el siguiente modelo conceptual de reglas [Leite'98]:

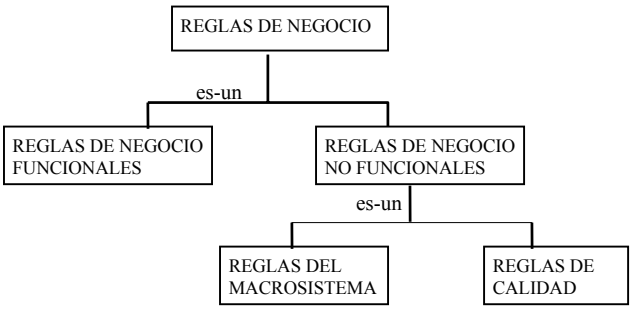


Fig. 3.1: Modelo de Reglas

A continuación se describe cada tipo de regla. La sintaxis de las reglas sigue nuestra estrategia de usar lenguaje natural restringido para representar los modelos en la *Requirements Baseline*. Asimismo las reglas se hayan conectadas naturalmente con el LEL.

3.2.1 Reglas No-Funcionales

Las reglas no funcionales pueden clasificarse en reglas generales o del Macrosistema y reglas de calidad.

3.2.1.1 Reglas del Macrosistema

Las reglas del Macrosistema describen políticas que restringen el comportamiento de la organización. La estructura es la siguiente:

- **[Propiedad] + Frase no-verbal + Relación + [Propiedad] + Frase no-verbal**

donde:

- Propiedad es una a frase que denota alguna característica o atributo de la frase no-verbal.
- La frase no-verbal es una frase que debiera ser una entrada en el Léxico extendido del Lenguaje (LEL) que pertenece al mismo Universo de Discurso que la regla
- La Relación es una frase verbal.
- Tanto la propiedad como la relación deben ser entradas en el LEL.
- Una combinación de una propiedad y una relación junto con la frase no-verbal pueden ser una entrada en el LEL.

Ejemplos:

El salario de un empleado senior debe ser mayor que el salario de un empleado junior.
(Propiedad) (Frase no-verbal) (Relación) (Propiedad) (Frase no-verbal)

Un estudiante de grado no puede realizar pedidos de libros de la biblioteca del
(Frase no-verbal) (Relación) (Propiedad)
departamento.
(Frase no-verbal)

3.2.1.2 Reglas de calidad

Las reglas de calidad son demandas de la organización sobre las características de sus productos. Usualmente reflejan políticas generales relacionadas a standard de calidad o expectativas de calidad de la organización. Su estructura esta definida de la siguiente forma:

- **Frase no-verbal + [PUEDE | NO PUEDE | DEBE | NO DEBE] + Frase Verbal + Propiedad + [DEBIDO a + Causa]**

donde
- Propiedad es una frase que denota una característica de calidad
- La frase no-verbal es una frase que debiera ser una entrada en el Léxico extendido del Lenguaje (LEL) que pertenece al mismo Universo de Discurso que la regla
- La propiedad debe ser entrada en el LEL.
- Una combinación de una propiedad junto con la frase no-verbal puede ser una entrada en el LEL.
- La causa es una sentencia, y debe tener una combinación de una frase no-verbal con una frase verbal.

Ejemplo:
La información sobre los datos de la reunión DEBEN estas disponibles tan rápido
(frase no-verbal) (frase verbal) (Propiedad)
como sea posible DEBIDO a que los participantes tiene que modificar sus agendas
(Causa)

3.2.2 Reglas Funcionales

Las reglas funcionales son políticas generales sobre la funcionalidad de la organización. El siguiente patrón describe su sintaxis:

- **Frase no-verbal + frase verbal + [frase no-verbal]**

donde:
- La frase no-verbal es una frase que debiera ser una entrada en el Léxico extendido del Lenguaje (LEL) que pertenece al mismo Universo de Discurso que la regla
- La frase verbal puede ser una entrada en el LEL.
- La combinación de una frase verbal con una frase no-verbal puede ser una entrada en la LEL.

Ejemplo:
Una reunión puede ser replaneada o cancelada.
(frase no-verbal) (frase verbal)

Los supervisores deben reportar al director general.
(frase no-verbal) (frase verbal) (frase no-verbal)

La organización soporta un plan de ayuda a sus empleados para reuniones científicas
(frase no-verbal) (frase verbal) (frase no-verbal) (frase no-verbal)

3.3 Proceso de Definición de Reglas

En esta sección se describe el proceso de definición de reglas de negocio presentado en [Leite'98; Leonardi'98[b]]. La primera consideración a tener en cuenta es que si bien las reglas que se definen son generales, están dentro del contexto del *Universo de Discurso* en estudio. Esto significa que, por ejemplo, si nuestro objetivo es producir una aplicación de un control de estudiantes de grado, identificaremos las reglas de la Universidad, pero no todas ellas. De esta forma, la primer "regla" básica es definir reglas dentro de los límites del Universo de Discurso. Idealmente la definición de reglas debe hacerse con una cercana y activa interacción con las gerencias de nivel medio y superior. Además, debe realizarse un ciclo de validación, es decir los administradores de la organización deben leer y aprobar el modelo de reglas identificado. Los pasos generales a seguir son descritos en la siguiente figura a través de SADT [Ross'80]:

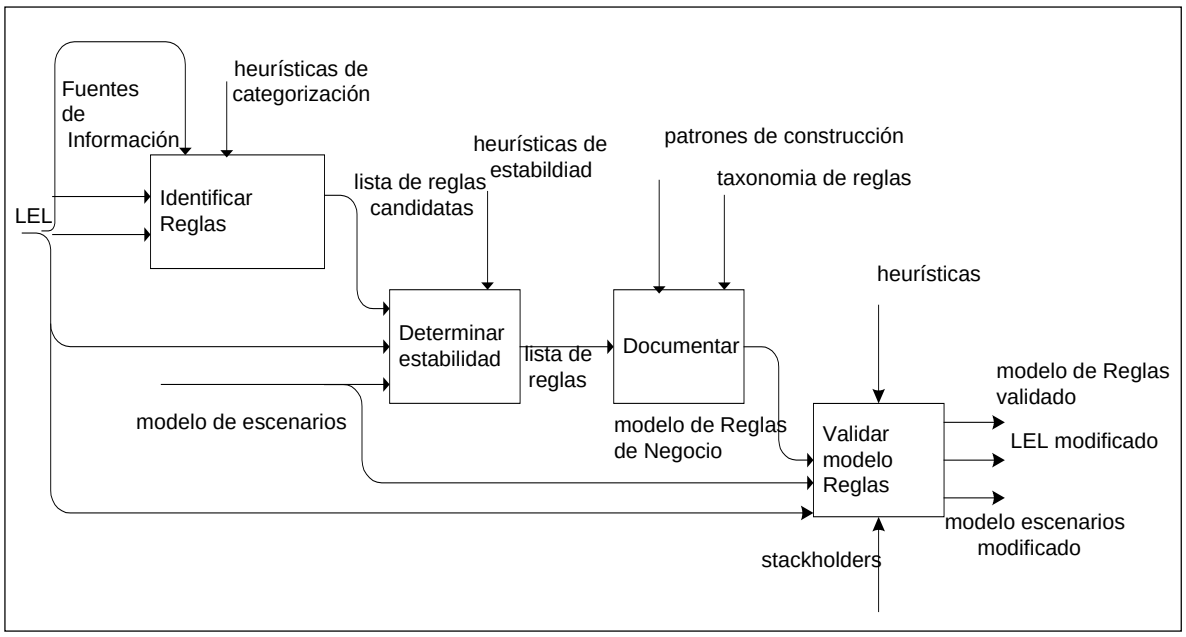


Fig. 3.2: Proceso de Construcción de Reglas

3.3.1 Identificación de Reglas

En esta etapa se utiliza el LEL construido siguiendo las heurísticas presentadas en el capítulo anterior y se determinan las fuentes de información para la identificación de las reglas. Generalmente los documentos de la organización constituyen la fuente de información más usual, principalmente si la organización tiene documentadas sus políticas, como por ejemplo manuales de calidad necesarios para una certificación ISO [Schmauch'95] y modelos de la empresa [Fiorini'97]. Si la empresa no cuenta con esta documentación, deben usarse otras técnicas para adquirir la información, por ejemplo, observación, entrevistas y reuniones.

A partir de la información dada en LEL y las fuentes de información se categorizan las sentencias de las mismas considerando su propósito en la organización: se busca aquellas sentencias que se refieren a los **límites**, **responsabilidades** y **derechos** de las entidades de la organización. Para decidir si una sentencia es una regla de negocio,

analizamos si es determinada por una decisión de la organización (por razones internas o externas) o es una sentencia inherente a la funcionalidad del Macrosistema (en cuyo caso no se considera una regla).

3.3.2 Determinación de la estabilidad de las Reglas

Se determina la estabilidad de cada una de las sentencias tomando el concepto de estabilidad de los trabajos de [Díaz'97]. Usamos el grado de estabilidad de la sentencia no sólo para determinar si es una regla, tal como se la utiliza en la propuesta de Díaz sino también para adjuntar esa información con la regla y utilizarla en etapas posteriores para tomar decisiones de diseño que permitan construir una arquitectura de solución que mejor se adapte a la naturaleza de cada una de las reglas. La propuesta de Díaz determina la estabilidad de una regla analizando el origen, arbitrariedad, impacto y complejidad. En nuestra propuesta este estudio es mejorado porque el LEL nos permite obtener mayor conocimiento de cada símbolo involucrado en la regla. A cada regla le asociamos la siguiente información:

Origen: En nuestra propuesta consideramos dos clases de origen: externo a la organización o interno a ella [Herbst'96]. Si el origen es externo se debe adquirir toda la información necesaria sobre la entidad que determinó la regla ya que seguramente esta entidad no pertenece al LEL porque no tiene una relación directa con el Macrosistema en estudio. Este tipo de reglas afecta a todas las aplicaciones del mismo estilo por lo que son candidatas a ser consideradas como "reglas del dominio". Es importante destacar este aspecto porque puede ser importante a la hora de tomar decisiones de diseño, por ejemplo en un diseño orientado a objetos pueden definirse "patrones" que contengan estas reglas. Si la regla es interna, debemos indicar el área de la organización que la ha determinado (en el caso que la organización cuente con una estructura dividida en áreas, secciones). El origen de la regla representa el responsable por la creación de la misma. Obtener información sobre el origen es útil desde el punto de vista de la *pre-traceability* de los requisitos [Gotel'94]. De esta forma se modela la fuente primaria de información involucrada en la definición, inclusión y refinamiento de cada regla en el modelo. También es importante determinar el grado de consenso de la definición de la regla, es decir si esa regla es producto de un acuerdo general (en donde puede haber opiniones contrarias, pero la mayoría concuerda) o bien es el resultado de una decisión impuesta por los gerentes o las personas que, en ese momento, tienen el poder para la toma de decisiones. Cuanto mayor consenso tenga una regla más estable será ya que es el producto de una decisión mayoritaria. Toda esta información sobre los orígenes de la regla facilita la toma de decisiones en el desarrollo de soluciones y ayudan en la evolución de la regla y la propagación de los cambios que provoca.

La *Arbitrariedad* intenta medir la "fuerza" de la regla observando la situación en la cual esa política es aplicada. El uso de constantes y de variables en la descripción de las reglas varía el grado de arbitrariedad, volviéndola más propensa al cambio. Por ejemplo, la regla "El pago debe efectuarse en el Banco X" es más arbitraria que "El pago debe realizarse en cualquiera de los bancos autorizados por la Administradora". ¿ Por qué el Banco X y no otro?. El uso de valores concretos es más arbitrario y por ende más factible de ser modificado. En nuestra estrategia se puede analizar todos los posibles valores que puede tomar una determinada entidad analizando el símbolo del LEL correspondiente. Analizando los impactos y las nociones se puede estimar la probabilidad de cambios de los valores de un símbolo en el Macrosistema.

El *Impacto* mide la repercusión de la aplicación de las reglas en la organización. Este estudio puede ser realizado analizando los símbolos del LEL que representen sujetos (entidades activas) y las frases verbales (representan las principales funciones realizadas en la organización). De acuerdo a este análisis podemos establecer la relevancia del "receptor" de la regla (quien la debe cumplir) y su propósito (límite, derecho o responsabilidad), es decir la acción de la regla que afectará al receptor. Todas las reglas tienen un receptor o destinatario, pero a veces no aparece explícitamente. Analizando y navegando por los símbolos del LEL podemos identificar los receptores que han quedado implícitos en algunas reglas. Utilizaremos un ejemplo extraído del caso de estudio del círculo de Ahorro presentado en el Capítulo 8 para ilustrar este concepto. A partir del texto del contrato presentado en el Anexo I, se identifican las siguientes sentencias:

"El pago de la cuota mensual debe ser realizado en un Banco de la lista de Bancos"
"La licitación debe realizarse el primer día hábil de cada mes."

¿Quién es el responsable de pagar la cuota? ¿Quién debe realizar la licitación? En un caso de estudio complejo no siempre resulta obvio responder a tales preguntas, sin embargo establecer quién es el receptor de la regla es fundamental a la hora de asignar las responsabilidades. A partir de estas sentencias, se pueden identificar las reglas que quedan formuladas de la siguiente forma:

Regla 11: El adherente debe pagar una cuota mensual
Regla 12 El adherente puede elegir entre una lista de Bancos para pagar la cuota mensual.
Regla 31: La Administradora debe realizar el sorteo y licitación el primer día hábil de cada mes.

Complejidad: mide la cantidad de propiedades que aparecen en una regla. Se entiende por propiedades a los símbolos que describen alguna característica de la frase no-verbal o verbal involucrada en la regla. Si la regla tiene muchas propiedades involucradas, tiene mayor probabilidad de cambiar, porque cualquiera de ellas puede variar. Este análisis debe realizarse utilizando la información que se obtuvo cuando se analizó la arbitrariedad y luego combinar todas las propiedades. De esta forma, no sólo determinamos la cantidad de propiedades sino también la probabilidad de cambio de cada una. Debemos tener en cuenta que existen dos tipos de cambios dentro de valores de los símbolos involucrados en las reglas:

- Cambio de los valores de una misma propiedad, por ejemplo, el día de vencimiento del pago puede cambiar de ser el 5^{to} día hábil, al 6^{to}, 7^{mo} o cualquier otro día del mes.

- Sustitución de una entidad por otra del Macrosistema, equivalente en su clasificación (por ejemplo de una sección por otra, un recurso por otro, un determinado rol por otro). Por ejemplo "El pago debe realizarse en un Banco" puede cambiar el lugar del pago, ¿Por qué no puede realizarse en el local de la Administradora, o a domicilio? Estos cambios son más complejos porque provocan un cambio de la funcionalidad del Macrosistema, por lo tanto es de gran importancia realizar un análisis temprano de la posibilidad de cambio para tener en cuenta a la hora de realizar las soluciones en el sistema de software. Por esta razón se debe analizar los símbolos del LEL involucrados en las reglas que identifiquen a este tipo de entidades y buscar si tiene posibles reemplazantes en el Macrosistema (esta búsqueda debe ser validada con personal de la empresa quien define la "factibilidad" del cambio).

Esta información debe ser almacenada junto con las Reglas, por lo que la Vista del Modelo de Reglas tiene la siguiente estructura:

Nro. De Regla:
Cuerpo de Regla: descripción de la Regla según Patrón de Descripción
Origen:
Arbitrariedad:
Impacto:
Complejidad

Fig. 3.3. Vista del Modelo De Reglas

Como todo modelo perteneciente a la *Requirements Baseline*, este modelo está conectado con la Vista de Configuración, que le permite registrar el responsable de cada regla y el manejo de versionado.

3.3.3 Documentación de Reglas

Finalmente, se clasifican y se escriben las reglas de acuerdo con los patrones descritos en la sección 3.2, conectando las reglas con las entradas correspondientes al LEL. A pesar que el LEL ya fue utilizado para determinar la estabilidad de la regla, pueden aparecer sentencias que contienen símbolos que no estaban en el LEL. Este tipo de situaciones llevan a analizar cual de los dos modelos presenta problemas, o la sentencia no representa una regla relevante o surge la necesidad de incorporar el elemento como una entrada en el LEL. A continuación se muestran ejemplos de Reglas del Negocio del Caso de Estudio presentado en el Capítulo 8.

La Administradora acepta solicitantes para sus planes de ahorro
FNV + FV + FNV + FNV

La Administradora debe ofrecer distintos planes de ahorro
FNV + FV + FNV

Donde en ambos casos la Frase Verbal (**FV**)que aparece en cada regla no es término del LEL, pero si lo son las Frases No Verbales (**FNV**)

La licitación realizada por la Administradora produce uno o más adjudicatarios por grupo.
FV + FNV + FNV + FNV

En este caso la Frase Verbal es un término del LEL (ver Anexo), que expresa una de las principales acciones de la Administradora.

La cantidad de miembros de un grupo es el doble de la cantidad de cuotas del plan elegido.
Propiedad + FNV + Relación + Propiedad + FNV

El grupo se identifica por un número de grupo
FNV + Relación + Propiedad + FNV

En ambos casos la propiedad no es una entrada en el LEL pero si el término al que afectan, representado por una Frase No Verbal.

3.3.4 Validación del modelo de Reglas

La validación de los modelos que representan al *UofD* debe ser realizada con tres objetivos [Kotonya'98]: validar que cada modelo es internamente consistente; validar que los diferentes modelos deben ser consistentes entre ellos y finalmente, la tarea más difícil validar que los modelos reflejen las necesidades de los stakeholders. Siguiendo esta propuesta, debemos validar el modelo de reglas desde estos tres aspectos. La *Requirements Baseline* modela aspectos del comportamiento del Macrosistema (modelo de escenarios), su vocabulario (modelo de LEL) y las reglas que los gobiernan a ambos (modelo de reglas). La validación permite descubrir si estos tres aspectos son consistentes entre sí. Asimismo, se valida si el modelo de Reglas no presenta inconsistencias o conflicto internos a él. Finalmente se valida el modelo de reglas con las stakeholders.

3.3.4.1 Validación del modelo de Reglas con los modelos de LEL y escenarios

Las reglas están inmersas en la organización, por lo que deben validarse contra los modelos que describen el comportamiento y estructura de la misma. En [Rosca'97] se presenta un modelo de reglas que se sitúa entre un modelo de la empresa y un modelo operacional del sistema. De esta forma se valida las reglas por un lado, contra los objetivos de la empresa, y por el otro contra las actividades, personas y recursos modelados en el sistema operacional. En [Herbst'96] existe una estrategia similar que valida las reglas con las componentes que modelan a los Procesos y al lenguaje de la organización. En esta propuesta se valida el modelo de reglas con los modelos de la *Requirements Baseline* que reflejan las actividades, entidades y recursos de la organización. Se realiza un análisis informal a través de una serie de preguntas que involucran los tres modelos, los cuales están naturalmente relacionados por los términos del LEL:

Para cada regla del modelo: ¿ Cuales son las entidades de la organización involucradas en esa regla?

Se identifican todos los símbolos de LEL involucrados en las Reglas. Debe analizarse que la regla no contradiga ningún concepto mencionado en las nociones o los impactos. Si este es el caso se analiza con los stakeholders para decidir si es un error de construcción que se debe modificar o si es un problema de consistencia a nivel de la organización que debe ser resuelto a través de la negociación. Los resultados deben ser identificados y recordados según se propone en la Vista de Configuración presentada en el Capítulo 2. Se identifican los escenarios que involucran a los términos del LEL que contiene la regla. Una vez identificados podemos contestar a las siguientes preguntas:

¿ Cuál es el contexto de la organización en el cual esa regla es aplicada?
¿ Con qué comportamiento está asociada?
¿ Cuales son las consecuencias que produce la aplicación de esa regla?

Se analiza el objetivo y el contexto del escenario para responder a la primer pregunta mientras que se analizarán los episodios para responder a la segunda y tercera. Estos no deben contradecir la regla, en cuyo caso debe analizarse con los stakeholders, como se mencionó anteriormente

Debido a problemas de omisión puede ocurrir que haya algún escenario o elemento del LEL que no haya sido referenciado por ninguna regla, escapando del análisis anterior por lo que el mismo se completa desde el punto de vista de estos modelos, para asegurarse que no quedan elementos pertenecientes al LEL y a los escenarios que no hayan sido analizados:

Dado *un símbolo LEL ¿Existe alguna política que gobierna el comportamiento o manipulación de esa entidad? ¿Cómo se comporta o es utilizada en la organización la entidad representada por el símbolo del LEL?* Debe analizarse con los stakeholder para identificar si se trató de omisión o no existen reglas para esa entidad más allá del comportamiento y restricciones propias del Macrosistema. En este análisis se utilizan los escenarios en donde aparece el término del LEL para determinar como se comportan o son manipulados los términos del LEL.

Dado un determinado escenario ¿ Existen políticas que pueden hacer variar el comportamiento de los episodios? Debe analizarse con los stakeholder para identificar que no se hayan omitidos reglas o que no existan reglas asociadas a él. Puede ocurrir que haya reglas de negocio funcionales o no funcionales del Macrosistema que hayan sido omitidas de los escenarios. Debe completarse los mismos para que los reflejen, ya sea agregando o modificando restricciones o episodios. En cuanto a las reglas de calidad, algunas pueden ser volcadas a los escenarios mientras que otros abarcan a todo el *UofD* por lo que no se reflejan en un determinado escenario.

El siguiente es un ejemplo del análisis del caso de estudio del Capítulo 8, cuyo modelo de LEL y escenarios están detallados en el Anexo:

Dada la siguiente Regla:

Regla 12: El adherente puede elegir entre una lista de Bancos para pagar la cuota.

Se buscan los términos del LEL involucrados: adherente, Banco, cuota

Adherente ***Nociones:*** • Solicitante aceptado por la administradora. • Integrante de un grupo. ***Impacto:*** • Pagar cuota mensual. • Transferir plan. • Licitar. • Rechazar bien tipo. • Renunciar al plan de ahorro. • Pagar Adjudicación	Cuota mensual | cuota ***Nociones:*** • Monto mensual que debe pagar el adherente y adjudicatario. • Se calcula sumando la cuota comercial al seguro de vida ***Impacto:*** • Pagar cuota por el adherente ó adjudicatario. • La Administradora calcula la cuota.
Banco ***Nociones:*** • Entidad encargada de cobrar las cuotas en caso de no estar vencidas. ***Impacto:*** • Cobrar las cuotas mensuales. • Enviar cupones de pago a la Administradora	

En el caso del adherente si bien no existen contradicciones, falta un impacto para que el adherente pueda elegir banco. Las otras dos entidades no entran en conflicto con la regla por lo que no es necesario modificarlas.

A partir de la regla 12 se buscan los escenarios que contengan a los tres términos, en este caso es el escenario "Pagar Cuota"

Pagar cuota
***Objetivo**:* El adherente cumple con la obligación contraída al ingresar a un plan de ahorro.
***Contexto**:* Ocurre en el banco. **Restricción**: La cuota no debe estar vencida.
***Actores**:* Adherente, banco, administradora.
***Recursos**:* Cupón de pago
***Episodios**:*
1. El adherente entrega el empleado del banco el cupón de pago con el importe correspondiente.
2. El empleado devuelve el cupón de pago sellado.
3. El banco envía el comprobante de pago a la administradora.

El escenario no contradice a la regla. En realidad el escenario involucrado seria "Solicitar ingreso" pero justamente por un error de omisión faltó modelar esta regla por lo que en dicho escenario no aparece el término Banco y no es seleccionado en esta etapa. Al analizar este escenario en busca de reglas omitidas, vemos que la regla 12 analizada anteriormente no es reflejada en el mismo:

Solicitar ingreso
***Objetivo**:* Una persona desea ingresar a un plan de ahorro.
***Contexto**:* El solicitante pide información acerca de los planes de ahorro existentes.
***Actores**:* Administradora, Solicitante
***Recursos**:* Planilla de adhesión, Cuota de ingreso.
***Episodios**:*
1. El solicitante pide la planilla de adhesión.
2. El solicitante completa dicha planilla y la entrega, abonando la cuota de ingreso.
3. Si la administradora rechaza la solicitud evaluada, ENTONCES ésta devolverá al solicitante el importe correspondiente a la cuota de ingreso.
4. Si la administradora acepta la solicitud, ENTONCES se debe ASEGURAR ADHERENTE.

Asimismo, analizando el escenario en busca de posibles omisiones de otras Reglas, se observa que la regla no funcional No. 32 "La Administradora devolverá el derecho de admisión si la solicitud de adhesión es rechazada, dentro de los 10 días" no es considerada en este escenario. La primera regla(Regla 12) es agregada como episodio mientras que la Regla 32 se modela como una restricción del tercer episodio

Solicitar ingreso
***Objetivo**:* Una persona desea ingresar a un plan de ahorro.
***Contexto**:* El solicitante pide información acerca de los planes de ahorro existentes.
***Actores**:* Administradora, Solicitante
***Recursos**:* Planilla de adhesión Cuota de ingreso.
***Episodios**:*
1. El solicitante pide la planilla de adhesión.
2. El solicitante completa dicha planilla y la entrega, abonando la cuota de ingreso.
3. Si la administradora rechaza la solicitud evaluada, ENTONCES ésta devolverá al solicitante el importe correspondiente a la cuota de ingreso. **Restricción**: "La Administradora devolverá el derecho de admisión si la solicitud de adhesión es rechazada, dentro de los 10 días"
4. Si la administradora acepta la solicitud, ENTONCES se debe ASEGURAR ADHERENTE.

5. Si la administradora acepta la solicitud, ENTONCES el nuevo Adherente elige un Bancos para pagar.

3.3.4.2 Validación del modelo de Reglas con los stakeholders

Este análisis tiene como objetivo detectar posible inconsistencia y conflictos entre las reglas, validando la consistencia interna del modelo y con respecto al stakeholder. Como ya se mencionó, es necesario destacar que esto puede producirse porque hubo errores en la identificación de las mismas, en cuyo caso debe modificarse, o negociarse porque hay un conflicto en la propia organización(esto puede ocurrir cuando hay más de un área o sector involucrado). Se analizan posibles inconsistencias de las reglas entre sí. La identificación de posibles conflictos y su tratamiento es de fundamental importancia para obtener una mayor y mejor comprensión del problema. Como se sugiere en [Kotonya'98] y se detalla en la Regla 8.8. *"Paraphrase System Models"* [Sommerville'97] es indispensable contar con un modelo en lenguaje natural para que los stakeholders puedan participar activamente en la validación. En nuestro caso, el lenguaje natural ya es la forma en que el modelo de Reglas está expresado. Contar con un modelo explícito de reglas escritos en un lenguaje natural, ayuda a poner en evidencia, plantear, y resolver estos conflictos, utilizando diferentes estrategias de negociación. Se agrupan las reglas que comparten los mismos símbolos (frases verbales y no-verbales) y se analiza que las reglas no definan sentencias contradictorias sobre los mismos símbolos. De aparecer estos casos, deben ser resueltos por los gerentes o personal con acceso a la toma de decisiones, para determinar si se trató de un problema de adquisición de información. Una vez que se realiza esta validación se debe analizar un segundo caso que consiste en determinar si la regla causa conflictos entre los miembros de la organización. Este análisis debe ser realizado por los ingenieros de requisitos en interacción con los responsables de las fuentes primarias de información de cada modelo, quiénes brindarán la información faltante y ayudarán a la resolución de conflictos

Dada a la naturaleza orientada a negocios de las reglas y su consecuente notación declarativa y no formal, este análisis no se puede automatizar, sino que se debe realizar en forma manual. Sin embargo se puede utilizar un procedimiento sintáctico para detectar posibles conjuntos de reglas con problemas. Se utiliza para ello los términos del LEL

- Identificar las reglas que *comparten totalmente* los *mismos símbolos del LEL*
- Identificar las reglas que *comparten al menos* un *mismo símbolo del LEL*
- Identificar las reglas que *comparten transitivamente los mismos símbolos del LEL,* donde esta transitividad puede darse de dos maneras:
  - Un término A hace referencia a un término C. Existe otro término B que también hace referencia al término C. Debe analizarse en conjunto las reglas que contienen a los términos A y B. ya que pueden indicar un conflicto no resuelto cuando se analizaron las reglas que compartían al símbolo C.
  - Analizar las reglas que contienen a un determinado símbolo X con las reglas que contiene a cualquier símbolo que el símbolo X referencia en la descripción del LEL. Se hace solo a un nivel.

Con estas heurísticas se obtiene subconjuntos de reglas relacionados. El ingeniero de software puede analizar manualmente si las reglas del subconjunto no se anulan una con otras o son inconsistentes, es decir, que una regla implica la imposibilidad de cumplir con otra. Si encuentra estas reglas debe analizarse con los stakeholder (en este punto el

stakeholder representa a personal de los niveles superior y medio de la organización). Puede darse dos situaciones:

- Las reglas estaban mal elicitadas. Se reformula las reglas junto con los stakeholders
- Las reglas evidencian un conflicto en organización (tal vez no detectado porque hasta el momento no se necesitaba integración entre áreas). En este caso son los stakeholders quienes deciden cual será la regla que prevalecerá sobre la otra o si se reformulan. Esta situación se presenta cuando la organización es más compleja y se utilizan varias Fuentes de información para determinar las reglas de negocio.

El siguiente conjunto de reglas se obtiene aplicando la heurística de compartir al menos un término, en este caso comparten el término sorteo.

Regla 1: La Administradora adjudica uno o más Bienes Tipos por sorteo y licitación
Regla 13: Si no existe deuda pendiente, el adherente participa del sorteo.
Regla 36: Si en un grupo existen fondos para más de un bien tipo, la Administradora adjudica el primero de ellos por sorteo y el resto por licitación.
Regla 37: Si en un grupo sólo existen fondos para un bien tipo, la Administradora realiza la licitación por sorteo.
Regla 38: Si en un mismo grupo existen ofertas de licitación iguales, la Administradora adjudica de acuerdo al orden de sorteo.

Estas reglas, al haber sido extraídas del documento correspondiente al Contrato de la Administradora(ver Anexo), no presentan conflictos entre si. La única objeción es la Regla 37 en donde aparece la palabra licitación mal utilizada ya que la regla establece que si no hay dinero para más de un bien tipo, sólo se hará una entrega mediante un sorteo. La regla es cambiada eliminado a la palabra licitación, ya que, según está definido en el LEL y expresado en la primer regla, el símbolo Licitación indica la otra forma de entregar el bien tipo. La regla definitiva queda expresada de la siguiente forma:

Regla 37: Si en un grupo sólo existen fondos para un bien tipo, la Administradora realiza la adjudicación por sorteo.

3.4 Necesidad de un modelo de Reglas de Negocio

Las Reglas de Negocio constituyen una parte importante del conocimiento de una empresa, el porqué de una organización, el cual es utilizado como entrada para la toma de decisiones, constituyendo parte del corazón de cualquier sistema de negocios [Gottesdenier'97]. Un modelo de reglas permite al cliente describir su conocimiento de una forma sencilla y cercana a la forma en que el cliente percibe a la organización. La identificación de los aspectos del negocio es un aspecto crucial para el desarrollo de un software. Cualquier empresa posee diferentes tipos de información y tratar de modelar de forma separadas sus facetas (por ejemplo estructural, técnica, política, cultural) es desconocer lo que es una organización, ya que las dimensiones estructurales y técnicas son simultáneamente humanas, políticas y culturales [Morgan'98]. Si bien no es posible realizar un modelo que contemple todos los aspectos de una organización, sí es deseable que un modelo contemple, no sólo los procesos, objetivos, recursos que tiene la organización, sino también aspectos intencionales, racionales, de la misma [Yu95]. Un modelo de reglas, siendo estas consideradas como políticas, provee un modelo que explica el porqué [Zachman'96] de los procesos y estructura actuales de la organización. Al

mejorar la comprensión de la organización, un modelo explícito de Reglas mejora la construcción del sistema de software, ya que permite que este se adapte a los constantes cambios de la organización a la cual pertenece. En su guía práctica de Requisitos, Sommerville aconseja considerar los aspectos de organización en la cual estará inserto el software (reglas 3.4: Makes a business Case for the system, 4.6:Use Business concerns to Drive Requirement Elicitation [Sommerville'97]), determinando que el costo de introducción y de aplicación de estas reglas es bajo.

Desde el paradigma de Orientación a Objetos, existe un creciente interés en las Reglas De Negocio, como se puede observar en una de las principales conferencias de objetos a través de la serie "Business Object Workshops" [OOPSLA'97, OOPSLA'98, OOPSLA'99] y más recientemente, en la misma Conferencia, un Workshop especifico de Reglas de Negocio [OOPSLA'00] y un tutorial sobre el tema [Arsanjani'00]

Integrar un modelo de reglas a la *Requirements Baseline* y utilizarlo en la estrategia de Derivación de objetos presentada en esta tesis, permite no sólo comprender mejor la organización agregando aspectos no definidos en los modelos actuales de la *Requirements Baseline*, sino que, junto con los modelos de escenarios y LEL, guía la construcción de los objetos conceptuales permitiendo que el mismo refleje las políticas de la organización y que sea adaptable a los cambios de la misma. Por esta razón, definimos heurísticas tanto para la construcción del modelo de CRCs(Capítulo 4) como para la construcción del modelo lógico(Capítulo 5). Estas heurísticas permiten aplicar las reglas en el modelo de objetos, ya sea modificando CRCs existentes a través de nuevas responsabilidades (heurística **HC9**), creando nuevas clases (sub-heurística **HC9.1**), así como también restringiendo métodos, atributos y asociaciones de las clases como se indica en la heurística **HM5** y sus sub-heurísticas, correspondientes a la Construcción del Modelo Lógico. Asimismo el modelo de Reglas es utilizado en el Capítulo 6 para ayudar a determinar las prioridades del cliente durante la definición de los límites del software.

Capítulo 4
Definición de un modelo de CRCs

"*Encontrar las clases de un sistema es una tarea central en la construcción de un sistema de software orientado a objetos*" [Meyer'97]. Siguiendo una filosofía similar a *RDD*[Wirfs'90; Wirfs'90*]*, *RBM*[Cockburn;99], y *XP* [Beck'00] (metodologías introducidas en el capítulo 1), nuestra propuesta comienza con la definición de las clases y sus responsabilidades, dejando para una etapa posterior el modelo estructural de objetos. La definición se hace principalmente a partir del LEL, utilizando también el modelo de escenarios para completar las responsabilidades y colaboraciones de una clase así como también el modelo de reglas funcionales para reflejar las políticas de la organización ya sea agregando nuevas clases o modificando existentes.

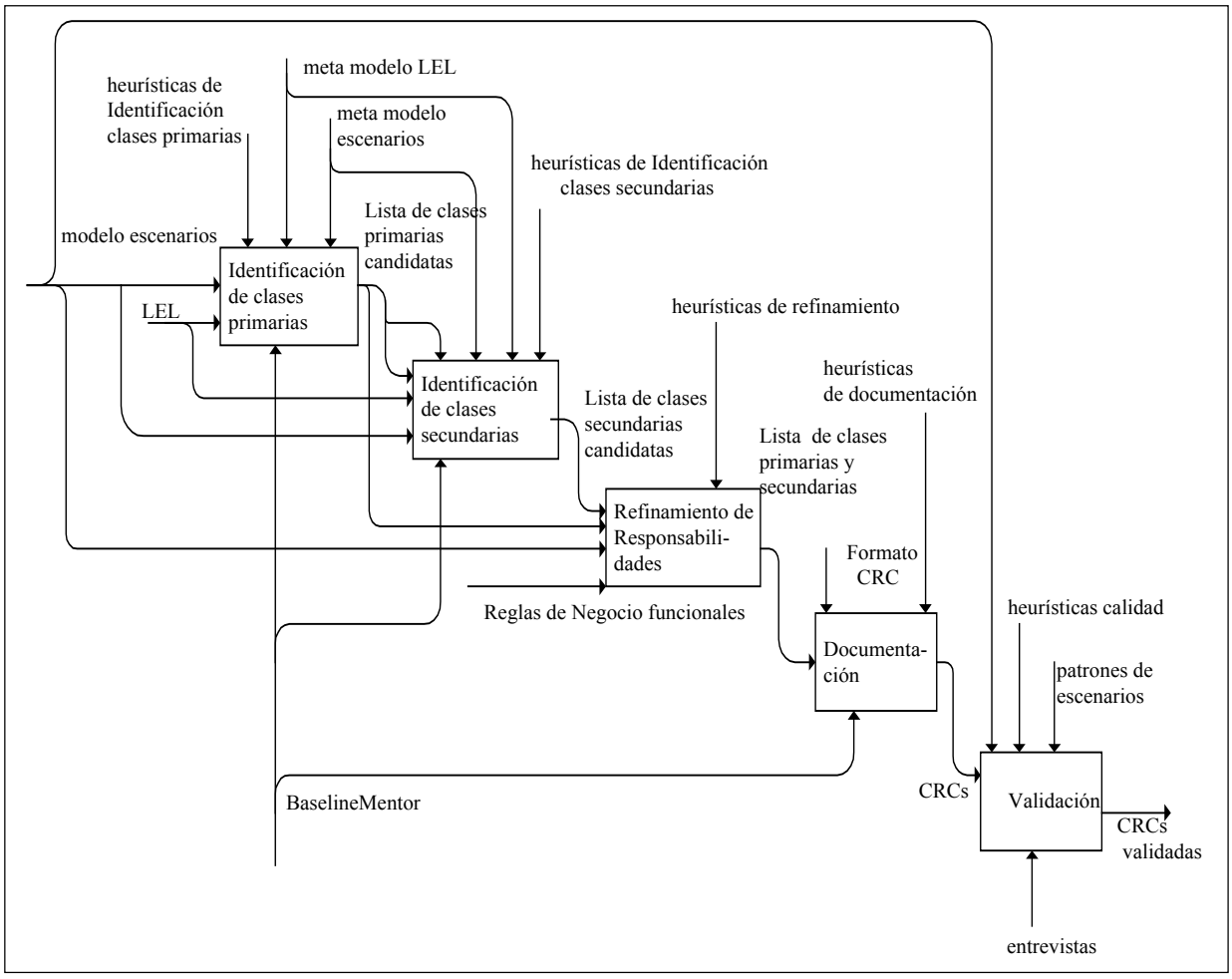
A pesar que la estrategia modela al Macrosistema sin tener los límites de software definidos, debemos tener en cuenta que se está modelando aplicando la filosofía de orientación a objetos. Esto significa que, los objetos (clases) que se identifican no representan solamente los objetos "físicos" del Macrosistema, sino que representan abstracciones relevantes para el Macrosistema. Así, un proceso o un concepto abstracto puede ser modelado en el Macrosistema, si es que representa una abstracción relevante para el Macrosistema. Cuando se definen las clases se lo hace en el contexto de la etapa de modelado conceptual, donde se modela el comportamiento real del Macrosistema. Al pasar a la etapa de diseño, el modelo conceptual se verá modificado por decisiones de diseño, por lo tanto algunos de estos objetos serán modificados, otros desaparecerán y se definirán nuevos objetos.

Al analizar el Macrosistema, se puede distinguir entidades que participan activamente y entidades que son utilizadas para poder llevar a cabo alguna acción. Esta distinción se refleja en la clasificación de actor / recurso considerada en el modelo de escenario y objeto / sujeto considerada en el modelo del *LEL*. Como el modelo de clases que estamos definiendo refleja al Macrosistema, tendrá estas mismas categorías. De ahí se puede hablar de clases Primarias y Secundarias. Si bien en términos de orientación a objetos no existen diferencias entre ellas, (ambas encapsulan datos y comportamiento) las heurísticas para definirlas son diferentes ya que dentro del Macrosistema tienen semántica diferente, siendo las clases primarias aquellas que representan a entidades activas del Macrosistema y las clases secundarias a entidades pasivas.

Esta primer etapa de la estrategia propone identificar las clases primarias y luego las clases secundarias. La información se modela en tarjetas estilo *CRCs* [Beck'89]. La identificación se hace principalmente a partir de los términos del *LEL*, pero luego se refinan las colaboraciones siguiendo los escenarios, en una técnica similar a la propuesta en *RBM*. La actividad principal del diseño de un modelo de objetos es la asignación de las responsabilidades [Cockburn'99] a los objetos. En este sentido, el modelo de LEL ayuda ya

que previamente se ha definido un conjunto de impactos del término en el Macrosistema y estos impactos guiarán la derivación y asignación de las responsabilidades. En general, la asignación de una responsabilidad es en la clase que representa al término que contiene al impacto correspondiente. Sin embargo, debe tenerse en cuenta, qué significa el impacto dentro del termino según sea su clasificación(objetos, sujeto, estado, frase verbal), ya que debido al significado del impacto para cada término (ver heurísticas de formación de términos del LEL, capítulo 2, Figura 2.3) la asignación puede variar. Esto se explicará más detalladamente en las heurísticas correspondientes. El siguiente SADT muestra las entradas y salidas de las sub_etapas.

Fig. 4.1: Proceso de modelización de *CRCs*



Para la descripción de las heurísticas se utilizará el siguiente patrón:

Identificador Se utilizará una Sigla para identificar a la heurística. Todas las heurísticas relacionadas a la definición de *CRCs* empiezan con HC, las del modelo lógico como HM y las relacionadas con la definición de límites con HL. Esta sigla es utilizada para las relaciones de trace que surgen en el modelo a partir de la aplicación de las heurísticas(capítulo 7).

Descripción: Presenta la descripción y justificación de la heurística. Pueden aparecer sub-heurísticas, que se identifican con un orden relativo a la heurística a la que pertenece.

Ejemplo: En general, se ejemplificarán las heurísticas con el caso de estudio ”Círculo de ahorro” descrito en el capítulo 8, de no existir el ejemplo para alguna

heurística en particular se utilizará algún ejemplo de los casos de estudio mencionados también en ese capítulo. Los ejemplos pertenecientes a los modelos de LEL, escenarios y reglas se presentan con el estilo de letra tahoma y los ejemplos correspondientes a los modelos de objetos se identifican con el estilo de letra Bookman old style

Para mejorar la calidad tanto del proceso de construcción como del producto final del modelo de objetos de software, se pueden utilizar distintos principios de calidad de modelado, como por ejemplo [Meyer'97; Cockburn'99] en donde se mencionan heurísticas para modelar clases con calidad, principios para establecer asociaciones entre clases, y cuestiones de reusabilidad, entre otros. En esta propuesta se mencionan aquellas cuestiones en donde el uso de los modelos de escenarios, LEL y reglas de Negocio ayudan en este proceso. Por consiguiente, esta estrategia debe ser considerada junto con principios generales de modelización.

Como se observa en la Figura 4.1, las heurísticas presentadas en este capítulo permiten definir el modelo de CRCs sobre la base de los modelos de la *Requirements Baseline* extendida. La aplicación de estas heurísticas genera relaciones de traceability entre los modelos generados y los generadores, permitiendo obtener información que justifica la existencia de las CRCs y sus responsabilidades. Este aspecto se detalla con detalle en el Capítulo 7.

4.1 Heurísticas de Identificación de clases primarias y sus responsabilidades

Se entiende por clases primarias aquellas que representan una abstracción de comportamiento significativo para el *UofD*, es decir aquellas que modelan parte de la funcionalidad del mismo. Como todavía no existe un modelo de software en donde existen factores de diseño que puede modificar la solución, el comportamiento más relevante es el realizado por las personas, organizaciones, o sectores del Macrosistema que se está modelando. Estas entidades son los Actores de los Escenarios y los Sujetos del LEL.

- **HCI:** MODELAR LOS SUJETOS DEL LEL COMO CLASES CANDIDATAS
  Descripción: En los escenarios estos sujetos son actores. Los sujetos corresponden a roles o partes de una organización. Al estar definiendo un modelo del Macrosistema, estas entidades son las que realizan las principales acciones del mismo por lo que su modelización como clases es automática.
  Ejemplo: El término del LEL Administradora clasificado como Sujeto corresponde a la organización que se está modelando, por consiguiente existirá una clase "Administradora". El término del LEL Adherente que representa a una persona con un determinado rol dentro de la organización (pertenecer a un grupo de Ahorro), es también modelado como una clase.

- **HC2**: DETERMINAR LAS RESPONSABILIDADES A PARTIR DE LOS IMPACTOS
  Descripción: Se analiza cada uno de los impactos del LEL correspondientes al término para el cual se ha definido la clase. Al ser clases que han surgido a partir de los Sujetos del LEL, por definición, todos los impactos son acciones que el sujeto realiza(capítulo 2, Figura 2.3), entonces se define al impacto como una responsabilidad del objeto. Existe un caso a tener en cuenta, y es cuando el impacto representa a una frase verbal

que ha sido modelada como un término del LEL. Este será analizado en la Identificación de las clases secundarias, pudiendo o no ser modelado como una clase, de todos modos, en principio la responsabilidad queda asignada en esta nueva clase y colaborará con la clase que representa a la acción. Puede haber modificaciones en las siguientes etapas, dependiendo de como se modelará el término del LEL que aparece en esa responsabilidad.

Ejemplo: Para la clase Administradora el impacto Expulsar Adherente se modela como responsabilidad. En el impacto Evaluar licitación existe un término Licitación en el LEL, por el momento queda como responsabilidad de la clase Administradora, sin embargo es candidata a ser una clase secundaria ya que aparece un término del LEL correspondiente a una Frase Verbal que define la Licitación.

4.2 Heurísticas de Identificación de clases secundarias y sus responsabilidades

Se entiende por clases secundarias aquellas que representan recursos para que las clases primarias puedan realizar sus responsabilidades. Son entidades pasivas dentro del Macrosistema, generalmente son repositorios de datos, a excepción de las clases provenientes de Frases Verbales.

- **HC3-** ANALIZAR LOS TÉRMINOS DEL LEL QUE CORRESPONDEN A OBJETOS, ESTADOS Y FRASES VERBALES

  Descripción: Se construye una lista con los términos del LEL que representen objetos, frases verbales y estados. La mayoría de estos términos figuran en cada una de las responsabilidades de las clases definidas previamente. Sin embargo, pueden aparecer mas términos que no están directamente relacionados con los términos que se han convertido en clase candidata. Se analiza cada uno de estos términos para determinar si serán modelados o no como objetos.

  HC3.1- OBJETOS DEL LEL COMO ATRIBUTOS O CLASES

  Descripción: para cada objeto se analizan sus impactos:

  - Si el término no tiene un comportamiento relevante se modela como un atributo. Para esto se busca en las nociones e impactos de los términos correspondientes a Sujetos que han sido modelados como clases, para determinar la clase a la que corresponde. Si existen dudas de la clase que lo debe contener, se revisan los escenarios en donde aparece este término. En etapas posteriores puede convertirse en objeto, pero en esta etapa no justifica que lo sea. Para decidir si será modelado como atributo se debe analizar si los impactos de los términos no están describiendo acciones similares a alguna clase primitiva (por ejemplo, Integer, String) es decir, que si bien es una abstracción propia del sistema, en términos de objetos, no justifica ser una nueva clase [Meyer'97]. Generalmente ocurre que es un término de valor simple o un conjunto de términos de valores del mismo tipo.
  - Si el término tiene comportamiento significativo para el Macrosistema se convierte en clase, ya que define una porción de comportamiento importante que puede ser modelado como una abstracción independiente necesaria para que las clases primarias puedan cumplir sus responsabilidades.

  Ejemplo: El término del LEL Cuota Mensual, no tiene un comportamiento propio que justifique que sea modelado como un objeto del modelo conceptual. A pesar que existe una regla no funcional que define su formación (Regla 70, capítulo 8) ésta puede expresarse en el modelo lógico como se explica en la heurística **HM5** y no justifica su modelización como clase. Por otro lado, el término Comunicación

Fehaciente es un término que representa un mecanismo de comunicación muy utilizado dentro de la Administradora, por esta razón es modelado como clase para independizar las diferentes formas de comunicación de las entidades que lo utilizan.

HC3.2: FRASES VERBALES DEL LEL COMO CLASES O RESPONSABILIDADES

Descripción: Si el término corresponde a una acción, se debe analizar la posibilidad o no de representarla como una clase. Esto ocurre generalmente cuando tiene características propias que no pueden ser asignadas a otras clases. Ejemplos típicos son las transacciones bancarias como por ejemplo depósito o extracción, que tienen atributos propios que justifican que sea modelada como clase. Si este es el caso se modela como objeto, sino, se convierte en una o más responsabilidades de los objetos involucrados. Cuando se modela sin tener en cuenta al software, las acciones que se encuentran son realizadas o disparadas por algún actor (por ejemplo el término del LEL correspondiente a Entregar Bien tipo representa una acción realizada por la Administradora). Luego, al pasar al diseño preliminar y marcar los limites del software, parte de la funcionalidad de los objetos quedan adentro y parte afuera de estos límites. Por esta razón, al desaparecer los objetos, responsabilidades que antes correspondían a algunos objetos del modelo conceptual son reasignadas a otro objeto, o bien son modeladas como un objeto de software. Existen estrategias de diseño que, ante determinadas características de esa acción, sugieren que sea modelada como un objeto, como por ejemplo el "*pattern strategy*" [Gamma'94]. El modelar un comportamiento como una clase, permite refinar al comportamiento a través de una jerarquía de herencia sin que se vean involucradas las clases que lo utilizan. Estos objetos son llamados "coordinadores o controladores " [Wirfs'95], y desde una perspectiva de Negocios y Orientación a Objetos, son conocidos como "objetos del proceso de negocio" [Jacobson'95]. En general, debieran surgir a partir de una decisión de diseño y no del modelado del Macrosistema. El abuso de este tipo de modelado es una de las principales causas de una mala definición de clases [Meyer'97]; si una clase es descrita solo por la capacidad de hacer una acción, ¿no es más adecuado que se convierta en una responsabilidad de otra clase? Solo podría justificarse que esa acción sea modelada como clase si representa un comportamiento complejo, o si guarda propiedades que le corresponde a esa acción y no a ninguna otra clase, como los ejemplos mencionados de transacciones bancarias o las clases que representan comandos de acción undo-redo [Meyer'97].
Ejemplo: el término Licitar tiene un conjunto de impactos que justifica sea representada como clase ya que describen a una de las principales acciones de la Administradora. Además, se necesita guardar información propia de la Licitación, que no puede almacenarse en otra clase. Por esta razón se modela como clase Licitación, la cual colaborará con la clase Administradora. Sin embargo, el término Expulsar Adherente no tiene impactos que justifiquen sea modelado como clase, por lo que queda como responsabilidad de la clase Administradora.

HC3.3: ESTADOS COMO CLASES O ATRIBUTOS

Descripción: Si el término corresponde a un estado, por definición del mismo, ese estado afecta a todo el Macrosistema o a algunas de sus componentes, reflejados como términos del LEL. El hecho de haber sido modelado como un término diferente, implica que tiene nociones o impactos que lo caracterizan y distinguen. Si el término del LEL al cual afectan fue modelado como clase, se analiza primero si el estado no puede convertirse en un atributo(con una responsabilidad que lo

modifique), pero si tiene impactos que pueden ser modelados como responsabilidades, entonces es candidato a ser modelado como una clase diferente.
Ejemplo: En el caso de estudio del Meeting Scheduler existe el término Posible Reunión ya que contiene características propias que la diferencian de una reunión. Se modela como clase ya que la reunión también lo es y posibleReunión tiene responsabilidades bien diferenciadas de la reunión. Sin embargo, en el Sistema de Pasaportes los términos correspondientes a Pasaporte inválido y Pasaporte renovado se modelan como atributos de Pasaporte ya que indican un estado del mismo sin anexar comportamiento extra. (en el caso de pasaporte inválido, invalida al Pasaporte pero no anexa comportamiento)

- **HC4:** DEFINIR LAS RESPONSABILIDADES DE LAS CLASES SECUNDARIAS UTILIZANDO EL LEL
  Descripción: Al igual que con las clases primarias, se analiza los impactos de los términos del LEL correspondientes a las clases definidas. Es necesario tener en cuenta la categoría del término para el cual se está determinando las responsabilidades ya que según sea la categoría cambia el sentido de los impactos pudiendo, en algunos casos, agregar responsabilidades a otras clases.

  HC4.1: DETERMINAR LAS RESPONSABILIDADES DE LOS TÉRMINOS CORRESPONDIENTES A OBJETOS
  Descripción: los impactos de un Objeto describen qué acciones se le pueden aplicar al mismo (capítulo 2, Figura 2.3). En términos del paradigma de orientación a objetos, esto no necesariamente implica que siempre el objeto va a tener una responsabilidad para responder a estas acciones. Depende de la semántica del impacto. En general, siempre habrá una responsabilidad asociado al impacto, pero a veces puede ser que el objeto sea un recurso para otro (sujeto u objeto) y no tenga una responsabilidad asociada. En este caso se debe buscar otro término dentro del impacto que representa a la clase en cuestión o analizar el conjunto de escenarios en donde aparece este término para buscar las acciones asociadas a este impacto y determinar las clases involucradas.
  Ejemplo: Si consideramos el impacto de una Solicitud "es llenada", nos damos cuenta que se debe definir una responsabilidad para la clase Solicitud que permita ser llenada, en este caso una responsabilidad básica de creación y modificación de los valores de la solicitud. Por otro lado, si analizamos el impacto "es pagada mensualmente" del término Cuota, nos damos cuenta que no debe modelarse como una responsabilidad "serPagada", ya que la cuota no lo tiene que registrar, sino que esto queda registrado en el sujeto que realizó la acción o en algún objeto encargado de registrar el pago de Cuotas, en este caso se asocia como una cuotaPaga a la clase Adherente.

  HC4.2: DETERMINAR LAS RESPONSABILIDADES DE LOS TÉRMINOS CORRESPONDIENTES A FRASES VERBALES
  Descripción: en el caso de las frases verbales, los impactos describen restricciones de la acción y nuevas acciones. Si es una restricción de las acciones pueden derivar responsabilidades así como también pueden afectar a responsabilidades existentes (eso se verá en el modelo lógico de objetos). Si es una acción, se debe analizar si no existe un término del LEL en el impacto que sea el ejecutor de la acción. En este caso existe una colaboración entre ambas clases. Si no es el caso

se modelará como responsabilidad de la clase que está representando a la clase verbal.
Ejemplo: analizando el término Sorteo para definir las responsabilidades de la clase correspondiente, se ve que el impacto "se define el adherente ganador..." queda como una responsabilidad de la clase. Por otro lado, el impacto "el adherente que ganó el sorteo puede rechazar el bien tipo ..." establece una acción de rechazo para el Adherente y una aceptación de rechazo para el Sorteo que a su vez lo manejará con la Administradora.

HC4.3: DETERMINAR LAS RESPONSABILIDADES DE LOS TÉRMINOS CORRESPONDIENTES A ESTADOS
Descripción: si el término es un estado se aplican algunas de las dos heurísticas presentadas anteriormente **(HC4.1** y **HC4.2)** dependiendo de la clasificación del término al que está afectando ese estado.
Ejemplo: En el caso de la clase posibleReunión del Meeting Scheduler se aplica la heurística **H4.1** estableciendo responsabilidades que permiten determinar los datos básicos de la Reunión que son utilizados por las clases Secretaria y Convocante.

4.3. Heurísticas generales para la definición de las clases

Tanto para la identificación de las clases primarias como para las secundarias se deben considerar las siguientes pautas:

- **HCG1:** ELEGIR EL NOMBRE MÁS CORTO Y SIGNICATIVO PARA DEFINIR A LA CLASE
  Descripción: El nombre debe ser el sinónimo más corto que se haya definido para cada término del LEL, excepto que sea similar a otro término que también se ha modelado como clase. En [Cockburn'99] se presentan heurísticas simples para elegir un nombre correcto teniendo como premisa que se pueden hacer suposiciones sobre una clase basándose en el nombre, el cual debe reflejar claramente la abstracción que la clase representa. Un nombre no debiera estar escrito como una frase verbal, ya que esto es indica una posible mala definición de clases [Meyer'97].
  Ejemplo: el término Administradora tiene el sinónimo Administradora del Círculo de Ahorro para la Compra de un automotor. Este nombre es muy largo y como solo se está modelando ese Macrosistema aparecería como redundante. Se elige como nombre de la clase a la palabra Administradora.

- **HCG2:** DEFINIR LOS NOMBRES DE LAS RESPONSABILIDADES CON VERBOS INFINITIVOS
  Descripción: Las responsabilidades deben estar escritas en verbos infinitivos, dejando bien en claro la semántica de la acción. La correcta elección de nombres de las responsabilidades permite a los analistas comprender rápidamente el comportamiento del sistema en estudio [Cockburn'99].
  Ejemplo: El término Expulsar Adherente tiene el sinónimo adherente es expulsado. Este término ha sido modelado como responsabilidad de la clase Administradora por lo que se utiliza el sinónimo infinitivo. El término correspondiente a la Administradora tiene un impacto que es Aceptación de solicitud de adhesión. La responsabilidad será aceptarSolicitud.

- **HCG3:** ANALIZAR SI EL IMPACTO NO REFLEJA MÁS DE UNA ACCIÓN.

Descripción: Si un impacto describe más de una acción relacionada debe analizarse la posibilidad de que se conviertan en dos responsabilidades. Para esto se puede analizar los escenarios en donde aparecen los términos para poder decidir si se justifica modelar como dos responsabilidades diferentes o son variantes de un mismo comportamiento. Muchas veces un impacto que describe dos acciones esta asociado con algún escenario que contiene episodios condicionales. Analizando estos escenarios se puede determinar si se necesita modelar uno a mas responsabilidades. Esta situación no es común por la propia definición de los impactos de los términos del LEL.
Ejemplo: en el término Adherente aparece Pagar cuota en Banco o pagar cuota vencida. Serán modeladas como dos responsabilidades diferentes.

- **HCG4:** ASIGNACIÓN DE RESPONSABILIDADES REDUNDANTES
  Descripción: Con respecto a la asignación de las responsabilidades a las nuevas clases, se debe tener en cuenta la redundancia del LEL. Esto significa que una misma acción puede estar descripta en distintos términos pero pertenecer a uno (esto sucede cuando pertenece por ejemplo a un Sujeto y una Frase Verbal). Se debe establecer claramente la colaboración, quien ejecuta la acción y quien es el cliente o si ambos son responsables.
  Ejemplo: esta situación fue reiterativa en el caso de la Auto-aplicación de la metodología. Por ejemplo se puede tomar el impacto: Realizar la detección de excepciones. Este impacto se encuentra en varios símbolos del LEL, a saber: Construcción de escenarios, Ingeniero de requisitos y Construcción del LEL. Claramente se puede determinar que la responsabilidad corresponde a la clase Ingeniero de Requisitos.

4.4 Refinamiento de colaboraciones y responsabilidades

Al definir las responsabilidades de las clases primarias y secundarias a partir de los impactos del LEL, aparecen naturalmente las clases que colaboran ya que son los términos del LEL que están en los impactos analizados que se han convertido en clases. En esta etapa se refinan las colaboraciones y responsabilidades, utilizando los escenarios

- **HC5:** REFINAR RESPONSABILIDADES Y COLABORACIONES ANALIZANDO EL MODELO DE ESCENARIOS
  Descripción: Construir una lista con los escenarios en donde aparece cada una de las clases definidas. Al identificar las responsabilidades a partir de los impactos del LEL, estas pueden ser muy generales. Algunas de las responsabilidades deben ser refinadas a partir de los escenarios defiendo las acciones o sub-responsabilidades asociadas (en la etapa de la definición del modelo de objetos, se definirá cuales se transforman en métodos). Se analizan los episodios y se refinan las responsabilidades con las acciones que la clase lleva a cabo para cumplir con esa responsabilidad. Al refinar la responsabilidad se refinan las colaboraciones porque se definen las colaboraciones a nivel de acciones de la clase. Para cada acción se especifica la colaboración y se verifica si existe en la otra clase alguna responsabilidad que puede llevar a cabo la colaboración.
  Ejemplo: analizando el escenario "Licitar" en el cual aparece la responsabilidad licitar de la clase Adherente, se determina que esta responsabilidad puede ser refinada como crear sobreLicitación y enviarSobre. Para la primera colabora con la clase sobreLicitación, mientras que para la segunda con la Administradora. Se verifica que

para ambas clases existan responsabilidades que las cumplan. En este caso, se agregó crearSobre a la clase sobreLicitación.

HC5.1: ANALIZAR LAS ESTRUCTURAS DE LOS ESCENARIOS QUE INDIQUEN CURSOS ALTERNATIVOS DE ACCIÓN
Descripción: En esta etapa se debe analizar si asociadas a cada responsabilidad que aparece en los escenarios hay excepciones, restricciones o cláusulas condicionales para evaluar las distintas alternativas de comportamiento. Se refina la colaboración o se generan nuevas responsabilidades para poder llevar a cabo el comportamiento detallado en el curso alternativo.
Ejemplo: analizando el escenario Evaluar Licitación se observa que existe una sentencia condicional que indica: Si hay ofertas similares se toma el primero en el orden establecido en el Sorteo... En este caso, si bien no se crea una nueva responsabilidad, se refina la responsabilidad EvaluarLicitación de la clase Licitación para que obtenga de la clase Sorteo el orden del sorteo y determine el ganador (posteriormente, durante la definición del modelo de objetos se podrá decidir si este refinamiento se convierte o no en un método privado de la clase que realice esta acción)

HC5.2: AGREGAR RESPONSABILIDADES BÁSICAS
Descripción: En los escenarios se encuentran nuevas responsabilidades que tratan aspectos de acceso y modificación del estado interno del objeto. Estos no siempre figuran en el LEL porque no encierran una semántica significativa para el sistema. Sin embargo, desde el punto de vista del modelo de solución es una responsabilidad de la clase. Si no se quiere agregar complejidad en esta etapa se puede dejar para la etapa del Diagrama de Objetos (heurística **HM2.4**)
Ejemplo: en los escenarios Armar Grupo y Asegurar Adherente, se observa que la Administradora toma los datos de la Planillas de adhesión. Se crea una responsabilidad básica darDatos en la clase la Planillas de adhesión con los datos necesarios. En la siguiente etapa, se determinará si esa responsabilidad se transforma en uno o más métodos (heurística **HM2**)

- **HC6:** DEFINICIÓN DE CLASES QUE NO PERTENECEN AL LEL
Descripción: Pueden modelarse como objetos entidades que aparecen en los escenarios interactuando con objetos pero que no figuran en el LEL por no ser términos propios del Macrosistema. Las responsabilidades se definen analizando los escenarios en donde aparecen.
Ejemplo: en el caso de estudio del "Meeting Scheduler" surge el directorio telefónico, que si bien no aparece como término en el LEL por ser un término de uso común, es utilizado como recurso en algunos escenarios.

- **HC7:** ANALIZAR VERBOS CON SEMÁNTICA DE REGISTRAR
Descripción: Se debe analizar los tipos de verbos cuya semántica es registrar información. Si los impactos de un término describen alguna acción relacionada a registrar, se debe analizar si la información que se está registrando no puede modelarse como objeto. En general, esta heurística no se aplica ya que al tener el LEL aquellos términos que son relevantes para el Macrosistema, existirá un término por cada elemento significativo que haya en el *UofD*. Sin embargo es útil considerarla para ser usada como validación del LEL.

Ejemplo: en la introducción se mencionó un ejemplo extraído de Meyer'97
Se debe guardar un registro cada vez que un elevador se mueve de un piso a otro
Se debe guardar un registro cada vez que se produce un movimiento de un elevador de un piso a otro.
La palabra mover aparece como una frase verbal en una sentencia y como sustantivo en otra. Si bien en la primera aparecería registrar como una posible clase, claramente es el registro de un movimiento el que interesa guardar, y esta abstracción es más correcta que la anterior.

- **HC8:** ANALIZAR SI LA CLASE REPRESENTA A UN INDIVIDUO O UNA COLECCIÓN DE INDIVIDUOS
  Definición: Tanto para las clases primarias que modelan Sujetos como para secundarias que modelan Objetos del LEL, debe analizarse si en los escenarios aparece una instancia o un conjunto de instancias. Si aparece un conjunto se modelará como una colección, sin crear otra clase, solo se indica con el plural. Se considera que es una colección con operaciones básicas, por lo que en las siguientes etapas, puede modelarse con clases predefinidas que representan a colecciones. Si tuviera otro comportamiento / significado en el Macrosistema, tendría que haber aparecido como un término independiente del LEL.
  Ejemplo: si se analiza el escenario Evaluar licitación se ve que como recurso se utiliza un conjunto de Sobres de Licitación, que son los que serán evaluados. Al no tener una semántica diferente, más allá que la propia de una colección, se utiliza el plural para diferenciarla de la clase que representa a un miembro de la colección pero no se crea una nueva clase. En el caso de los adherentes, existe el término del LEL Grupo que si bien se define como un conjunto de adherentes, tiene su propia semántica dentro del Macrosistema; por esta razón es modelado como clase.

- **HC9:** UTILIZAR EL MODELO DE REGLAS FUNCIONALES PARA COMPLETAR LAS CLASES**.**
  Descripción: Las reglas funcionales definen el comportamiento esperado por la organización; por lo tanto, en general, las reglas se modelarán como responsabilidades o refinarán responsabilidades existentes. Se analiza si cada regla ha sido modelada como una responsabilidad de la clase correspondiente a algunos de los términos del LEL que aparecen en la regla y han sido modelados como clase.
  Ejemplo: La siguiente regla:

  > Regla 5: La Administradora debe convocar a Asamblea de adherentes ante la discontinuación del bien

  refleja una obligación de la Administradora, sin embargo no está modelada como responsabilidad de la clase que la representa. Se definen una nueva responsabilidad para representarla: convocarAsamblea.
  Las reglas:

  Regla 11: El Adherente debe pagar su cuota mensual
  Regla 12: El Adherente puede elegir entre una lista de Bancos

  representan comportamiento asociado a Adherente y sí han sido modeladas como responsabilidades de la clase correspondiente.

  HC9.1: MODELAR LA/S REGLA/S COMO CLASES
  Descripción: Existen situaciones en que se justifica modelar un conjunto de reglas como clases, para aislar determinadas políticas en un solo objeto o

porque las reglas afectan a diferentes objetos y no es posible asociarlos a ninguno en particular [Høydalsvik'93]. Para esto se puede agrupar las reglas similares (funcionales y no funcionales del Macrosistema) por términos en común (por ejemplo si se quiere agrupar por actividad, se agrupan las reglas que compartan la misma Frase Verbal). Si a partir de la complejidad, estabilidad e importancia de este grupo de reglas dentro del Macrosistema (capítulo 3 sección 3.2), así como también su imposibilidad de asignarlas a una determinada clase ya existente, se determina que el mismo puede ser aislado, entonces se crea una clase para modelar a estas reglas y se definen las colaboraciones con las clases involucradas. Si se determina la creación de una nueva clase, se debe analizar si el comportamiento originado por las reglas no está asignado a otra clase, en cuyo caso se debe reasignar responsabilidades y colaboraciones. Definir las reglas como clases mejora la comprensión de las políticas que gobiernan al sistema ya que las reglas se describen explícitamente como clases separadas y no se embeben en ninguna otra clase del sistema. Por otro lado mejora el mantenimiento del sistema, ya que si una regla cambia se pueden detectar las reglas asociadas encapsuladas en el mismo objeto y analizar posibles de las mismas. Asimismo, se pueden especializar independientemente de los objetos a los que afecta y se puede tener un mecanismo de activación/ desactivación en el mismo objeto que modela la/s regla/s, para reflejar el dinamismo de éstas sin afectar a los objetos conceptuales involucrados [Diaz'97]. La desventaja de modelar reglas como objetos es que aumenta la complejidad del modelo ya que se agregan más objetos y relaciones. Por esta razón, estas heurísticas son fuertemente dependientes de lo que están modelando las reglas con respeto al Macrosistema.

Ejemplo: En el caso de estudio del Círculo de Ahorro, se creará un objeto Administración de pagos (ver capítulo 8) para manejar los pagos. Esto se justifica porque existen varias reglas asociadas(reglas funcionales 4,7,10 y 12 reglas no funcionales 40-44,51, 66 y 72) y es una de las actividades fundamentales que afectan muchas componentes del Macrosistema. Pero por ejemplo no tiene sentido creer reglas para modelar cómo se realiza la inscripción(a pesar que existen varias reglas asociadas a esta actividad) ya que está actividad está concentrada en la Administradora, no tiene gran impacto en el Macrosistema y es estable.

4.5 Documentación de las clases

- **HC10:** DOCUMENTAR LAS CLASES CON TARJETAS ESTILO *CRCs*

  Descripción: Las *CRCs* inicialmente propuestas en [Beck'89] son tarjetas que describen las clases, sus responsabilidades y colaboraciones. Son usadas en varias metodologías como *RDD*, *RBM* y *XP* (mencionadas en el capítulo 1). En nuestra propuesta han sido adaptadas para reflejar las heurísticas de identificación de las clases. La figura 4.2 muestra los componentes de una *CRC*.

Nombre de la clase
Justificación de la clase: conjunto de heurísticas que cumplió y que le permiten ser modelada como una clase
Conjunto de { *Responsabilidad* (se puede agregar la heurística que la generó, dependiendo del nivel de trace que se quiera tener*)*
Colaboraciones
Contexto: conjunto de escenarios en donde se realiza esa responsabilidad*}*

Fig. 4.2: **tarjetas de documentación de clases**

Ejemplo: La figura 4.3 muestra una descripción parcial de la tarjeta correspondiente a la clase Adherente.

nombre: Adherente justificación: cumple heurística HC1		
Responsabilidades	Colaboraciones	Contexto (HC5)
PagarCuotaMensual (HC2) entregarCupon(HC4.1)	cupón de Pago Banco, Administradora,	Pagar cuota
PagarCuotaMensual vencida(HCG3)	cupón de Pago Administradora,	Pagar Cuota Vencida
TransferirPlan (HC2) EnviarComFehaciente (HC5)	Administradora, Comunicación fehaciente	Transferir Plan
Licitar (HC2) crearSobreLicitacion (HC5) enviarSobre (HC5)	sobreLicitación Administradora	Licitar
RechazarBienTipo (HC2, HC3.2)	Bien tipo, Administradora, Comunicación fehaciente	Rechazar Bien tipo
Renunciar (HC2)	Plan de ahorro, Administradora Comunicación fehaciente,	Dejar Plan
PagarAdjudicación (HC2)	Administradora Adjudicación	Adjudicación
ElegirBanco (HC5)	Administradora Banco	Solicitar Ingreso
RecibirCuponPago(HC5)	Administradora cupones	Solicitar Ingreso

Fig. 4.3: ***CRC*** **correspondiente al término Adherente**

4.6 Evaluación de las *CRCs*

El objetivo de esta etapa es determinar junto con el cliente si el conjunto de CRCs obtenidas modelan, desde una perspectiva de orientación a objetos, el comportamiento del Macrosistema. Además se buscan posibles errores u omisiones en las CRCs con respecto a

los modelos de reglas, escenarios y LEL. Esto último puede causar la modificación de los modelos de la *Requirements Baseline*.

- **HC11:** "EJECUTAR" LOS ESCENARIOS SOBRE LA BASE DE LAS CRCS
  Descripción: Si la heurística **HC5** es aplicada, HC11 sería redundante, salvo que las validaciones las realice un equipo diferente al que formuló las CRCs. Las CRCs pueden ser evaluadas siguiendo los pasos propuestos en [Cockburn'99], es decir tomar cada escenario y "avanzar " por él determinando si las clases y sus responsabilidades llevan a cabo las acciones de los episodios. Dentro de esta heurística, se puede analizar los patrones de escenarios [Ridao'00]. Según sea el tipo de patrón al que corresponde cada escenario en donde aparecen las CRCs se puede validar si existen colaboraciones con las otras CRCs para llevar a cabo los tipos de episodios pedidos por cada tipo de patrón. Por el momento esta actividad sólo sirve para ver si se necesita una colaboración y cuantas CRCs como mínimo debe intervenir(a partir de la cantidad mínima de participantes que exige cada tipo de patrón). Profundizar en este tema está dentro de las actividades futuras de este trabajo (ver Trabajos Futuros en Capítulo 9). Por otro lado, es importante tener en cuenta que si una determinada responsabilidad involucra clases servidoras, éstas deben tener responsabilidades que puedan llevar a cabo la tarea mencionada. Con respecto a las clases clientes, tiene que aparecer alguna responsabilidad en la CRC correspondiente que tenga a la clase como servidora. Este análisis completa la estrategia anterior, ya que en un escenario puede figurar implícitamente las responsabilidades de las clases. Por ejemplo, el cliente reserva un libro. Esta oración esta bien escrita y es válida dentro de un episodio. Desde el punto de vista de objetos, tanto el cliente como el libro deben tener una responsabilidad que describa el comportamiento que le corresponde de la reserva.

- **HC12:** APLICAR PRINCIPIOS DE DISEÑO ORIENTADO A OBJETOS
  Descripción: Se deben tener en cuenta distintos aspectos de calidad en el modelado con objetos [Cockburn'99; Meyer'97] y métricas de orientación a objetos, por ejemplo [Lorenz'94]. En esta etapa de la estrategia se utilizan aquellas métricas que se refieren sólo a aspectos de cantidad de responsabilidades y colaboraciones de una clase, como por ejemplo: ¿ tiene demasiadas responsabilidades una componente? ¿Tiene pocas responsabilidades? ¿Qué tipo y grado de cohesión existen entre las clases? (En esta etapa sólo se puede analizar la cohesión de una clase por la relación de sus responsabilidades, ya que la parte interna de la clase aun no ha sido definida). Esta clase de preguntas mejora la calidad de las CRCs. En etapas posteriores se deben aplicar de una forma más sistemática alguno de los sistemas de métricas para objetos propuestos como por ejemplo [Chidamber'94; Lorenz'94], así como también principios generales de diseño OO por ejemplo [Martin'00; Meyer'97].

  HC13: VALIDAR EL MODELO DE CRCS CON EL STAKEHOLDER
  Descripción: Esta heurística ayuda en la validación con el stakeholder, analizando si el modelo conceptual de objetos corresponde a las expectativas y necesidades del mismo, de una forma similar a la propuesta en *XP*. El ingeniero de Software conduce una entrevista con los clientes. Los modelos de LEL, reglas y los escenarios, junto con las relaciones de trace que les permite relacionarse con el modelo de objetos, son usados para chequear si los términos y comportamiento claves del Macrosistema fueron bien interpretados en las CRCs. Por ejemplo: ¿Cómo esta considerada la regla

XX en el modelo de objetos? ¿Cómo se modela un determinado escenario? ¿Cómo se "implementa" este concepto del LEL en el modelo? entre otras preguntas. Si hay un cambio en alguna CRC producto de una mala definición de los escenarios o LEL, éste debe ser reflejado en los modelos correspondientes. Las relaciones backward de traceability de las CRCs ayudan a determinar los modelos involucrados (capítulo 7) ya que permiten obtener las componentes de los modelos de LEL, escenarios y Reglas involucradas en la creación de cada CRC, sus responsabilidades y colaboraciones.

El modelo de CRCs permite descomponer al *UofD* en un conjunto de clases, distribuyendo el comportamiento entre ellas a través de las responsabilidades y colaboraciones. En esta etapa se modelan los aspectos estructurales de estas clases para tener una visión orientada a objetos de los aspectos estáticos. Para esto se utilizan diagramas de clase al estilo *UML*[Booch'98] y diagramas de interconexión. El siguiente SADT muestra los pasos de este proceso de definición

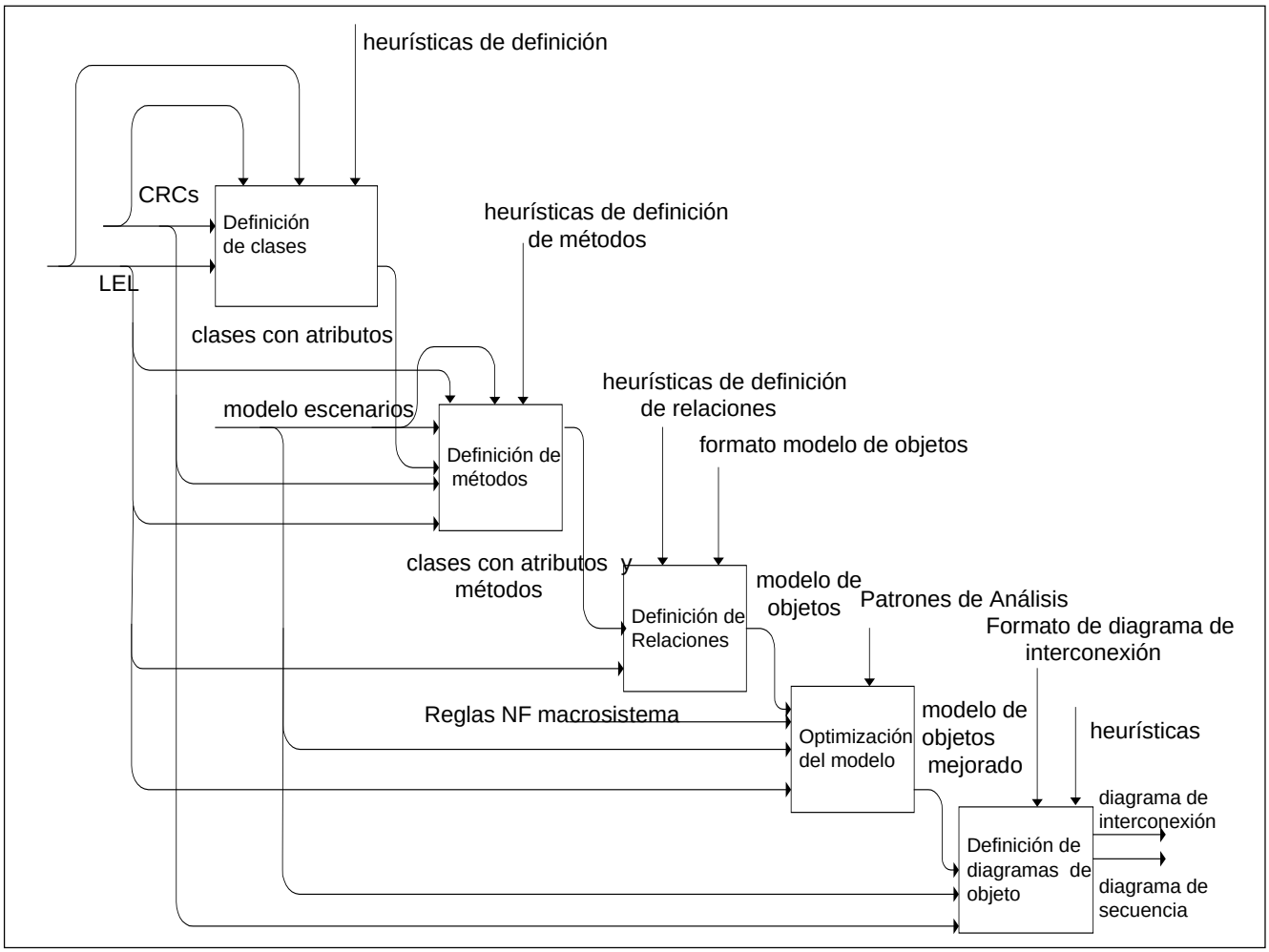


Fig. 5.1: Proceso de definición del modelo lógico

Como se observa en la Figura 5.1, las heurísticas presentadas en este capítulo permiten definir el modelo lógico en base al modelo de CRCs obtenido en el Capítulo anterior y los modelos de la *Requirements Baseline* extendida. La aplicación de estas

heurísticas genera relaciones de traceability entre los modelos generados y los generadores, permitiendo obtener información que justifica la existencia de las componentes del modelo Lógico, como se detalla en el Capítulo 7.

5.1 Definición de un diagrama de clases

En esta sección se definen las heurísticas que definen los aspectos básicos de objetos, es decir las clases, sus atributos, métodos y asociaciones.

HM1: DEFINICIÓN DE LA CLASE Y SUS ATRIBUTOS USANDO LAS CRCS S Y EL LEL
Descripción: Cada *CRC* se convierte en una clase del modelo lógico. Por cada clase se analizan las nociones del término correspondiente en el LEL en busca de propiedades que caractericen los objetos. Es necesario analizar también las responsabilidades (surgidas de los impactos) de la CRCs ya que nuevos atributos pueden aparecer para llevar a cabo la acción de la responsabilidad. En este análisis pueden suceder dos cosas:

HM1.1- MANEJO DE PROPIEDADES SIMPLES
Descripción: La propiedad es modelada directamente como un atributo de la componente, ya que en el LEL representa el nombre de la propiedad
Ejemplo: en las nociones del término Planilla adhesión aparece que una planilla tiene los datos de un solicitante. Se define un atributo datos cuyo tipo será Solicitante.

HM1.2- MANEJO DE PROPIEDADES QUE REPRESENTAN DIFERENTES VALORES**.**
Descripción: En el término del LEL aparecen varias propiedades que modelan diferentes estados o valores concretos del término. Si este es el caso, se puede optar por cualquiera de las dos alternativas:
- la primera es definir un único atributo cuyo dominio sea el conjunto de valores que aparecen en el término.
- La segunda es definir un atributo booleano para cada propiedad. Se puede detectar si existe alguna regla no funcional del Macrosistema que afecte a estos valores, si afecta a alguno de ellos, entonces es mejor modelarlo como un atributo independiente para poder modelar mejor la regla (heurística **HM5.1**). Si el resultado de aplicar **HC3.4 (**ESTADOS COMO CLASES O ATRIBUTOS DE LOS ESTADOS)fue la creación de atributos, la heurística **HM1.2-** es válida para ellos.
En general, por el encapsulamiento de objetos, esta heurística no es muy importante, por lo menos en esta etapa, a menos que se quiera obtener un modelo lógico avanzado. Salvo el caso que se descubra que un atributo tenga una regla o este explícitamente definido, más importante es la interfase del objeto (a través de sus métodos) y no cómo está modelada internamente.
Ejemplo: en el caso del Sistema de Pasaportes se ha decidido que los siguientes estados de un pasaporte (válido, perdido, vencido) serán atributos de un Pasaporte. Pueden modelarse cada uno como un atributo booleano o bien un atributo estado cuyos posibles valores son estos estados. Analizando los modelos, decidimos crear un sólo atributo, estado, para colocar los estados, ya que si bien tienen semántica diferente y provocan diferentes comportamientos, en sí sobre esos estados no existen reglas particulares. Por ejemplo, al estado "vencido" se pasa automáticamente usando la fecha de vencimiento y no existe ninguna regla particular para manejarlo. Lo mismo pasa con el estado "perdido", se pasa a este estado ante una denuncia pero no tiene ninguna regla que lo afecte y que justifique su modelización como un atributo aparte.

HM2: DEFINICIÓN DE LOS MÉTODOS A PARTIR DE LAS RESPONSABILIDADES DE LAS CRCS

Descripción: Cada responsabilidad es una candidata a un método. Sin embargo, al igual que con los atributos, puede haber diferentes alternativas para definirlos. Esta heurística afecta la interfase del objeto, su complejidad y legibilidad por lo que es importante tenerla en cuenta para la definición del modelo lógico.

HM2.1: ANALIZAR RESPONSABILIDADES QUE ACCEDEN AL MISMO ATRIBUTO

Descripción: Varias responsabilidades de una clase pueden modificar de diferente forma al mismo atributo. Es decisión del ingeniero de software definir un método para cada impacto o definir uno solo manejando las diferentes alternativas a través de parámetros. Si bien esto es una decisión de diseño y depende del ingeniero de software, a través de los modelos de escenarios, reglas y LEL, se puede obtener información que ayuda a tomar una mejor decisión. (por ejemplo, cuantas veces se usa cada "método candidato" en los escenarios o si existe alguna regla que afecte a una responsabilidad en particular)

Ejemplo: para el ejemplo planteado en la heurística anterior, se decidió definir un sólo método que cambie el estado del Pasaporte: Pasaporte.cambiarEstado(nuevoEstado)

HM2.2: ANALIZAR LA DIVISIÓN DE UNA RESPONSABILIDAD

Descripción: Puede ser que una responsabilidad sea divida en varios métodos, ya que identifica acciones en esa responsabilidad que son realizadas también en otras responsabilidades. Por esta razón es mejor aislar esta acción y convertirla en un método (se puede indicar como método privado, ya que no surgió de una colaboración entre clases sino de un refinamiento de las responsabilidades de las CRCs aunque la decisión final es dependiente del lenguaje de programación que se elija).

Ejemplo: En el caso de Estudio de la "Auto-aplicación de la metodología" en la clase Ingeniero de Requisitos, existe un refinamiento Eliminar escenarios que aparece en varias responsabilidades: Reorganizar los escenarios por generalización, Reorganizar temporalmente los escenarios. Se define un método EliminarEscenarios para la clase Ingeniero de Requisitos.

HM2.3: COMPLETAR LOS PARÁMETROS DE UNA RESPONSABILIDAD ANALIZANDO LOS ESCENARIOS

Descripción: Para definir los parámetros de los métodos, se analiza el contexto de las responsabilidades de las CRCs, es decir los escenarios en donde aparece la clase que contiene la responsabilidad. Por cada escenario involucrado se determina los datos utilizados para realizar la acción descripta por el método, los cuales pueden ser atributos u objetos.

Ejemplo: analizando el escenario EvaluarLicitación que es el contexto de la responsabilidad CrearRankingAdherentes de la clase Adjudicación, se determina que los parámetros son sobreLicitación, Orden Del Sorteo(no siempre utilizado), cantidad de Adjudicaciones a realizar (dado por la Administradora) y da como resultado el / los ganadores. En este caso los parámetros serán objetos que pueden devolver las propiedades solicitadas. El método es: Licitación. CrearRankinkAdherentes (sobresLic, sorteo, administradora)

HM2.4: COMPLETAR LAS CLASES CON MÉTODOS BÁSICOS

Descripción: Para completar la definición de métodos, se agregan los métodos básicos de acceso / modificación a atributos en caso que no hubieran sido definidos en los pasos anteriores aplicando la sub-heurística **HC5.2**. Se analiza si el atributo no representa una propiedad que no se puede modificar explícitamente (un caso típico es la edad) Si este es el caso, no deben definirse métodos de modificación [Fowler'97].
Ejemplo: En el caso del Sistema del pasaporte, se omite cambiar explícitamente el número de Pasaporte. Un Pasaporte nunca cambia de número, sólo si se emite uno nuevo.

HM2.5: ANALIZAR POSIBLE FACTOREO DE RESPONSABILIDADES
Descripción: Si bien en la definición de las responsabilidades se expresaron concretamente los servicios de cada clase, a la hora de definir los métodos y pensando en una solución orientada objetos, se debe analizar si existen responsabilidades similares. Para esto se debe intentar pensar en las formas más genéricas de las mismas, esto es, si una determinada responsabilidad es por ejemplo "imprimir XX" y otra responsabilidad "imprimir MM", el método será imprimir. De esta forma se reduce el número de métodos por clase y se mejora el futuro diseño, puede ser que las nuevas clases tengan métodos similares y se decida construir relaciones de herencia con componentes genéricas o bien, si esto no sucede, se aplicará polimorfismo. Estas decisiones mejoran una solución orientada a objetos [Cockburn'99]. Este análisis se podría haber realizado cuando se definían las responsabilidades, como se propone en *RDM*, pero se propone introducirlo en esta etapa, ya que se pretende que las CRCs reflejen el Macrosistema en su forma más pura, y estas modificaciones son propias de una solución con técnicas de orientación a objetos.
Ejemplo: Analizando la comunicación fehaciente, se ve que es una clase que solo implementa un mecanismo legal en la organización para las comunicaciones. En las CRCs tienen una responsabilidad para cada tipo de comunicación fehaciente pero se puede pensar en tener un método para la creación de una comunicación fehaciente y por parámetros se le envía cual es la comunicación. De esta forma se simplifica la interfase y la modificación de la clase en caso de agregarse un nuevo tipo de comunicaciónFehaciente.

HM3: DEFINICIÓN DE RELACIONES ENTRE CLASES UTILIZANDO EL LEL Y ESTRUCTURAS SINTÁCTICAS

Descripción: Esta heurística utiliza el principio de circularidad del LEL. Se analizan las nociones de los términos del LEL correspondiente a cada clase. De las nociones se extraen aquellos términos que han sido definidos como clase y se determina la asociación con la clase analizada. La cardinalidad se puede deducir a partir de las nociones del LEL que menciona la asociación o eventualmente analizando todos los escenarios en donde interactúan ambos términos para determinar la cantidad de las entidades involucradas.

HM3.1: DETERMINAR RELACIONES DE HERENCIA
Descripción: Se analiza las nociones y los impactos de cada termino del LEL para el cual se construyó la *CRC*. Este análisis puede ser realizado de igual forma sobre la clase recién definida, sin embargo por la aplicación de heurísticas puede modificarse algunos nombres de métodos, pudiendo provocar confusión si son

tomadas como la fuente para esta heurística. Una relación de herencia puede ser identificada de varias maneras:

- En la noción de los términos del LEL analizados se sugiere explícitamente que dicho término es una especialización / generalización de otro término del LEL. Este concepto está expresado como un patrón lingüístico en Juristo [Juristo'98] identificado con los tipos de verbos *bottom-up* y *top-down* y que son: "ser, ser un tipo, ser una clase, puede ser". Otros tipos de verbos pueden ser "especializar, generalizar o refinar".
    - También debe analizarse dos términos correspondientes a clases relacionadas con algún verbo general que le permita al objeto transformarse. Generalmente es un verbo que cae dentro del contexto del *UofD* (por ejemplo, Adherente que ganó una licitación). Esta estrategia completa la anterior, ya que dependiendo de la forma de escribir hay veces en que los verbos generales no aparecen; sin embargo están implícitos. Seria más correcto escribir el ejemplo mencionado de la siguiente forma: "es el adherente que ganó... ", sin embargo la notación simplificada se entiende y es válida en el lenguaje natural.
    - En diferentes términos del LEL correspondientes a las clases recién definidos aparecen los mismos impactos. Esto sugiere un comportamiento en común que determina se analice una posible relación de herencia. Sin embargo este caso no es común, ya que al modelar relaciones del Macrosistema, el cliente ya define esta relación explícitamente en el LEL usando el principio de circularidad para referenciar al otro término y obviar la repetición de impactos.
    - Un caso particular es cuando aparecen términos correspondientes a estados que han sido modelados como clases (a partir de la heurística **HC.3.3**) Se debe estudiar si los impactos son similares.

En esta etapa de modelado conceptual solo existirán relaciones de herencia que surgen de la realidad modelada y no por una decisión de diseño. En etapas posteriores pueden variar las relaciones de herencia por dichas decisiones.

Ejemplo: en la noción del término Adherente, Solicitante que "es aceptado"...., aparece el término Solicitante con el verbo ser seguido de un verbo que le permite cambiar de estado, lo cual sugiere una relación de herencia entre Solicitante y Adherente. En el término Adjudicatario aparece la noción Es el adherente que ganó.... lo que también sugiere una relación de herencia ente Adherente y adjudicatario.

HM3.2: DETERMINAR RELACIONES RELACIÓN PARTE-DE

Descripción: Esta relación se identifica si en el término del LEL de la clase que seria la Contenedora aparece explícitamente sentencias que sugieran que la clase esta formada por las otra clases, generalmente acompañadas de verbos tipos *"component_composition_verb"* [Juristo'98]: "consistir/ contener/ tener/ poseer/ incluir / formar, componer, dividir" (estos tres últimos en voz pasiva). Asimismo en el término correspondiente a la componente deberá aparecer explícitamente alguna noción que sugiera que esa componente está involucrada en la relación "parte-de" identificadas con los *content_composition_verb*: forma parte, pertenece, es un componente, es incluido, entre otros. Por otro lado, los impactos de la clase candidata a ser la "contenedora" deben ser analizados para determinar si interviene de alguna forma en la creación / destrucción de las componentes. Es más difícil detectar el tipo de composición, (es decir agregación compartida o agregación simple) vs. agregación no compartida o composición, siguiendo la terminología de UML ya que la semántica de esta relación aun está en discusión [Henderson

Sellers'99]. En este sentido solo se puede observar la dependencia de existencia de la primera, la cual puede ser deducida de las nociones e impactos, si este no es el caso se debe buscar en los escenarios en donde aparece el término del LEL correspondiente a la clase componente, para determinar si la vida de la componente está ligada a la vida de la clase que la contiene.

Ejemplo: analizando el término Grupo se ve que hay una noción que indica "*está formado* por adherentes" apareciendo un verbo de la clasificación "component_composition_verb" en voz pasiva. Esto indica claramente una relación "parte-de" entre el Grupo (clase contenedora) y el adherente (componente). Asimismo en las nociones de Adherente aparece la noción "forma parte de un grupo". Esta noción indica la cardinalidad 1 del lado de adherente expresando la exclusividad local [Pirotte'99] que indica que una componente sólo puede ser parte de una sola instancia de la clase Contenedora, en este caso Grupo. Asimismo si se analiza la disolución de un grupo se ve que implica la desaparición de los adherentes, ya que aunque los mismos pasan a ser parte de otro grupo cambian su condición como adherentes del grupo. Esto indica una relación de agregación no compartida.

HM3.3: DETERMINAR RELACIONES DE ASOCIACIÓN Y CLASES ASOCIATIVAS

Descripción: Este análisis puede hacerse analizando las colaboraciones entre las CRCs. Toda colaboración entre clases que no representa una relación del tipo de la dos anteriores, representa una asociación. El verbo que figura en la responsabilidad (categorizado como verbo general en Juristo) determina el nombre para la misma. Si se quiere completar los roles de cada clase en la asociación se puede analizar ambos términos en LEL, obteniendo en cada uno el rol que cumple, en esa asociación. Otra forma es indicar el sentido de la asociación a través de la navegación de UML, indicada con la flecha que muestra el origen y destino de esa asociación. Una asociación especial puede ser la de Identificación [Juristo'98]. Generalmente estas asociaciones no existen, porque raramente se identifican clases que representen los identificadores de otras clases (serán modelados como atributos) sin embargo, puede existir un objeto como identificador. Si existe una clase que identifica a otra, claramente debe aparecer en las nociones de los términos del LEL correspondientes, expresiones con el verbo "Identificar".

Se debe analizar si no existen asociaciones entre clases que comparten una misma jerarquía de herencia y una determinada clase. Si este es el caso debe analizarse la semántica de la relación (a través del LEL y eventualmente en los escenarios en donde aparece), si la asociación es la misma, la relación se traslada a las superclases. En este análisis puede surgir la posibilidad de representar clases asociativas [Booch'98]: Si en el análisis de las nociones se identifican tres términos del LEL y uno de los sirve como conexión de los otros dos, puede ser considerado una clase asociativa (en lugar de relación ternaria) si su existencia depende de la existencia de los otros dos. Esto se decide analizando el término en el LEL. Si este análisis no es suficiente se debe detectar los escenarios donde aparecen las tres clases para ver si instancias de esta clase se crean como resultado de colaboraciones entre las otras dos clases

Ejemplo: Analizando las responsabilidades de la clase adherente se encuentra Pagar cuota en Banco. Se establece una asociación pagar cuota en el Banco. Se puede establecer los roles(en el término LEL del Banco aparece Cobrar cuotas por lo que el Banco tiene el rol de cobrador y el Adherente de pagador / cliente) Sino, a partir de la misma responsabilidad se indica con una flecha el sentido de la

asociación, como se eligió el nombre pagar (ya que se analizó desde el punto de vista del adherente) el origen de la asociación es la clase Adherente y el destino Banco.

5.2 Modificación del diagrama a partir de Patrones de análisis y el Modelo de Reglas No funcionales

HM4: APLICAR PATTERNS DE ANÁLISIS AL DIAGRAMA DE CLASE
Descripción: el modelo lógico puede mejorarse aplicando patterns de análisis [Coad'95; Fowler'96; Fowler'97] en caso que sea posible. Si bien el modelo preliminar debe mantenerse porque es el que refleja de manera más pura al Macrosistema, es útil adaptarlo según patrones ya existentes ya que estos, además de brindar soluciones probadas a determinados problemas, expresan un modelo más cercano a la etapa de diseño. De los patrones de análisis conocidos se pueden utilizar los patrones independientes de dominio. No se puede especificar una heurística para ayudar a determinar un patterns en particular, sólo es posible indicar que tipo de posibles patterns se puede aplicar a cada tipo de término (por ejemplo los patterns de roles se pueden aplicar en los términos correspondientes a objetos, los de calendario en frase verbales).

HM4.1: APLICAR PATTERNS DE ROLES
Descripción: Determinar posibles roles entre los términos analizados que corresponden a Sujetos en el LEL. Pare esto se debe analizar las nociones de los términos correspondientes a los objetos definidos. En principio, se proponen dos formas de modelar roles:

- Si la persona / organización cumple un sólo rol, éste se modela según el pattern de Persona- Participante [Coad'95]: las características propias de la persona son modeladas como una clase y el rol que cumple en el Macrosistema se define como otra clase.
- Si la persona / organización cumple varios roles, se pueden optar por los patrones propuestos por [Fowler'97]. El análisis de los impactos de los términos del LEL así como también los escenarios para determinar las colaboraciones, ayudan a determinar las similitudes y diferencias existentes entre los roles involucrados, para optar entre las distintas alternativas.

Ejemplo: en el caso de estudio de la Administradora existen las personas que pueden tener varios roles (adherente, solicitante adjudicatario). Se aplica el pattern de rol "object rol"[Fowler'97] modificando el diagrama de objetos como se indica en la siguiente figura.

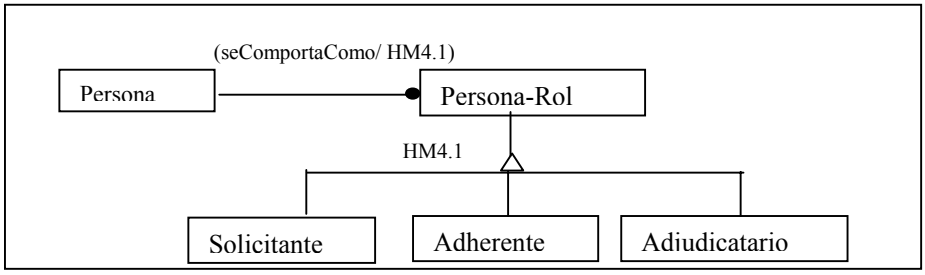


Fig. 5.2: modelo de objetos modificado por el patrón "object rol"

HM4.1: APLICAR PATTERNS DE SCHEDULING
Descripción: Para aquellos términos que corresponden a frases verbales (modelados como clases o métodos) debe buscarse en los contextos de los

escenarios en donde aparecen, posibles estados temporales en que suceden. Si los eventos suceden bajo determinados patrones de tiempo, pueden utilizarse los patrones de scheduling [Fowler'97].

Ejemplo: en el caso de Estudio de la Administradora la regla 31 determina que el sorteo y la licitación deben ser el primer día hábil de cada mes. Se mejora el modelo aplicando un pattern de scheduling (ver capítulo 8).

El resto de los patrones de análisis independientes de dominios (como por ejemplo todos los relacionados con propiedades de objetos) se aplican en general a cualquier objeto independientemente del tipo de término del LEL de donde proviene.

HM5: APLICAR EL MODELO DE REGLAS NO FUNCIONALES DEL MACROSISTEMA A LOS DIAGRAMAS DE CLASES

Descripción: Así como el modelo de reglas funcionales puede generar y/o modificar responsabilidades o clases, las reglas no funcionales introduce modificaciones en los diagramas para permitir modelar las reglas de negocio de la organización que afectan la estructura y el comportamiento de los componentes. La incorporación de las reglas en el modelo de objetos tiene como objetivo identificar las componentes, y más concretamente, los métodos, atributos, clases y relaciones entre clases que son afectados por cada regla. De esta forma se facilita la tarea al diseñador del modelo físico, ya que se brinda la información necesaria en el modelo lógico. En los diagramas de objetos, las reglas pueden ser expresadas utilizando el lenguaje OCL [Warner'99]

A continuación se muestra cómo los diferentes términos que aparecen en las reglas pueden modelarse como clases, atributos, métodos (o responsabilidades) u asociaciones. La sintaxis de la Regla no-funcional del Macrosistema es:

[Propiedad] + Frase no-verbal + Relación + [Propiedad] + Frase no-verbal

donde la explicación de cada componente puede encontrarse en el capítulo 3.Aplicando el conjunto de heurísticas las componentes pueden transformarse en:

Propiedad se puede transformar en	atributo / Clase
frase no-verbal se puede transformar en	Clase/ atributo
Relación se puede transformar en	Método/ Clase

Desde el punto de vista de objetos, a partir de esta estructura pueden surgir diferentes combinaciones que son afectadas por las reglas para las cuales habrá distintas heurísticas:

a) Reglas que afecten a una sola Clase **(HM5.1)**
b) Reglas que afecten a Clase con Clase **(HM5.2)**
c) Reglas que afecten a Atributo de una Clase [con Atributo de una Clase] (de una misma clase o diferente) **(HM5.1) (HM5.2)**
d) Reglas que afecten a método de una Clase [con método de una Clase] (puede estar la clase implícita si se escribió al método como Frase-no verbal) **(HM5.3)**
e) Reglas que afecten a métodos y atributos (puede estar la clase implícita) **(HM5.3)**

HM5.1: ANALIZAR REGLAS QUE AFECTAN A UNA SOLA CLASE

Descripción: Puede suceder que afecte a la clase como un todo o a atributos de la clase:

- Cuando la regla afecta a la clase como un todo, la única característica que puede afectar como tal es la cardinalidad, ya que cualquier otra

característica implicaría un atributo, clase o método (por ejemplo una propiedad de pertenencia implica necesariamente otra clase, la clase que lo contiene o la contenedora). No es una situación frecuente. Esto se puede expresar en OCL con el método size de Collection (className.size -> cantidad de instancias de la clase)

- Si es un atributo se marca en la clase como una invariante de atributo. Se debe buscar si existen responsabilidades (y por ende métodos) implícitos en la regla. Para esto se analiza si la regla no tiene asociada una regla funcional o directamente se busca en las responsabilidades de las CRCs correspondientes. Si se encuentra se aplicará sub-heurística **HM5.3.**

Ejemplo: La Regla 39 "Un adjudicatario puede rechazar hasta tres veces el bien tipo" indica que un adherente sólo puede rechazar sólo tres veces la adjudicación". Existe regla funcional asociada, la 28 (la cual se ejemplificará en sub-heurística **HM5.3**). La Regla se expresa de la siguiente forma:

```
Context : Adherente
        cantidadRechazos <= 3
```

Esta restricción puede ser mejorada indicando un atributo en la Administradora (cantidad RechazosPermitidos = 3) y modificando la restricción:

```
Context : Adherente
cantidadRechazos <=  Administradora. cantidadRechazosPermitidos
```

HM5.2: ANALIZAR REGLAS QUE AFECTAN A VARIAS CLASES

Descripción: Debe tenerse en cuenta si afecta a la clase como un todo (ídem anterior) o a atributos de la clase (situación más frecuente)

- Si la regla afecta a dos clases, puede darse el caso de cardinalidad (no necesita una relación) o una restricción sobre relaciones de composición, o generalización entre ellas. Estas asociaciones ya fueron generadas anteriormente, a través del análisis de las nociones del LEL a través de la heurística **HM3**. Si este no es el caso, debe agregarse. Se indica la restricción en la asociación ente las clases en el diagrama de objetos. En el caso de una asociación común, si existe una restricción, será modelada como atributo (ver siguiente opción) o método (**HM5.3**)
- Si es entre atributos de clases, se debe analizar si existe una asociación que vincule esos atributos y sobre la cual se puede expresar esa asociación. Si no existe, se puede modelar a nivel de atributos en una de ellas o en las dos (depende si existen restricciones sobre ambas) usando una expresión OCL.

Ejemplo: La regla 67: "La cantidad de miembros de un grupo es el doble de la cantidad de cuotas del plan elegido" determina la cantidad mínima de adherentes de un grupo que debe tener un plan de Ahorro que puede ser expresado de la siguiente manera:

```
context: Plan de ahorro
 tamañoMinimoGrupo := self.cantidad cuotas  * 2

context: Grupo
(Adheretentes.size + Adjudicatarios.size) >= Plan de ahorro.tamañoMinimoGrupo
```

HM5.3: ANALIZAR REGLAS QUE AFECTAN A MÉTODOS

Descripción: En la regla aparecen los métodos o responsabilidades de las CRCs. Si la regla tiene asociada alguna responsabilidad de la CRCs, se debe buscar los métodos asociados a la regla, el cual puede estar en la misma clase a la que pertenece el atributo o bien en otra clase que acceda mediante un método a ese atributo. En ese método se pueden especificar las pre/post asociadas a la regla. Se debe analizar si es necesario o no agregar un atributo a la clase del método.

Ejemplo: Continuando con el ejemplo de la sub-heurística **HM5.1,** la regla 28 "Si el adjudicatario rechaza por tres veces el bien tipo la Administradora puede expulsarlo del Grupo" indica el comportamiento asociado a la restricción de rechazo. Se indica en el método correspondiente.

```
Adherente:
RechazarBienTipo
post=  pre@cantidadRechazo = 3 implies adherenteGrupo -> includes (adherente)
       = false
```

5.3 Definición de diagramas de interconexión y secuencia

El modelo obtenido con las heurísticas anteriores corresponde a un diagrama de objetos [Booch'98] donde se muestran los objetos, sus métodos atributos y relaciones. El siguiente paso es la definición de un modelo que muestra el comportamiento del modelo de objetos y los datos que se envían. Puede hacerse a partir de dos opciones: la primera es definir un diagrama de interconexiones que muestre los datos que se pasan los objetos de manera global. Este diagrama es útil si el modelo no es complicado y junto con las CRCs da una visión pura orientada a objetos del problema. Si el *UofD* es complejo y existe mucha intercambio de datos, se puede optar por una "realización" de cada escenario, al estilo *RUP/UML* y definir un diagrama de secuencia para cada escenario en donde se ve cómo se comportan e interactúan los objetos para llevarlo a cabo.

HM6: DEFINIR UN DIAGRAMA DE INTERCONEXIÓN A PARTIR DEL DIAGRAMA DE CLASES Y LOS ESCENARIOS
Descripción: A través de las CRCs se pueden determinar las colaboraciones entre clases. Sin embargo, falta establecer los datos que se pasarán entre las mismas en cada colaboración. Por esta razón se realiza un diagrama de interconexión que muestra el pasaje de datos entre los objetos.

Para cada clase
  Para cada método
    Analizar las entradas y salidas del método (a partir de la descripción de los métodos del modelo estructural o analizando nuevamente los escenarios involucrados)
    Dibujar líneas de conexión con las clases colaboradoras
    Escribir los parámetros con las clases con flecha indicando la clase destino

Ejemplo: Dada la clase Administradora y los métodos "DeterminarCambioBienTipo" y "LiquidarGrupo"(entre otros) en donde aparece Grupo como colaborador. Analizando los escenarios "Cambiar bien tipo", "Disolver Un Grupo" que son contextos de las responsabilidades asociadas a cada método, se encuentra que los datos que se pasan son: Administradora le envía el aviso Disolución del Grupo y Cambio del Bien Tipo y el Grupo le envía el Rechazo / aceptación del bien Tipo. En el capítulo 8 se define un diagrama de interconexión para el caso de estudio del Círculo de ahorro.

HM6: MODELAR LOS ESCENARIOS POR MEDIO DE DIAGRAMAS DE SECUENCIA
Descripción: los diagramas de secuencia muestran la interacción de los objetos en una secuencia de tiempo, detallando la secuencia de mensajes que intercambian los objetos. Este diagrama es uno de los más usado en *RUP/UML* para especificar los *Use-Cases.*

Para cada Escenario

Para cada episodio

1. Analizar si los sujetos están representados como clases primarias

Crear una línea con el nombre de la clase primaria para representar al rol que realiza la actividad

sino

Definir las clases primarias correspondientes (esto implica modificar el modelo de LEL que omitió el sujeto correspondiente) y retomar este proceso.

2 Analizar si los recursos usados en el episodio son las clases secundarias

2.1 Crear una línea con el nombre de la clase secundaria para representar al rol que realiza la actividad

2.2 Si el objeto es creado como consecuencia de ese mensaje colocar la línea y el rectángulo superior que indica el nombre al final de la flecha que indica el envío del mensaje.

sino

Definir las clases secundarias correspondientes (esto puede implica modificar el modelo de LEL) y retomar este proceso.

3- Analizar si cada actividad del episodio está representada en las responsabilidades de las clases correspondientes

3.1 Cada responsabilidad se transforma en un mensaje. Crear una flecha con el mensaje que envía la clase que requiere de los servicios a la clase que lo realiza, ambas representadas con líneas.

3.2 En caso de ser necesario definir los datos de entrada en el mensaje y los de salida en las líneas punteadas que vuelven a la clase emisora

3.3 Si existe sentencias condicionales o restricciones del contexto, indicar las "guard" correspondientes en cada flecha alternativa que salen de un mismo punto.

sino

Agregar responsabilidades (esto puede implicar agregar impactos al LEL) y retornar este proceso.

Las sentencias condicionales de este proceso tienen sentido si no se han realizado validación de las CRCs(heurística **HC11**) para comprobar que existen todas las clases relevantes al *UofD* y con las responsabilidades adecuadas para poder realizar el comportamiento descrito en los escenarios.

Capítulo 6
Definición de un modelo preliminar de objetos de software

6.1 Identificación de los límites del software

Los requisitos de software son establecidos por una negociación entre el cliente y los ingenieros. Las negociaciones nunca son conducidas utilizando únicamente argumentos lógicos y técnicos, siempre están influenciadas por las personalidades de las personas involucradas y por las consideraciones organizacionales y políticas de la empresa [Sommerville'97]. Dentro de las guías que propone Sommerville para determinar los requisitos de software, está la guía 5.5 "*Priorisite Requirements*" que ayuda a los stakeholders a definir el núcleo de los requisitos para el sistema de software. Esta guía puede ser llevada a cabo si existe un análisis previo del sistema y un conjunto razonable de requisitos definidos, ya que de otra forma se dificulta la tarea de asignar prioridades al no disponer de una "pintura completa" de todos los requisitos. En este contexto, creemos que el modelo de Reglas del Negocio puede ser usado para ayudar al cliente a determinar sus prioridades. Este modelo es clave para negociar con el cliente, ya que le ayuda a determinar qué se necesita automatizar sobre la base de las prioridades de las políticas de la organización. De esta forma, el modelo de reglas es utilizado para decidir qué se quiere automatizar mientras que las CRCs y el modelo lógico ayudan a estimar tiempo y costos. El resultado surge de una negociación entre los ingenieros y los clientes.

El objetivo de este capítulo es demostrar como el uso de los modelos de la *Requirements Baseline*, en particular el modelo de Reglas, puede ayudar en algunos de los aspectos de la negociación que es la determinación de prioridades y sobre la base de esto la definición de los limite de software. Las heurísticas que se presentan aquí pueden ser complementadas con la Regla 5.1 "System bounderies" [Sommerville'97] en donde se plantea una guía para ayudar a determinar, en base a las características de los requisitos, cuáles de ellos serán del sistema de software, cuales del sistema operacional en donde el software estará inserto y cuales quedarán afuera del sistema. En este capítulo no se exploran todos los aspectos de la negociación de los requisitos, los cuales son muchos y complejos, como por ejemplo el tratamiento de conflictos entre stakeholders, el cual necesita un estudio y análisis detallado, que escapa al objetivo de esta tesis.

- **HL1:** UTILIZAR EL MODELO DE REGLAS FUNCIONALES PARA GUIAR LA DEFINICIÓN DE LOS LÍMITES DEL SOFTWARE.
  Descripción: las reglas del negocio son "parte del corazón" de cualquier organización [Gottesdenier'97]. Al ser orientadas al cliente, escritas en lenguaje natural y declarativas, el cliente fácilmente puede analizarlas y determinar cuáles son las que desea implementar en cada negociación. A partir de allí, a través de las relaciones de

trace (capítulo 7) se determinan cuáles son las clases de software que las implementarán. La estrategia es la siguiente:

1. El cliente debe definir los Requisitos del Sistema de Software, estableciendo qué debe hacer el software. El cliente establece prioridades en las reglas funcionales de la organización que le ayudarán a determinar qué se quiere automatizar.

2. Se identifican los términos del LEL que parecen en las reglas seleccionadas. A partir de las relaciones de trace se determinan las CRCs creadas a partir de las reglas y a partir de los términos del LEL. Por cada CRC, se determina las clases del modelo lógico.

3. Definir las clases de software a partir de las clases del modelo lógico
   Para cada clase
      Para cada método
         Determinar si se implementa sobre la base de las prioridades de las Reglas
         Si se implementa
            Utilizando este procedimiento, analizar las clases asociadas a la analizada en el diagrama de asociaciones.
         Si no se implementa
            Determinar la comunicación con los objetos de software, que constituirán las entradas y salidas del sistema de software.

En esta etapa es útil volver a analizar los escenarios en donde aparecen los términos del LEL que corresponde a las clases para completar la información del comportamiento asociado a cada responsabilidad analizando el contexto del Macrosistema en donde acontece cada una de ellas. Para eso se utilizan las relaciones de trace backward que relacionan a las *CRC* con los términos del LEL y las relaciones intra *Requirements Baseline* que relaciona los términos del LEL con los escenarios(capítulo 7).

Dentro de los métodos de los objetos que están dentro de los límites del software, pueden surgir modificaciones. Al estar modelando el Macrosistema sin tener límites de software y representar a los Sujetos del LEL como clases, es altamente probable que exista comportamiento referido a pasaje de información, es decir un actor pide información de un recurso y lo provee a otro, sirviendo solo de nexo. Si bien este modelo es el reflejo del mundo real que se está definiendo, en este punto se puede pensar que con la tecnología de objetos, todos los objetos son "activos" en el sentido que ellos mismos pueden requerir la información necesaria para llevar a cabo sus tareas. Por esta razón, sino se altera ninguna regla del negocio, se puede reemplazar las relaciones que sólo funcionan como intermediarias para llevar la información a otra clase. Es decir, si la clase A le manda información a la clase B y ésta se la envía sin procesar o registrar a la clase C, se reemplazan estas dos responsabilidades por una sola que representa el envío de la información necesaria entre A y C.

- **HL2:** UTILIZAR EL MODELO DE ESCENARIOS PARA GUIAR LA DEFINICIÓN DE LOS LÍMITES DEL SOFTWARE.

Descripción: Otra alternativa es la utilización del modelo de escenarios. Esta estrategia, similar a la propuesta en *RUP/UML* con respecto a los *Use-Cases*, es definir junto con el cliente un conjunto de escenarios candidatos a ser implementados y a partir de ahí determinar con las relaciones de trace las futuras clases de software. Sin embargo,

determinar un conjunto significativo no siempre es fácil y por eso el uso del modelo de reglas puede ayudar a definirlo.

1 Determinar los escenarios más significativos utilizando las Reglas
2 Para cada escenario
    A través de las relaciones de trace identificar su diagrama de secuencia (obtenido con la heurística **HM6**)
    Para cada clase del diagrama de secuencia
        Para cada mensaje
            Determinar si se implementa
                Analizar las clases de las colaboraciones y determinar si se implementan (dependiendo de su participación en el conjunto de escenarios elegidos)
            Si no se implementa
                Determinar la comunicación con los objetos de software, que constituirán las entradas y salidas del sistema de software.

HL1 y **HL2** pueden ser usadas de manera complementaria, tomando como guía **HL1**, los pasos de **HL2** pueden ser usados para cuando se desea analizar una clase en particular, ya que en el diagrama de secuencia, se puede estudiar en detalle como interactúa con las otras clases. Para esto se necesita usar las relaciones de trace que permitan establecer para una clase en particular los escenarios (y su diagrama de secuencia) que figuran en el contexto de la CRC correspondiente.

- **HL3:** DEFINIR UN DIAGRAMA DE NIVEL

Descripción: un diagrama de nivel [Castro'94] permite reflejar el sistema de software, los sistemas externos y los flujos de comunicación entre las entidades externa al sistema y las clases internas al mismo. Este diagrama permite visualizar rápidamente los límites del software y su comunicación con el medio externo, definido éste en diferentes capas que finalmente abarca todo el Universo del Discurso en estudio.

Se encapsulan todos las clases del sistema de software obtenidas anteriormente y se crea el sistema de software. Se modelan las entidades externas y se modelan los flujos que van hacia y desde el sistema de software y los flujos de información entre las entidades externas. Los flujos de datos se obtienen del diagrama de asociación.

6.2 Aplicando Reglas no Funcionales de Calidad

- **HL4:** INCORPORACIÓN DE REGLAS NO FUNCIONALES DE CALIDAD

Descripción: Las reglas no funcionales de calidad determinan los Requisitos No-funcionales (RNF). Por esta razón, en esta etapa puede utilizarse la propuesta de [Neto'00] para incorporar los RNF en el modelo de objetos. Si la restricción no puede ser aplicada a una clase, método o atributo según la propuesta de Neto, significa que involucra la arquitectura de solución, por lo que es descripta en una lista aparte y adjuntada al modelo lógico. Es importante resaltar que esta integración sólo indica las clases, métodos y atributos involucrados en las reglas, sin realizar modificaciones agregando nuevos objetos, dividiendo los existentes o eliminándolos según sea la restricción a cumplir. En etapas de diseño, sobre la base de esta información, se utilizará alguna estrategia para cumplirlas, teniendo en cuenta como afectan las reglas en el modelo, por ejemplo si afectan a un objeto o varios métodos de diferentes objetos. En particular existen patterns

arquitecturales y de diseño que cubren determinadas restricciones [Buschmann'96], como por ejemplo los patterns *Bridge* [Gamma'94] y *Reflection* que cubren aspectos de "changeability", el Broker pattern que permite cumplir el requerimiento de "interoperability", finalmente aspectos de tolerancia a fallas y robustez son atacados con el *master-slave* [Buschmann'96].

Capítulo 7
Traceability de los modelos

Sea cual fuere el ciclo de vida utilizado, los requisitos deben poder rastrearse hacia su origen o a partir de ellos para verificar que se están produciendo el diseño y la implementación correctas y que las técnicas para elaborarlas son las adecuadas; esto se conoce con el nombre de *pre* y *post traceability* [Gotel'94]. La traceability da una asistencia fundamental en el entendimiento de las relaciones que existen intra requisitos, con los modelos de diseño y con los modelos de implementación, y es crítica para el desarrollo de software porque provee un mecanismo para analizar el porqué y como un determinado sistema de software satisface los requisitos de los clientes. Provee un medio para validar el grado de satisfacción de las necesidades de los clientes y brinda información para procedimientos de testing, medidas de performance, características no-funcionales y de comportamiento del sistema en desarrollo [Palmer'97]. Es necesaria para proveer un rápido acceso a la información, para abstraer información y proveer visualización de las técnicas utilizadas en el desarrollo de software. Por otro lado, la traceability permite que las decisiones que se tomen en estados más tardíos del desarrollo del sistema sean consistentes con las tomadas en los primeros momentos. Los beneficios de la traceability son a largo plazo, cuando es necesario introducir cambios una vez que el sistema ya esta en marcha. La administración de Traceability se realiza a lo largo de todo el proceso de desarrollo de software y una de sus funciones más importante es en la etapa de mantenimiento, cuando nuevos requisitos son agregados y los viejos pueden ser modificados o borrados. A partir de la información rastreada, se puede hace un análisis del impacto y establecer si la modificación se incluye adecuadamente en los modelos obtenidos en el proceso de desarrollo.

En este trabajo no se propone un modelo general de Traceability como los propuestos en [Pinherio'96; Pohl'96] sino que se define las relaciones de trace entre los modelos generados por la aplicación de las heurísticas, estableciendo una relación de TRACE similar a la propuesta por *UML*, pero a diferencia de ésta, se consideran las relaciones de trace entre documentos de un mismo modelo o etapa, en este caso, los documentos del modelo conceptual de objetos. Asimismo es importante destacar que se define relaciones de trace referidas a una evolución de desarrollo, y no desde una perspectiva de mantenimiento, que podría cubrirse con la Vista de Configuración (capítulo 2) que se extienda hacia el modelo conceptual. En [Leonardi'98^{c}] se presenta una versión preliminar al modelo presentado en esta tesis, mientras que en [Leonardi'00] este modelo es extendido a un modelo general de requisitos que permita traceability.

7.1 Relaciones de trace entre las modelos generados por la estrategia

En este capítulo se consideran dos clases de relaciones de trace: por un lado, las relaciones de trace que existen entre los modelos de la *Requirements Baseline* y los modelos conceptuales Orientados a Objetos, y por el otro relaciones internas en este último modelo. Las relaciones intra-modelo de la *Requirements Baseline* ya están

consideradas por la Vista de Hipertexto, mencionada en el capítulo 2 y extendida hacia el modelo de Reglas de Negocios con la misma filosofía, es decir controladas por los términos del LEL. La figura 7.1 muestra el meta-modelo de los modelos utilizados en esta estrategia y sus componentes. Las clases involucradas son:

- Clases correspondientes a los modelos de la *Requirements Baseline:* `LEL`, (con sus componentes `Impacto` y `Noción`), `Escenario` y `ReglasNegocio` (con su jerarquía de `RF` para las reglas funcionales y `RNF` para las reglas no funcionales, quien a su vez se divide en `Rcal` y `Rmac` para distinguir las reglas de calidad y las del Macrosistema)
- Clases que representan a los modelos y sus componentes correspondiente a la Definición de las CRCs: `CRC`, `Responsabilidad`
- Clases que representan a los modelos y sus componentes correspondiente a la Definición del Modelo Lógico: `Clase`, `Asociación`, `Método` y `Atributo`.

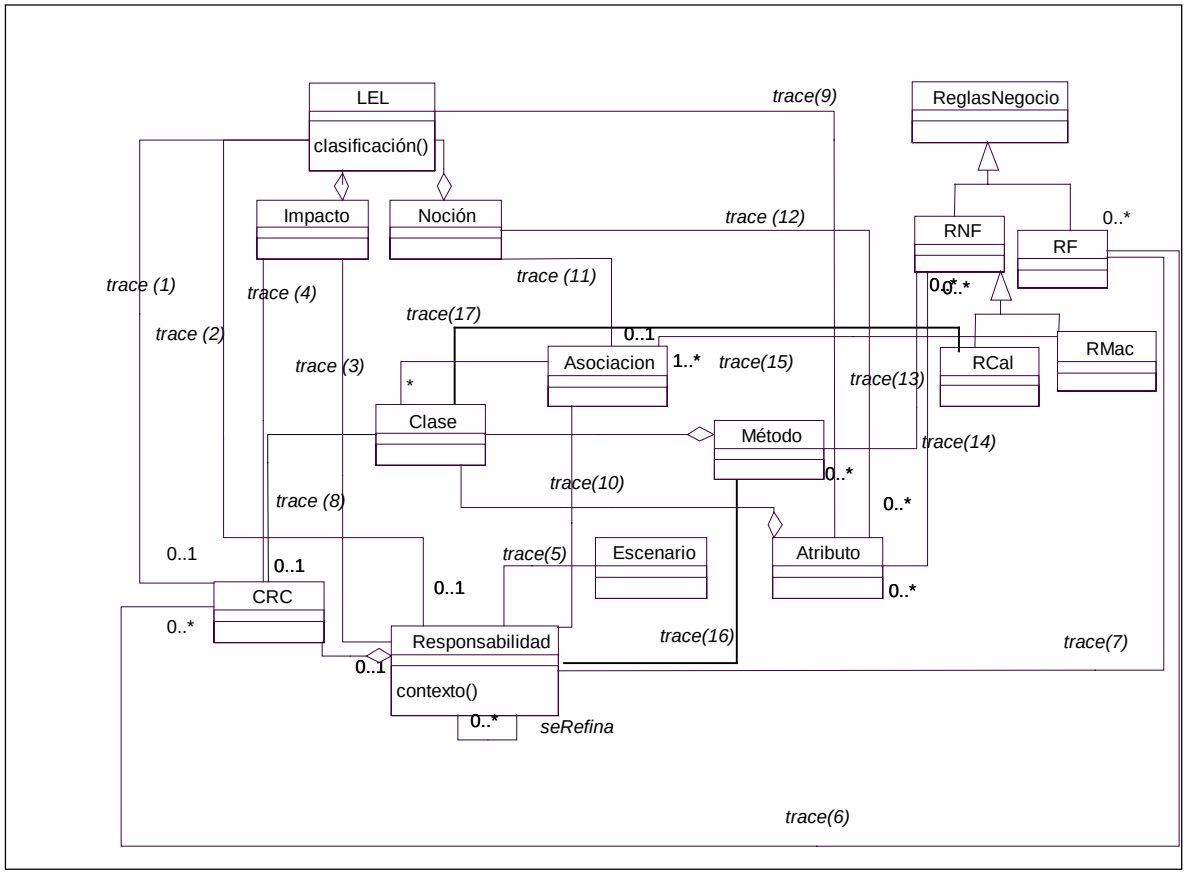


Fig. 7.1 Relaciones de traceability entre las componentes de la Estrategia

Las relaciones de trace de la Figura 7.1 corresponden a las relaciones surgidas por las heurísticas entre el modelo o componente que sirvió de origen y el modelo o componente que creó o modificó. Por ejemplo, en el caso de Estudio del Capítulo 8 se puede mencionar una relación entre el término del LEL Adherente y la CRC Adherente surgida de la heurística **HC1** reflejada por la relación `trace(1)` en la Figura 7.1. Otro ejemplo, en un nivel más detallado, es una relación entre el atributo "planes de Ahorro" de la clase Administradora que fue originado de la noción "empresa que se encarga de organizar planes de ahorro..." del término Administradora a partir de la heurística **HM1** reflejada en la Figura 7.1 por la relación de `trace(12)`. También las relaciones señalan una modificación entre componentes, por ejemplo, la regla "Si el adjudicatario rechaza por tres veces el bien tipo la Administradora puede expulsarlo" afecta al método "RechazarBienTipo" de la clase Administradora a partir de la heurística **HM5.3,** reflejada

en la figura por la relación `trace(13).` Las relaciones de trace representan una asociación directa entre componentes o modelos. No se modelan relaciones indirectas. Por ejemplo, la clase `"clase"` de la Figura 7.1 correspondiente al modelo lógico no se relaciona directamente con la clase `escenario,` sino que esta relación se obtiene transitivamente por la relación de `trace(8)` que existe entre la `clase` y la `CRC`, la relación `trace(5)` que existe entre la `CRC` y los `escenarios` a través de las `responsabilidades` que son una componente de la clase `CRC`.

En la figura 7.1 sólo se señala las posibles relaciones de trace que genera la aplicación de las heurísticas de la estrategia entre los componentes del meta-modelo. Cada una de estas relaciones tiene su propia semántica, pudiendo existir más de una relación entre las mismas componentes, dependiendo de las heurísticas(crea o afecta). Las figuras 7.2 y 7.3 muestran la semántica de cada relación de trace entre los modelos generados. Se definió una tabla donde la columna izquierda representa el Origen de la relación de trace y la fila superior el Destino. Las celdas con información indican una relación de trace. En las figuras están consideradas las relaciones que dan origen o afectan a la nueva componente, es decir relaciones forward [Davis'93]; sin embargo, a partir de ellas también se puede plantear relaciones backward, como se explica en la próxima sección. Por cada relación de trace se indica:

- **Semántica de la relación:** como afecta la componente origen en la componente destino (crea, seModelaComo, puedeModelarseComo, apareceEnContexto, seTransforma, afectaA).
- **Cardinalidad de origen:** cuantas componentes se utilizaron para crear / modificar la nueva componente
- **Cardinalidad destino:** cuántas componentes se crearon o afectaron en esa relación.
- **Heurísticas** que originaron la relación de trace. En el caso que para una determinada relación de traceability intervienen todas las sub-heurísticas, sólo se indica la heurística correspondiente.
- **Relación de trace** correspondiente a la Fig. 7.1

La Figura 7.2 muestra las relaciones de trace entre la *Requirements Baseline* extendida y el modelo de CRCs y la Figura 7.3 las relaciones de trace con respeto al modelo lógico.

Destino **Origen**	**CRC**	**Responsabilidad**	**Refinamiento Responsabilidad**
Término LEL Sujeto	SeModelaComo(1/1) **HC1** **(trace 1)**		
Término LEL Objeto, o Estado	PuedeModelarseComo (0,1/ 0,1) **HC3.1 HC3.3** **(trace 1)**		
Termino LEL Frase Verbal	PuedeModelarseComo (0,1/ 0,1) **HC3.2** **(trace 1)**	puedeModelarseComo (0,1/ 0,1) **HC3.2** **(trace 2)**	
Impacto de Termino LEL	PuedeModelarseComo (0,1/ 0,1) **HC7** **(trace 4)**	PuedeModelarseComo (0,1/ 0,1) **HC2 HC4 HC5** **(trace 3)**	

Fig. 7.2: Relaciones forward de trace con respecto al modelo de CRCs

Destino **Origen**	**CRC**	**Responsabilidad**	**Refinamiento Responsabilidad**
Escenario		ApareceEnContexto (1,+ / 1,+) **HC5**	ApareceEnContexto (1,+ / 1,+) **HC5**

		(trace 5)	**(trace 5)**
Regla Funcional	puedeModelarseComo (0,+/ 0,+) **HC9.2 (trace 6)**	puedeModelarseComo (0,+/ 0,+) **HC9 (trace 7)**	puedeModelarseComo (0,+/ 0,+) **HC9 (trace 7)**
Vocabulario básico	puedeModelarseComo (0,+/ 0,+) **HC6**		

Fig. 7.2: Relaciones forward de trace con respecto al modelo de CRCs (continuación)

Destino Origen	**Clase**	**Método**	**Atributo**	**Asociación**
CRC	seTransforma (1/1) **HM1 (trace 8)**			
Pattern de Análisis	Crea(0,+/0,+) **HM4**			
Responsabilidad (con colaboración)		puedeModelarseComo (0,+/ 0,+) **HM2 (trace 16)**		puedeModelarseComo (0,1/ 0,1) (1) **HM3.3 (trace 10)**
Refinamiento Respons. (con colaboración)		puedeModelarseComo (0,1/ 0,1) **HM2.3 (trace 16)**		puedeModelarseComo (0,1/ 0,1) (1) **HM3.3 (trace 10)**
Término LEL = objeto o estado			PuedeModelarse Como (0,1/ 0,1) **HC1.2 (trace 9)**	
Noción Término LEL			origina (0+ / 1+) **HM1 (trace 12)**	puedeModelarseComo (0,1/ 0,1) (1) **HM3.1 HM3.2 (trace 11)**
Regla No Func. Calidad		afecta (0+/0+) **(H límites soft) (trace 13)**	afecta (0+/0+) **(H límites soft) (trace 14)**	
Regla No Func. Macrosistema	afecta (0+/0+) **HM5.1 HM5.2 (trace 17)**	crea (0+/0+) ó afecta (0+/0+) **HM5.3 (trace 13)**	crea (0+/0+) ó afecta (0+/0+) **HM5.1 HM5.2 (trace 14)**	afecta (0+/0+) **HM5.2 (trace 15)**

Fig. 7.3 Relaciones forward de trace con respeto al modelo lógico

(1) estas relaciones de trace son válidas para aquellas nociones / impactos que hacen referencia a otros términos del LEL diferentes del que describen y que hayan sido definidos como clases.

7.2 Modelización de las relaciones de trace: una Vista de Trace

Con una filosofía similar a la propuesta para la Vistas de Configuraciones de la *Requirements Baseline* (capítulo 2), se plantea una Vista de Trace para reflejar las relaciones entre las clases de la Figura 7.1 surgidas de la aplicación de las heurísticas de la estrategia. La figura 7.4 define a una vista de Trace.

Trace:
Nombre: indica la semántica de la relación.
Justificación: explica el porqué de esa relación (conjunto de heurísticas)
Responsable: indica la/s persona/s responsable/s de la aplicación de las heurísticas.
Origen: instancia de alguna clase de la Figura 7.1. Parámetro de Entrada del contexto de Navegación.
Destino: instancia de alguna clase de la Figura 7.1.
CardinalidadDestino: cantidad de instancias destino que origina esa relación.

Fig. 7.4 Vista de Trace

Una instancia de una vista de trace relaciona instancias de las clases presentadas en la figura 7.1 que están relacionadas según se indica en las Figuras 7.2 y 7.3. A continuación se detalla cada uno de los atributos de la Vista de Trace presentada en la Figura 7.4.

Nombre: como se puede observar en los Capítulos 4, 5 y 6 y se resume en las figuras 7.2 y 7.3 la aplicación de las diferentes heurísticas relacionan modelos, ya que a través del uso de una heurística en particular, un determinado modelo (o una componente del mismo) crea o modifica otro modelo (o componente), asociando de esta forma los diferentes modelos y sus componentes usados en esta estrategia. Las diferentes relaciones que se pueden dar son: crea, seModelaComo, puedeModelarseComo, apareceEnContexto, seTransforma, afectaA. Estos nombres surgen de la semántica de cada heurística con respecto a los modelos que se usan como origen y los modelos resultado, como se indica en las Figura 7.2 y 7.3.

Justificación: se detalla el conjunto de heurísticas que cumple esa relación.

Responsable: indica la/s persona/s responsable/s de la aplicación de las heurísticas, es decir quienes crearon las componentes destino a partir del origen. Esta información se obtiene de la instancia de la Vista de Configuración de cada componente destino.

Origen: componente que es origen de la relación del trace. Es una instancia de alguna clase de la Figura 7.1. Como se detalla más adelante es el parámetro del *Contexto de Navegación*. A pesar que en las figuras 7.2 y 7.3 se indica que puede haber más de un origen para la relación de trace, en la Vista de Trace se considera la cardinalidad del origen como 1, porque siempre se da como entrada una sola instancia a partir de la cual se determinará la relación de trace.

Destino: conjunto de instancias de alguna clase de la Figura 7.1 obtenidas de la relación de trace.

Nota: para el caso del modelo de objetos, el **destino** siempre devuelve una instancia de la clase o CRC que interviene en la relación de trace independientemente que el destino sea una clase o una componente. De esta forma se obtiene toda la información de la relación, en particular las heurísticas que justificaron la relación (como se indica en las figuras 7.2 y 7.3). Por ejemplo si existe una relación de trace entre un impacto y una responsabilidad, se devuelve la *CRC* que contiene a la responsabilidad, remarcando esta última.

CardinalidadDestino: representa la cantidad de modelos o componentes que se han creado como destino de la aplicación de las heurísticas indicadas en esa relación de trace. La obligatoriedad de la cardinalidad indica que es una derivación automática(solo para la heurística HC1).

Como se mencionó anteriormente, las instancias de Vistas de Trace son creadas por las relaciones de trace indicadas en la Figura 7.1, Expresamos estas relaciones utilizando el lenguaje de definición de los contextos navegacionales de OOHDM [Rossi'96]. Un *Contexto de navegación* es un conjunto de nodos relacionados con algún criterio (en este caso relaciones de trace) que pueden ser navegados de una determinada

forma. En este caso las clases de la figura 7.1. son las clases que contienen la información y que serán navegadas a través de las Vistas de Trace. Los contextos navegacionales devuelven un conjunto (a veces unitario) de componentes que cumplen con la condición definida en el contexto para el parámetro dado como entrada. Este conjunto es el atributo destino de la Vista de Trace.

7.2.1 Relaciones Forward con respecto al modelo de CRCs:

Se describen y ejemplifican algunos de los contextos navegacionales surgidos de las relaciones de las figuras 7.2. y 7.3. Los ejemplos son instancias de la Vista de la Trace, omitiendo en este caso el responsable de la aplicación de las heurísticas, tal como está indicado en la figura 7.4 . Cada contexto se precede con la relación de trace de la Figura 7.1 que le dio origen y las variables utilizadas son instancias de las clases de la misma Figura, como se indica a continuación:

t es una instancia de la clase `LEL`.
crc es una instancia de la clase `CRC`.
i es una instancia de la clase `Impacto`
e es una instancia de la clase `Escenario`
rf es una instancia de la clase `RF`
r es una instancia de la clase `Responsabilidad`
n es una instancia de la clase `Noción`
rnfc es una instancia de la clase `RCal`
rnfm es una instancia de la clase `RMac`
at es una instancia de la clase `Atributo`

El formato de un contexto de navegación es:

$Trace_n$: Contexto $_{nombre}$(parámetro Entrada) = {tipo parámetro Salida / sentencia condicional}

Donde

El $Trace_n$ y el nombre indican la relación de trace de la Figura 7.1 y su semántica expresada en las figuras 7.2 o 7.3.

El parámetro de entrada se indica con alguna de las variables expresadas anteriormente y determina la instancia que genera la relación de trace.

El tipo parámetro de Salida indica que tipo es la/las instancias devueltas, indicado con alguna de las variables definidas.

La sentencia condicional representa las diferentes condiciones que deben cumplir las instancias resultado con respecto al parámetro de entrada, las cuales se basan en las Relaciones y clases de la Figura 7.1.

- Trace1: Contexto $_{seModeladaComo}$ (t) = {crc / t.clasificación= Sujeto AND seModelaComo (t,crc)}

Este Contexto retorna todas las instancias de las CRCs correspondiente a las clases primarias. La figura 7.5 muestra dos instancias de la Vista de Trace relacionadas a este Contexto para el caso de Estudio del "Círculo de Ahorro" (capitulo 8). La primer instancia evalúa el contexto para el término del LEL t = Adherente devolviendo la CRC que él originó a través de la heurística a **HC1**. En este caso la cardinalidad de la relación es 1 en el destino y el origen, ya que se necesitó un solo término de LEL para generar una CRC. La misma situación ocurre para la segunda instancia del ejemplo, que muestra la relación entre el término t = Administradora y su CRC correspondiente.

Trace: Nombre: seModelaComo Justificación: cumple heurística HC1 Origen: **adherente ε LEL** Destino: **Adherente ε CRC** CardinalidadDestino: 1	***Trace:*** Nombre: seModelaComo Justificación: cumple heurística HC1 Origen: **Administradora ε LEL** Destino: **Administradora ε CRC** CardinalidadDestino: 1

Fig. 7.5: instancias de la Vista de Traceability creadas por la heurística HC1

- Trace1: Contexto $_{\text{puedeModelarseComo}}$ (t) = {crc / t.clasificación <> Sujeto AND puedeModelarseComo(t,crc)}

Este contexto devuelve las clases secundarias generadas a partir de los términos del LEL que no son Sujetos, siguiendo la heurística **HC3**. La figura 7.6 muestra un ejemplo de dos instancias de la Vista de Traceability obtenidas a partir de este Contexto. La primer instancia muestra la relación entre un Objeto del LEL t = Bien Tipo con la crc = Bien Tipo correspondiente a la clase secundaria que generó. La segunda instancia relaciona el término t = Licitación (Frase Verbal en el LEL) con la CRC que generó.

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HC3 (HC3.1) Origen: **Bien Tipo ε LEL** Destino: **Bien Tipo ε CRC** CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple heurística HC3 (HC3.3) Origen: **Licitación ε LEL** Destino: **Licitación ε CRC** CardinalidadDestino: 1

Fig. 7.6: instancias de la Vista de Traceability creadas por la heurística HC3

- Trace2: Contexto $_{\text{puedeModelarseComo}}$ (t) = {crc / res ε Responsabilidad, t.clasificación = FraseVerbal, parte-de (crc, res) AND puedeModelarseComo(t,Res)}

Este contexto retorna las CRCs que posean responsabilidades que son términos en el LEL. La Fig.7.7 muestra dos instancias generadas por la heurística **HC2**. La primer instancia es el resultado de aplicar el contexto para el término del LEL t = ExpulsarAdherente que se convirtió en responsabilidad de la CRC Administradora aplicando la heurística **HC2**. Como se explicó en una nota al comienzo de esta sección, la vista de Trace devuelve la CRC que contiene a la responsabilidad ya que si se devolviese sólo la responsabilidad se pierde contexto para interpretar la relación. La segunda instancia corresponde a la aplicación del contexto para t = Rechazar Bien Tipo relacionado a crc = Adherente, resaltando la responsabilidad rechazarBientipo generada por el término.

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HC2 Origen: **Expulsar Adherente ε LEL** Destino: **Administradora ε CRC** (responsabilidad: **expulsarAdherente**) CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple heurística HC2 Origen: **Rechazar Bien Tipo ε LEL** Destino: **Adherente ε CRC** (resp: **rechazarBientipo)** CardinalidadDestino: 1

Fig. 7.7: instancias de la Vista de Traceability creadas por la heurística HC2

- Trace 3: Contexto $_{puedeModelarseComo}$ (i) = { crc/ ∋ t ε LEL, parte-de (t, i) , res ε Responsabilidad, puedeModelarseComo(i, Res) AND parte-de (crc, res)}

Este contexto relaciona los impactos con las responsabilidades de las clases primarias y secundarias que originaron. La Figura 7.8. muestra dos instancias, la primera relaciona al impacto i = La Administradora lo entrega a los Adherentes, perteneciente al término del LEL Cupón de Pago con la responsabilidad enviarCupónPago de la CRC Administradora. En este caso, el impacto del término no se convirtió en una responsabilidad para su propia clase sino que crea una responsabilidad para otra CRC, como se explica en la heurística **HC4.1** para términos que corresponde a objetos. La segunda instancia, en cambio, muestra el impacto "cobrar la cuota mensual" del término Banco que genera una responsabilidad en la CRC Banco, según se indica en la heurística **HC2**.

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HC4.1 Origen: **impacto(La Administradora lo entrega a los Adherentes) ε Cupón de Pago ε LEL** Destino: **Administradora ε CRC** (responsabilidad: **enviarCupónPago**) CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple heurística HC2 Origen: **impacto(cobrar la cuota mensual) ε Banco ε LEL** Destino: **Banco ε CRC** (resp: **cobrarCuotaMensual)** CardinalidadDestino: 1

Fig. 7.8: instancias de la Vista de Traceability creadas por las heurísticas HC4 y HC2

- Trace 4: Contexto $_{puedeModelarseComo}$ (i) = {crc/ ∋ t ε LEL AND parte-de (t, i) , crc ε CRCs AND puedeModelarseComo(i, crc)}

Este contexto devuelve aquellas CRCs que corresponden a impactos en algún término del LEL (los cuales son generalmente términos del LEL). La Fig. 7.9 muestra un ejemplo de la aplicación de este contexto. La primer instancia relaciona al impacto i = realizar sorteo del término del LEL Administradora con la CRC Sorteo. En este caso Sorteo es también un término del LEL, por lo que se pudo obtener por la heurística **HC3**. Lo mismo ocurre con la segunda instancia que relaciona al impacto evaluar Licitaciones de la Administradora con la CRC Licitación.

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HC3 (HC3.1) Origen: **impacto (realizar Sorteo) ε Administradora ε LEL** Destino: **Sorteo ε CRC** CardinalidadDestino: 1	***Trace:*** Nombre: seModelaComo Justificación: cumple heurística HC3 (HC3.3) Origen: **impacto (evaluar Licitaciones) ε Administradora ε LEL** Destino: **Licitación ε CRC** CardinalidadDestino: 1

Fig. 7.9: instancias de la Vista de Traceability creadas por la heurística HC3

- Trace 5: Contexto $_{ApareceEnContexto}$ (e) = { crc/ crc ε CRC, res ε Responsabilidad, parte-de (crc, res) AND res contexto includes(e)}

donde el método contexto de una responsabilidad analiza si la responsabilidad o su refinamiento contiene al escenario.

Dado un escenario, este contexto devuelve todas las CRCs que tienen responsabilidades que aparecen en este escenario. La Fig. 7.10 muestra un ejemplo, en este caso la cardinalidad destino es N. Dado un escenario e = Pagar Cuota Mensual Vencida

el contexto devuelve un conjunto de CRCs, esto se refleja en una instancia de Vista de Trace como se indica en la figura 7.10 para destino = {Adherente, Administradora .....} ya que tienen responsabilidades que aparece en dicho escenario.

Trace: Nombre: apareceEnContexto   Justificación: cumple heurística HC5   Origen: **Pagar Cuota Mensual Vencida ε Escenarios**   Destino: { **Adherente ε CRC** (**resp**: **pagar Cuota Mensual Vencida**)   **Administradora ε CRC (resp: cobrar cuota mensual vencida)**   ..... }   CardinalidadDestino: N

Fig. 7.10 instancias de la Vista de Traceability creadas por la heurística HC5

- Trace 6: Contexto $_{puedeModelarseComo}$ (rn) = { crc/ crc ε CRCs AND puedeModelarseComo(rn,crc)}

Este contexto devuelve las CRCs creadas a partir de una regla funcional. La Fig. 7.11 muestra un ejemplo de aplicación de este contexto. La primer instancia relaciona a la regla "La Administradora debe calcular el haber de los adherentes con la clase Administración de Pagos", que generó. Aunque en la Vista de Trace no se expresa cardinalidad de origen, como se indica en la Figura 7.2, la cardinalidad de origen es N ya que en general son varias las reglas que determinan la creación de una nueva clase, como se muestra en la segunda instancia donde se relaciona a otra regla "El incumplimiento en el pago de la cuota mensual de un adherente provoca la liquidación de intereses punitorios" con la misma clase. Este contexto se complementa con el siguiente, que muestra cómo se modelaron las reglas en responsabilidades.

Trace: Nombre: puedeModelarseComo Justificación: cumple sub-heurística HC9.1 Origen: **La Administradora debe calcular el haber de los adherentes ε RF** Destino: **AdministraciónDePagos ε CRC** CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple sub-heurística HC9.1 Origen: **El incumplimiento en el pago de la cuota mensual de un adherente provoca la liquidación de intereses punitorios ε RF** Destino: **AdministraciónDePagos ε CRC** CardinalidadDestino: 1

Fig. 7.11 instancias de la Vista de Traceability creadas por la sub-heurística HC9.1

- Trace 7: Contexto $_{puedeModelarseComo}$ (rf) = { crc/ crc ε CRCs, res ε Responsabilidad, puedeModelarseComo(rf,res) and parte-de (crc, res)}

Este contexto devuelve todas las responsabilidades que han sido creadas a partir de una Regla Funcional. La Fig. 7.12 muestra un ejemplo del contexto, la primer instancia relaciona la regla "Si la Asamblea de adherentes decide continuar con el plan por un bien análogo, se deberá realizar un nuevo contrato" con la crc = Administradora ya que la regla generó a la responsabilidad "realizarNuevoContrato" de esa CRC. La segunda instancia muestra una relación para la regla "El incumplimiento en el pago de la cuota mensual de un adherente provoca la liquidación de intereses punitorios", con la crc = AdministracióndePagos ya que la regla generó la responsabilidad "determinarIncumplimiento" de esa clase.

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HC9 Origen: **Si la Asamblea de adherentes decide continuar con el plan por un bien análogo, se deberá realizar un nuevo contrato. ε Reglas Funcionales** Destino: **Administradora ε CRC (resp: realizarNuevoContrato)** CardinalidadDestino: 1	***Trace:*** Nombre : puedeModelarseComo Justificación: cumple heurística HC9 Origen: **El incumplimiento en el pago de la cuota mensual de un adherente provoca la liquidación de intereses punitorios. ε Reglas Funcionales** Destino: **Administración de Pagos ε CRC (resp: determinarIncumplimiento)** CardinalidadDestino: 1

Fig. 7.12 instancias de la Vista de Traceability creadas por la heurística HC9

7.2.2 Relaciones Forward con respecto al modelo lógico

- Trace 8: Contexto $_{\text{seTransforma}}$ (crc) = {cl / cl ε Clase AND seTransforma a (crc, cl)}

Este contexto devuelve instancias para todas las clases generadas a partir de las CRCs. La figura 7.13 muestra instancias de la Vista de Trace creadas a partir de este contexto, la primera relaciona a la CRC Adherente con su clase Adherente en el modelo lógico, mientras que la segunda instancia hace lo mismo para la CRC Banco.

Trace: Nombre: seTransforma Justificación: cumple heurística HM1 Origen: **adherente ε CRC** Cardinalidad Origen: 1 Destino: **Adherente ε Clase** CardinalidadDestino: 1	***Trace:*** Nombre: seTransforma Justificación: cumple heurística HM1 Origen: **Banco ε CRC** Cardinalidad Origen: 1 Destino: **Banco ε Clase** CardinalidadDestino: 1

Fig. 7.13 instancias de la Vista de Traceability creadas por la heurística HM1

- Trace 9: Contexto $_{\text{puedeModelarseComo}}$ (t) = {cl / t.clasificación = Objeto, cl ε Clase, at ε Atributo, parte-de (cl, at) AND puedeModelarseComo (t, at) }

Este contexto devuelve instancias con todas las clases que posean atributos que sean términos del LEL. La figura 7.14 muestra ejemplos de Vista de Trace obtenidas a partir de dicho contexto. La primer instancia muestra el atributo valorCuotaMensual de la clase Plan de Pago que se generó a partir del término del LEL t = cuota mensual aplicando la heurística **HM1.2.** La segunda instancia muestra como el término t = cuota de ingreso se transforma en un atributo de la clase Administradora

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HM1.2 Origen**: cuota mensual ε LEL** Destino: **Plan de Pago ε Clase (atributo: valorCuotaMensual)** CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple heurística HM1.2 Origen: **cuota de ingreso ε LEL** Destino: **Administradora ε Clase (atributo: MontoDerecho Admisión)** CardinalidadDestino: 1

Fig. 7.14 instancias de la Vista de Traceability creadas por la sub-heurística HM1.2

- Trace 10: Contexto $_{\text{puedeModelarseComo}}$ (r) = {as/ crc ∋ CRC, parte-de(crc,r) , as ε Asociación / puedeModelarseComo(r,as)}

Como se observa en las instancias de vistas de Trace obtenidas por este contexto en la Figura 7.14, este contexto devuelve el origen de las relaciones de asociación, que corresponden a colaboraciones de las CRCs. La primer instancia muestra el origen de la asociación "paga cuotas" entre las clases Adherente y Banco surgida por la aplicación de la heurística **HM3.3** analizando la colaboración de las CRCs Banco y Adherente para la responsabilidad del Adherente " pagar cuota mensual".

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HM3.3 Origen**: paga cuota mensual en Banco ε CRC (Adherente)** Destino: **asociación(pagaCuotas) (Adherente/ Banco)** CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple heurística HM3.3 Origen: **asegurar Adherente ε CRC(Aseguradora)** Destino: **asociación** (**realiza) (Aseguradora / SeguroVida)** CardinalidadDestino: 1

Fig. 7.15 instancias de la Vista de Traceability creadas por la heurística HM3

- Trace 11 $_{puedeModelarseComo}$: Contexto (n) = {as/ ə t ε LEL AND parte-de (t, n) , as ε Asociación / puedeModelarseComo(n , as)}

La Fig. 7.16 muestra ejemplos de instancias de Trace obtenidas por la aplicación de este contexto, devolviendo las asociaciones de relación parte-de y "es-un" entre dos clases del modelo lógico a partir de las nociones. La primer instancia muestra el origen de la relación "es-un" entre las clases Solicitante y Adherente surgida por la noción "solicitante aceptado por la Administradora" del término Adherente.

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HM3.3 Origen**: solicitante aceptado por la Administradora ε noción de Adherente** Destino: **es-un (Solicitante / Adherente)** CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple heurística HM3.3 Origen: **conjunto de adherentes que desean .... ε noción de Grupo** Destino: **parte –de (Grupo / Adherente)** CardinalidadDestino: 1

Fig. 7.16 instancias de la Vista de Traceability creadas por la heurística HM3

- Trace 12: Contexto $_{origina}$ (n) = { cl / ə t ε LEL AND parte-de (t, n), cl ε Clase, at ε Atributo, parte-de (cl,at) AND origina(n,at) }

La Fig. 7.17 muestra un ejemplo de la aplicación de este contexto, devolviendo instancias de Trace cuyo atributo destino son las clases que tiene atributos que se originaron en nociones del término del LEL correspondiente a la clase. La primer instancia muestra el origen del atributo "planes de Ahorro" de la clase Administradora a partir de la noción Administradora "....empresa encargada de planes de ahorro .....". La segunda instancia relaciona la noción de Adherente n = integrante de un grupo con el atributo que generó llamado "#grupo" de la clase Adherente. En ambos casos la cardinalidad de las relaciones es 1.

Trace: Nombre: origina Justificación: cumple heurística HM1 Origen**: "....empresa encargada de planes de ahorro ....." ε noción de Administradora** Destino: **Administradora (atributo: planes de Ahorro)** CardinalidadDestino: 1	***Trace:*** Nombre: origina Justificación: cumple heurística HM1 Origen: integrante de un grupo **ε noción de Adherente** Destino: **Adherente (atributo: #grupo)** CardinalidadDestino: 1

Fig. 7.17 instancias de la Vista de Traceability creadas por la heurística HM1

- Rel13: Contexto $_{\text{afectaMétodos}}$ (rnf) = {cl/ ɜ m ε Método, parte-de(cl,mt) and afecta(rnf,mt) }
- Rel13: Contexto $_{\text{creaMétodo}}$ (rnfm) = { cl/ ɜ mt ε Método, parte-de(cl,mt)AND crea (rnfm,mt)}

Este contexto devuelve las clases cuyos métodos son creados o afectados por las Reglas no Funcionales. La Fig. 7.18 muestra un ejemplo de la aplicación de este contexto, en este caso devuelve los métodos que han sido afectados a partir de una Regla No funcional. La primer instancia muestra la Regla no funcional "Si el adjudicatario rechaza por tres veces el bien tipo la Administradora puede expulsarlo del Grupo" que afecta a los métodos "rechazarBienTipo" de la clase Adherente y " analizarRechazoBienTipo" de la clase Administradora, siendo la cardinalidad destino N ya que afecta a varios métodos de diferentes clases. En el segundo caso, la cardinalidad es 1 ya que la regla afecta a un solo método de la clase Adherente.

Trace: Nombre: afecta Justificación: cumple heurística HM5.3 Origen**: Si el adjudicatario rechaza por tres veces el bien tipo la Administradora puede expulsarlo del Grupo"** ε Regla Destino: **Adherente (método: rechazarBienTipo) Administradora(método: analizaRechazoBienTipo)** CardinalidadDestino: N	***Trace:*** Nombre: afecta Justificación: cumple heurística HM5.3 Origen**: La Administradora puede aceptar o denegar la solicitud de cambio del bien dentro de los diez días de realizada. ε Regla** Destino: **Adherente (método: AceptarSolicitudAdhesión)** CardinalidadDestino: 1

Fig. 7.18 instancias de la Vista de Traceability creadas por la Sub-heurística HC5.3

- Trace 14: Contexto $_{\text{afectaAtributos}}$ (rnf) = {cl/ ɜ at ε Atributo, parte-de(cl, at) and afecta(rnf,at) }
- Trace 14: Contexto $_{\text{creaAtributo}}$ (rnfm) = { cl/ ɜ at ε Atributo, parte-de(cl,at) AND crea (rnfm,at)}

Este contexto devuelve las clases con atributos creados o afectados por Reglas no Funcionales. La Fig. 7.19 muestra dos ejemplos de Vista de Trace creados a partir de este Contexto. La primer instancia muestra el atributo de la clase Adherente "cantidadRechazo " afectado por la Regla "Un adjudicatario puede rechazar hasta tres veces el bien tipo". En este caso define una restricción sobre el atributo (Adherente.cantidadRechazo<=Administradora.cantidadRechazosPermitidos.) La segunda instancia muestra la creación del atributo de la clase Administradora (cantidaRechazosPermitidos) a partir de la Regla "Un adjudicatario puede rechazar hasta tres veces el bien tipo". En ambos casos la cardinalidad del origen y el destino es 1.

Trace: Nombre: afecta Justificación: cumple HM5.1 Origen**: Un adjudicatario puede rechazar hasta**	***Trace:*** Nombre: crea Justificación: cumple HM5.2 Origen**: Un adjudicatario puede rechazar hasta**

tres veces el bien tipo" ε Rnfmac Cardinalidad Origen: 1 Destino: **Adherente (atributo: cantidadRechazo <= Administradora.cantidadRechazosPermitidos** CardinalidadDestino: 1	**tres veces el bien tipo" ε Rnfmac** Cardinalidad Origen: 1 Destino: **Administradora (atributo: cantidaRechazosPermitidos=3)** CardinalidadDestino: 1

Fig. 7.19 instancias de la Vista de Traceability creadas por la sub-heurística HM5.2

- Trace 15: Contexto $_{\text{afectaAsociación}}$ (rnfm) = { as/ as ε Asociación) AND afecta (rnfm,as)}

Este contexto devuelve instancias de Asociaciones afectadas por reglas no funcionales del macrosistema; sin embargo son más útiles los contextos anteriores, porque indican concretamente cómo se afectaron los atributos y métodos en base a las reglas. La figura 7.20 muestra una instancia de la Vista de Trace que relaciona a la asociación parte-de entre el Grupo y el planDeAhorro) que es afectada por la Regla "La cantidad de miembros de un grupo es el doble de la cantidad de cuotas del plan elegido".

Trace: Nombre: afectaAsociación Justificación: cumple HM5.1 Origen**: La cantidad de miembros de un <u>grupo</u> es el doble de la <u>cantidad de cuotas del plan</u> elegido ε Regla No funcional del macrosistema** Destino: **parte-de (Grupo/ planDeAhorro)** CardinalidadDestino: 1

Fig. 7.20 instancias de la Vista de Traceability creadas por la sub-heurística HM5.1

7.2.3 Relaciones Backward

Todas las relaciones de trace presentadas anteriormente tienen su relación inversa para permitir la *pre-traceability*, es decir, dado una componente del modelo de objetos se obtiene el origen que justifica su existencia. Algunos de los Contextos de Navegación de relaciones Backward son:

- Trace 3: Contexto $_{\text{seOriginoEn}}$ (r) = {t / t ε LEL, i ε Impacto, parte-de (t,i) AND PuedeModelarseComo (i,r) }

La figura 7.21 muestra dos instancias de la Vista de Trace correspondiente a este contexto. La primera justifica la aparición de la responsabilidad r = pagar cuota en Adherente por la aplicación de la heurística **HC2** ya que la responsabilidad fue creada a partir de un impacto del término del LEL Adherente. La otra instancia justifica la modelización de la responsabilidad r = cambiarBienTipo de la clase Plan de Ahorro que fue modelada a partir del impacto **"**cambiar el Bien Tipo**"** del término del LEL "Plan de Ahorro" por la heurística **HC4.1**.

Trace: Nombre: puedeModelarseComo Justificación: cumple heurística HC2 Origen: **pagar Cuota Mensual ε CRC Adherente** Destino: **adherente ε LEL (impacto: pagar Cuota Mensual**) CardinalidadDestino: 1	***Trace:*** Nombre: puedeModelarseComo Justificación: cumple heurística HC4.1 Origen: **cambiarBienTipo ε CRC Plan de Ahorro** Destino: **Plan de Ahorro ε LEL (impacto: cambiar el Bien Tipo**) CardinalidadDestino: 1

Fig. 7.21 instancias de la Vista de Traceability creadas por las heurísticas HC2 y HC4

- Trace 1: Contexto $_{\text{seOriginoEn}}$ (crc) = {t / t ε LEL, t.clasificación = sujeto, crc ε CRC AND modeladaComo (t,crc) }

La Fig. 7.22 muestra instancias de la Vista de Trace que justifican la modelización de las CRCs correspondientes a las clases primarias por las heurística **HC1**, en particular para las CRCs Adherente y Administradora

Trace: Nombre: seModelaComo Justificación: cumple heurística HC1 Origen: **adherente ε CRC** Destino: **Adherente ε LEL** CardinalidadDestino: 1	***Trace:*** Nombre: seModelaComo Justificación: cumple heurística HC1 Origen: **Administradora ε CRC** Destino: **Administradora ε LEL** CardinalidadDestino: 1

Fig. 7.22 instancias de la Vista de Traceability creadas por la heurística HC1

- Trace 14: Contexto $_{seOriginoEn}$ (at) = {rnfm/ cl ε Clase , parte-de(cl,at) AND afecta(rnf,at) }

Este contexto devuelve las Reglas No Funcionales que afectan a los atributos de las clases. La Fig. 7.23 muestra un ejemplo de la aplicación de este contexto creando una instancia de la vista de Trace que relaciona al atributo cantidaRechazosPermitidos de la Administradora con la Reglas no Funcional "Un adjudicatario puede rechazar hasta tres veces el bien tipo" que lo afectó.

Trace: Nombre: crea Justificación: cumple HM5.2 Origen**: Administradora (atributo: cantidadRechazosPermitidos= 3)** Cardinalidad Origen: 1 Destino: **"Un <u>adjudicatario</u> puede rechazar hasta tres veces el bien tipo" ε Regla** CardinalidadDestino: 1

Fig. 7.23 instancias de la Vista de Traceability creadas por la heurística HC9

7.3 Soporte automatizado de traceability

Como puede observarse a partir de los contextos navegacionales, las relaciones de trace que se generan a partir de la aplicación de las heurísticas constituyen un gran volumen de información y con mucho nivel de detalle (pensar que se obtiene información de trace hasta el nivel de un atributo) En general, debido al gran volumen de información que se genera durante el proceso de desarrollo de software, es costoso, en términos de tiempo y recursos, almacenarla en su totalidad [Kotonya'98]. Por esta razón es necesario definir un modelo de traceability que estructure la información de forma tal de facilitar su acceso a través de mecanismos de búsqueda y que sea adaptable a las necesidades particulares de cada proyecto y organización [Domges'98; Ramesh'98].

Si bien la estrategia presentada en este capítulo define todas las relaciones de trace generadas por las heurísticas en base a Contextos Navegacionales, una herramienta que la implemente necesariamente debe permitir que el Administrador de un Proyecto seleccione que tipo de trace necesita, ya que al no ser las heurísticas automáticas, los Ingenieros de Aplicación son los responsables de almacenar la información, generando un volumen imposible de administrar, como se mencionó anteriormente. En este sentido, El meta-modelo *MenDor* permite al ingeniero definir los contextos navegacionales entre los modelos que va a utilizar, de esta forma se decide cuales son las relaciones que se quiere rastrear. Esto se hace a partir de la definición e implementación de una clase Contexto Navegacional que puede ser parametrizada por el ingeniero en las diferentes

instanciaciones del Meta-modelo (para mayor detalle ver [Petersen'99]). La instanciación actual, HeAR, permite navegar entre los escenarios y LEL, ya sea por términos del LEL, relaciones de escenarios/sub-escenarios, y realizando inspecciones [Doorn'98] que también son resueltas con contextos navegacionales. Por otro lado, *BaselineMentor* permite la traceability a partir de contextos navegacionales a nivel de escenarios, LEL y CRCs pero con un nivel de granularidad mayor que el presentado en este capítulo, es decir, permite la navegación desde / hacia un término del LEL o un escenario completo (no sus componentes) y una CRC completa [Antonelli'00]. Las siguientes figuras muestran algunas pantallas de *BaselineMentor* correspondiente a los aspectos de navegación. La figura 7.24 muestra los menús del LEL y los escenarios que permiten navegar entre los modelos; dado un símbolo del LEL, devolver todos los escenarios, y dado une escenario devolver todos los símbolos del LEL que aparecen en él.

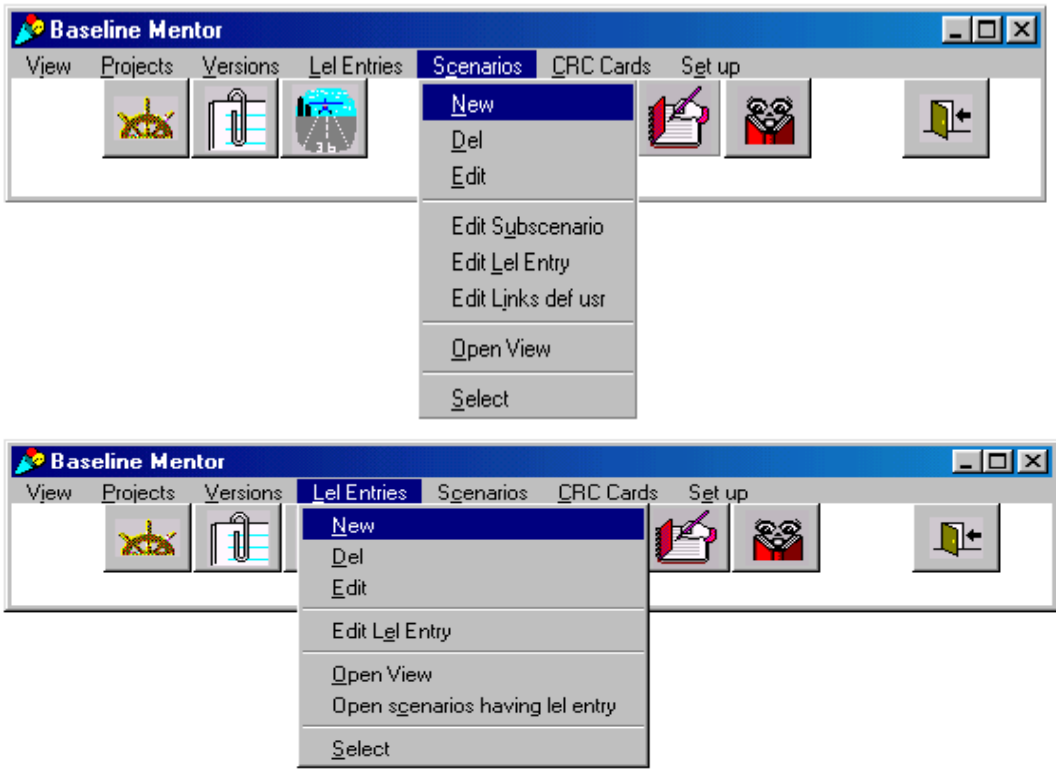


Fig. 7.24 Menús de los escenarios y LEL del *BaselineMentor* [Antonelli'99].

La figura 7.25 muestra una parte del escenario "Schedule the Meeting" correspondiente al caso de estudio del "Meeting Scheduler"(capítulo 8, sección 8.2) que se visualiza como resultado de aplicar al símbolo Agenda del LEL la función del menú "open scenarios having LEL entry". Asimismo en el escenario se visualiza otro escenario (Move the Meeting Date) que lo referencia y que puede ser accedido desde él y diferentes términos del LEL que aparecen en él de forma subrayada y que también pueden ser navegados a partir de esta ventana. La Figura 7.26 muestra una CRC correspondiente a un Requester. En ella aparecen subrayados y en diferente color términos del LEL en las responsabilidades, escenarios y otras CRCs que colaboran con la clase Requester y que pueden ser accedidas desde la misma.

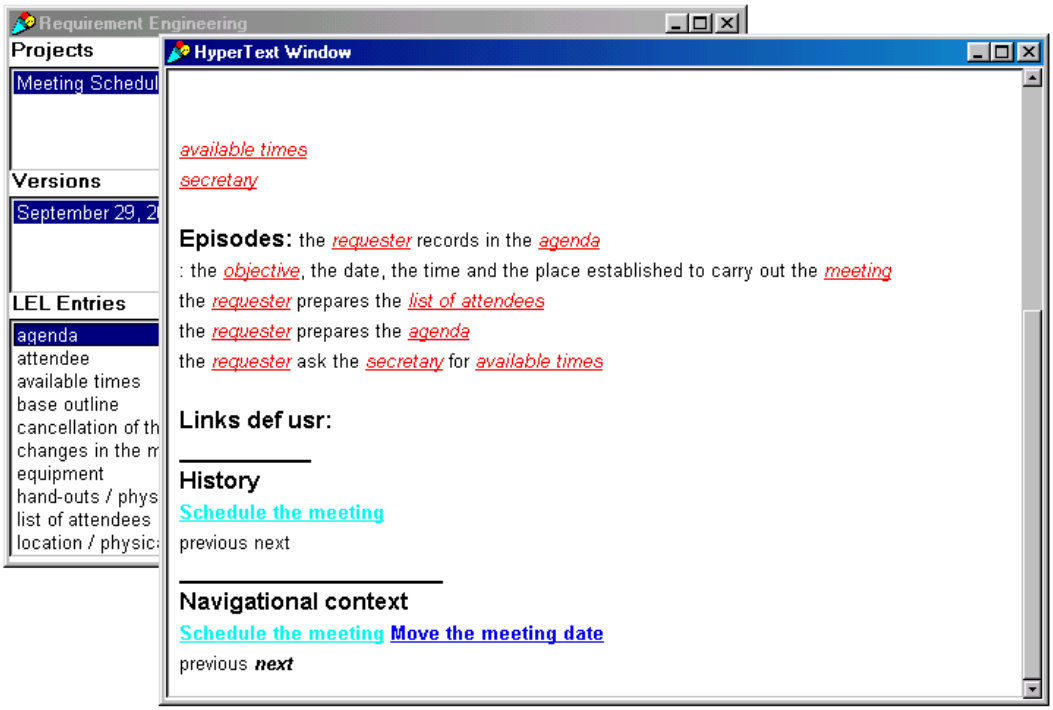


Fig. 7.25: símbolo del escenario "Schedule the Meeting" y componentes accesibles desde él.

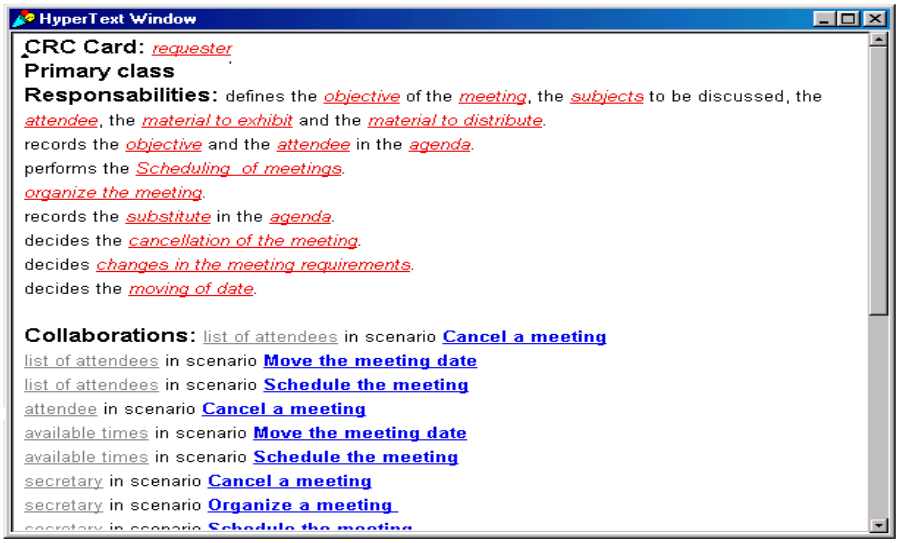


Fig. 7 .26: CRC Requester y componentes accesibles desde ella.

Capítulo 8:
Aplicaciones de la Estrategia

En este capítulo se presenta la aplicación de la estrategia en diversos casos de estudio. En la primer sección se desarrolla un caso de estudio completo. Luego se mencionan diversos casos de estudios realizados y las conclusiones obtenidas a partir de los mismos.

8.1 Desarrollo de un caso de estudio: sistema de Plan de Ahorro para la adquisición de una automóvil

Este trabajo se realizó durante un curso de especialización dirigido por el Dr. Leite en la Universidad de Tandil (1997). Se obtuvieron 9 conjuntos de escenarios LEL y modelos de objetos. El análisis estuvo basado en documentación real de una Agencia de autos que realiza círculos cerrados de compra sorteo y licitación. El resultado de aplicar parte de la estrategia aquí definida se presenta en [Rivero'98]. En el Anexo 1 se detalla el modelo de escenarios y de LEL producido por uno de los grupos así como también se presenta el Documento que refiere al contrato standard de una Agencia de autos que trabaja con este tipo de planes de ahorro. Este contrato fue considerado como la fuente de Información utilizada para la definición de los modelos de LEL y escenarios. En esta sección se detalla todo el desarrollo del modelo de objetos siguiendo la estrategia presentada en esta tesis. También se define el modelo de Reglas, ya que el mismo no fue desarrollado en el marco del curso mencionado.

8.1.1. Resumen del Caso

Se desea modelar el funcionamiento de una administradora de planes de ahorro para automóviles. El objetivo de la administradora es la conformación de grupos de adherentes, los cuales realizan compra de automóviles a través de un sistema de ahorro y licitación. Para ser adherente de un grupo el solicitante deberá completar un formulario y presentar documentación. Si es aceptado se suscribirá a un plan de acuerdo a un contrato de adhesión. Existen diferentes tipos de planes. Los adherentes pagan una cuota mensual, lo que le da derecho a participar en actos mensuales de adjudicación por sorteo o licitación. El sorteo es para todos los adherentes con las cuotas al día, mientras que la licitación es para aquellos que lo han solicitado mediante una oferta de dinero (utilizando un sobre de licitación). Un adherente se convierte en adjudicatario al obtener el automóvil por alguno de estos dos medios; en este caso deberá solicitar el bien mediante un formulario especial y abonar cargos especiales en concepto de la entrega del automóvil y continuar con el pago de las cuotas mensuales. El adjudicatario puede optar por un automóvil de mayor o menor valor, para lo que existe una serie de reglas.

Un adherente puede ceder sus derechos del plan a un tercero. Un adherente puede renunciar al plan o ser separado por la Administradora. El fabricante puede dejar de producir la marca y/o modelo objeto del contrato, lo que afecta de diferente forma a los adherentes u adjudicatarios. La adhesión al grupo implica el pago de un seguro de vida y una vez que el auto ha sido adjudicado, se le exige un seguro para el mismo. Un grupo

puede ser disuelto por decisión de los adherentes (en caso de que no se fabrique un reemplazo del bien) o por la Administradora en caso de falta de miembros en un grupo.

8.1.2 Definición de un modelo de Reglas

Siguiendo las heurísticas propuestas en el Capítulo 3, se construyó y documentó un modelo de Reglas. Utilizando como fuente de información el documento presentado en el Anexo, se seleccionaron las sentencias que denotaban límites, derechos y responsabilidades de las principales entidades del documento, modeladas en el LEL. A continuación se presenta el documento final, clasificado según la taxonomía. Cada regla presenta un indicador de su semántica dentro del macrosistema: Límites (L), Derechos (D), Responsabilidades (R). Asimismo se indican subrayados los términos del LEL.

Reglas Funcionales

1. La Administradora entrega uno o más Bienes Tipos por sorteo y licitación (R)
2. La Administradora acepta solicitantes para sus planes de ahorro (R)
3. La Administradora debe ofrecer distintos planes de ahorro (R)
4. La Administradora debe calcular el haber de los adherentes. (R)
5. La Administradora debe convocar a Asamblea de adherentes ante la discontinuación del bien. (R)
6. Si el adjudicatario no cumpliera con los requisitos de adjudicación, la Administradora dejará la adjudicación sin efecto. (R)
7. Si existe incumplimiento imputable al grupo, la Administradora procede a la liquidación del grupo. (R)
8. Si no existen adherentes en el grupo en condiciones de ser adjudicatarios, la Administradora debe proceder a la liquidación del grupo. (R)
9. Si el bien tipo queda sin adjudicar en una licitación, la Administradora lo agrega al siguiente acto de adjudicación. (R)
10. El incumplimiento en el pago de la cuota mensual de un adherente provoca la liquidación de intereses punitorios. (R)
11. El adherente debe pagar una cuota mensual. (R)
12. El adherente puede elegir entre una lista de Bancos para pagar la cuota. (L)
13. Si no existe deuda pendiente, el adherente participa del sorteo. (D)
14. Si ocurre una cambio del bien tipo, los adherentes absorberán el importe total de la variación del precio. (L)
15. El solicitante debe completar la solicitud de adhesión con los datos personales del solicitante, el bien tipo y el plan elegido(R)
16. El solicitante debe pagar el derecho de admisión al presentar la solicitud de adhesión. (R)
17. La Asamblea de adherentes decide por mayoría absoluta las acciones a seguir ante la discontinuación del bien. (D)
18. La licitación produce uno o más adjudicatarios por grupo. (R)
19. Si la Asamblea de adherentes decide continuar con el plan por un bien análogo, se deberá realizar un nuevo contrato. (D)
20. Si la Asamblea de adherentes decide no continuar con el plan, La Administradora debe reintegrar el haber a los adherentes no adjudicatarios y realizar la liquidación del grupo. (R)
21. Si la Asamblea de adherentes lo decide en caso de la discontinuación del bien, la Administradora debe proceder a la liquidación del grupo. (R)
22. El fabricante puede informar de nuevos modelos o la discontinuación del bien. (D)
23. La Administradora debe determinar un seguro de vida al aceptar un adherente.
24. El adjudicatario puede cambiar el bien tipo adjudicado (D)
25. El adjudicatario debe pagar una cuota mensual. (R)
26. El adjudicatario debe completar la solicitud de adhesión. (R)

27. El adjudicatario debe contratar un seguro de bien tipo. (R)
28. Si el adjudicatario rechaza por tres veces el bien tipo la Administradora puede expulsarlo del Grupo (D)

Reglas No-funcionales del Macrosistema

29. La Administradora debe notificar la liquidación de un grupo por comunicación fehaciente. (R)
30. La Administradora debe notificar la sustitución del bien por comunicación fehaciente. (R)
31. La Administradora debe realizar el sorteo y licitación el primer día hábil de cada mes.(L)
32. La Administradora devolverá el derecho de admisión si la solicitud de adhesión es rechazada, dentro de los 10 días (R)
33. La Administradora entregará el vehículo dentro de los sesenta días de recibido el formulario de pedido de vehículo. (R)
34. La Administradora puede aceptar o denegar la solicitud de cambio del bien dentro de los diez días de realizada. (D)
35. Si se reciben tantas solicitudes de adhesión como requiere el plan, la Administradora procede a constituir un grupo. (R)
36. Si en un grupo existen fondos para más de un bien tipo, la Administradora adjudica el primero de ellos por sorteo y el resto por licitación. (L)
37. Si en un grupo sólo existen fondos para un bien tipo, la Administradora realiza la adjudicación por sorteo. (L)
38. Si en un mismo grupo existen ofertas de licitación iguales, la Administradora adjudica de acuerdo al orden de sorteo. (D)
39. Un adjudicatario puede rechazar hasta tres veces el bien tipo(L)
40. Si el adherente renuncia al plan, se le devolverá su haber disminuido en un 2% a favor del grupo. (D)
41. El adherente debe pagar la cuota mensual en los primeros cinco días del mes. (L)
42. El adherente debe pagar sus cuotas mediante el cupón de pago. (R)
43. El adherente paga sus cotas en el Banco elegido (L)
44. El adherente paga sus cuotas atrasadas en la Administradora (L)
45. El adherente puede transferir el plan a un tercero. (D)
46. El adherente puede realizar una oferta de licitación por un monto no mayor al total de cuotas comerciales que resten pagar ni inferior a la cuota pura del plan. (D)
47. La transferencia del plan del adherente debe hacerse mediante comunicación fehaciente. (R)
48. Si el adherente se encuentra al día con los pagos, puede renunciar al plan por comunicación fehaciente. (D)
49. El adjudicatario debe elegir la compañía Aseguradora de una lista confeccionada por la Administradora. (R)
50. El adjudicatario debe manifestar su acuerdo por comunicación fehaciente. (R)
51. El adjudicatario debe pagar sus cuotas mediante el cupón de pago. (R)
52. El adjudicatario debe pagar un derecho de adjudicación previo a la entrega del bien. (R)
53. El adjudicatario debe ser notificado por comunicación fehaciente. (D)
54. El adjudicatario puede no aceptar la adjudicación o dejar vencer el plazo para hacerlo hasta tres veces. (D)
55. El adjudicatario puede realizar una cancelación anticipada de su deuda. (D)
56. El adjudicatario puede rechazar la adjudicación por comunicación fehaciente. (D)
57. El adjudicatario puede solicitar el cambio del bien. Si es aceptado el cambio, el adjudicatario puede cambiar el bien tipo adjudicado, pagando la diferencia si el nuevo bien es de mayor valor que el bien adjudicado, ó computándose la diferencia para cancelar las últimas cuotas del plan, en caso de ser un bien de menor valor. (D)
58. Si el adjudicatario no acepta la adjudicación del bien por cuarta vez, la Administradora tiene derecho a dar por resuelta la solicitud de adhesión. (D)

59. Si el adjudicatario no acepta la adjudicación por cuarta vez y se trata del último acto de adjudicación, la Administradora puede optar por exigir el cumplimiento de la obligación. (D)
60. Si la entrega del bien no se realiza en el plazo establecido, el adjudicatario puede reclamar el pago de una penalidad. (D)
61. La licitación se debe realizar en presencia de un escribano el 5to. día hábil de cada mes. (R)
62. Los resultados de una licitación deben notificar por comunicación fehaciente al/los adjudicatario/s. (R)
63. Los resultados de una licitación se deben publicar en diarios de circulación nacional.(R)
64. El bien tipo puede ser adjudicado por sorteo o por licitación. (L)
65. El grupo se identifica por un número de grupo. (R)
66. Si se encuentran un 60% o más de las cuotas del grupo vencidas e impagas, se produce incumplimiento imputable al grupo. (R)
67. La cantidad de miembros de un grupo es el doble de la cantidad de cuotas del plan elegido. (L)
68. El plan especifica la duración total en meses, cantidad de adherentes, bien tipo, valor de la integración mínima, valor de las cuotas y porcentajes de derechos y cargas. (R)
69. La comunicación fehaciente debe hacerse por medio de: telegrama colacionado, telegrama con aviso de entrega, nota con aviso de entrega o carta documento. (L)
70. La cuota mensual se calcula como la cuota pura más los gastos de administración. (R)
71. La solicitud de adhesión aceptada tiene un número de orden dentro del grupo. (R)
72. El solicitante debe realizar los pagos correspondientes a la solicitud de adhesión en la Administradora.

8.1.3 Definición de las CRCs

En esta sección, se muestran las CRCs resultantes de aplicar la estrategia mostrada en la Figura 4.1 cuyas entradas son los modelos de LEL, escenarios y Reglas de Negocio definidos previamente. Para simplificar el ejemplo, sólo se muestran las heurísticas que generaron cada componente, sin crear instancias de la Vista de Traceability como las definidas en las figuras 7.5 hasta 7.26 del capítulo anterior.

8.1.3.1 Definición de las clases Primarias

La definición de las clases primarias es automática ya que como se indica en la heurística **HM1** se modela cada término del LEL correspondiente a un Sujeto como una clase primaria. Luego se definen las responsabilidades y colaboraciones siguiendo las heurísticas y finalmente se las documenta con el formato presentado en la Fig. 4.2 del Capítulo 4. A continuación se muestran las CRCs de las clases primarias obtenidas:

nombre: Adherente justificación: HC1		
Responsabilidades	Colaboraciones	Contexto (HC5)
Pagar Cuota Mensual (HC2) entregarCupon (HC4.1)	cupón de Pago Banco, Administradora,	Pagar cuota
Pagar cuota mensual vencida(HCG3)	cupón de Pago Administradora,	Pagar CuotaVencida
Transferir plan (HC2) enviarComFehac(HC5)	Administradora, Comunicación fehaciente	Transferir Plan
Licitar (HC2) crearsobreLicitacion (HC5) enviarSobre (HC5)	sobreLicitación Administradora	Licitar
Rechazar bien tipo (HC2, HC3.2)	Bien tipo, Administradora, Comunicación fehaciente	Rechazar Bien tipo
Renunciar (HC2)	Plan de ahorro, Comunicación fehaciente, Administradora	Dejar Plan
PagarAdjudicación (HC2)	Administradora Adjudicación	Adjudicación
ElegirBanco (HC5)	Administradora Banco	Solicitar Ingreso
RecibirCuponPago(HC5)	Administradora cupones	Solicitar Ingreso

nombre: Adjudicatario *Justificación:* HC1		
Responsabilidades	*Colaboraciones*	*Contexto*(HC5)
Pagar el derecho de adjudicación (HC2) Cambiar el bien tipo adjudicado (HC2) Elegir aseguradora para el bien tipo(HC2) Retirar bien Tipo(HC2)	Administradora Administradora, Bien tipo Administradora, Aseguradora Administradora	Adjudicación Cambiar Bien tipo adjudicado Asegurar Bien tipo Entregar Bien tipo

nombre: Aseguradora Justificación:		
Responsabilidades	*Colaboraciones*	*Contexto* (HC5)
Asegurar adherente (HC2) Asegurar bien tipo (HC2)	Adherente, Plan de Ahorro, SeguroVida Administradora Adjudicatario, Bien tipo, SeguroBienTipo	Asegurar Adherente Asegurar Bien tipo

nombre: Administradora (HCG1) Justificación: (HC1)		
Responsabilidades	*Colaboraciones*	*Contexto* (HC5)
Aceptar la solicitud de adhesión asegurar (HC2)	Planilla de Adhesión, Solicitante SeguroDeVida	Solicitar Ingreso
Devolver el valor de la cuota de ingreso (HC9)	Adherente	Solicitar Ingreso
ExpulsarAdherente (HC2) eliminar AdherenteGrupo (HC5) si falta cantidad DisolverGrupo (HC5.1)	grupo Adherente Comunicación fehaciente,	Expulsar Adherente Dejar Plan
Determinar seguro de vida(HC2)	Bien tipo, Seguro del bien tipo, Adjudicatario	Asegurar Adherente
Entregar bien tipo (HC2)	Adherente	Asegurar Bien tipo Entregar bien Tipo
Adjudicar un bien tipo (HC2) recibirPagoDerechoAdjudic(HC5) RecibirFormulario (HC5)	Planilla de adhesión, Comunicación fehaciente, Plan de ahorro, Adherente	Adjudicación
Armar grupo (HC2) crearGrupo (HC2) agregarAdherentes (HC5) enviarCuponesPago (HC4.1)	Plan de ahorro, Adherentes, Comunicación fehaciente, Grupo	Armar un grupo
Disolver grupo (HC2) EnviarComFehaciente (HC5)	Plan de ahorro, Grupo, Adherentes, Comunicación fehaciente	disolver Grupo
Determinar cambio de bien tipo para un grupo (HC2)	Adherente Bien tipo grupo	Cambiar Bien tipo
Analizar cambio de bien tipo adjudicado (HC2)	Adherente	Cambiar Bien tipo adjudicado
Cobrar cuota mensual vencida (HC2)	Adherente	Pagar Cuota Vencida
EvaluarLicitación (HC2)	licitación Sorteo	
AnalizarRechazoBienTipo analizarCantidadRechazos eliminarAdherente	Adherente Grupo Sorteo	Evaluar Licitación Rechazar bien Tipo
RealizarSorteo (HC2)	Adherente	
Aceptar Transferencia (HC2) crearReemplazo (HC5) eliminarAdherente (HC5)	Comunicación fehaciente grupo solicitante	Sorteo Transferir Plan
CalcularHaber (HC9)		

ConvocarAsamblea (HC9) RealizarNuevoContrato (HC9) LiquidarGrupo (HC9) DevolverCuotas EntregarCuponPago ConcovarLicitación (HC11)	grupo grupo bien tipo grupo Adherente, cupones ComunicaciónFehaciente, adherentes	SolicitarIngreso CambiarBienTipo CambiarBienTipo LiquidarGrupo SolicitarIngreso EvaluarLicitación

Tanto las responsabilidades Evaluar Licitación, Expulsar Adherente y Realizar sorteo de la clase Administradora corresponden a Frases Verbales del LEL. Siguiendo la heurística de **HC2**, se define como responsabilidades de esta clase, pero son candidatas a ser modeladas ellas mismas como clases. Como se ve en la colaboración, se definieron la clase Sorteo y la clase Licitación (detalladas en la definición de las clases secundarias). En cambio la responsabilidad " expulsar adherente" quedó asignada en la Administradora sin crear una nueva clase.

nombre: Banco *Justificación:* (HC1)		
Responsabilidades	*Colaboraciones*	*Contexto*
Cobrar cuota mensual (HC2) crearComprobantePago (HC5.2) sellarCupón (HC5) Devolver comprobantePago (HC5.2)	Adherente cupónDePago Administradora	Pagar cuota Pagar cuota

nombre: Fabricante *Justificación:* (HC1)		
Responsabilidades	*Colaboraciones*	*Contexto*
Entregar bien tipo (HC2) Avisar cambios de precios de un bien tipo (HC2) Avisar discontinuaciónModelo(HC2) InformarNuevosModelos (HC9.1)	Administradora, Bien tipo Administradora Administradora, Administradora	Entregar Bien Tipo Cambiar Bien tipo Cambiar Bien tipo

nombre: Grupo *Justificación*: (HC1)		
Responsabilidades	*Colaboraciones*	*Contexto*
Aceptar el cambio de un bien tipo (HC2) Rechazar el cambio de un bien tipo(HC2) Eliminar adherente(HC2) devolverDatos (HC5.2)	Administradora, Bien tipo Administradora, Bien tipo Adherente Administradora Administradora	Cambiar Bien tipo Cambiar Bien tipo Dejar Plan Expulsar Adherente Disolver grupo

nombre: Solicitante *Justificación*: (HC1)		
Responsabilidades	*Colaboraciones*	*Contexto*
Solicitar planilla de adhesión (HC2) Entregar la planilla de adhesión completa (HC2) PagarCuota (HC5)	Planilla de adhesión, Administradora Planilla de adhesión, Administradora	Solicitar Ingreso Solicitar Ingreso

8.1.3.2 Definición de las Clases Secundarias

Siguiendo con las etapas para la definición de CRCs presentada en el capítulo 4, se modelan las clases secundarias a partir de los términos del LEL correspondientes a objetos, Frases Verbales y Estados cuya importancia en el Caso de Estudio justificó que sean modelados como clases.

nombre: BienTipo (en este caso a pesar de HCG1 decide poner el nombre de BienTipo ya que es más significativo que Bien) *Justificación*: (HC3.1)		
Responsabilidades	*Colaboraciones*	*Contexto*
SeAdjudica (HC5.2) SeEntrega (HC5.2)	Administradora, Adherente Administradora, Adjudicatario, Aseguradora	Adjudicación Entregar bien Tipo

nombre: Comunicación fehaciente Justificación: (HC3.1)		
Responsabilidades	*Colaboraciones*	*Contexto*
NotificarCierreGrupo (HC4.1)	Administradora, Grupo	Disolver un Grupo
NotificarGanadorAdjudicación (HC4.1)	Administradora, Adjudicatario	Evaluar Licitación
Notificar CambioBienTipo(HC4.1)	Administradora, Fabricante, Grupo	Cambiar Bien tipo
Notificar el cambio de bien tipo adjudicado (HC4.1)	Adjudicatario	Cambiar Bien tipo adjudicado
NotificarRenunciaPlan(HC4.1)	Administradora, Fabricante, Grupo Adjudicatario	Dejar Plan
NotificarExpulsiónPlan (HC4.1)	Adherente, Administradora	Expulsar Adherente
NotificarLicitación	Adherente, Administradora	EvaluarLicitación

Esta clase representa la forma de comunicación de la organización para todas las actividades que debe informar. Se crea una clase para representar todos los formularios que se deben enviar, independizándolo de las clases que los utilizan.

nombre: Cupón de Pago Justificación: (HC3.1)		
Responsabilidades	*Colaboraciones*	*Contexto*
esSellado (HC4.1)	Adherente administradora Banco	Pagar cuota Pagar cuota Vencida

Aquí el impacto "es entregado" del término del LEL Cupón de Pago no se

transformó en una responsabilidad de la clase, sólo indica que el cupón es entregado por la Administradora a los Adherentes, con lo cual, como se explica en la heurística **HC4.1**., habrá responsabilidades en la clase Administradora para enviarlo y recibirlo.

nombre: Licitación *Justificación*: (HC3.2)		
Responsabilidades	*Colaboraciones*	*Contexto* (HC5)
DarDatos (HC5.2) Crear un ranking de adherentes (HC4.2) buscarMaximoMonto Si hay mas de un máximo(HC5.1) UsarOrdenSorteo IndicarAdjudicatarios	Administradora Administradora sobresLicitacion Sorteo	EvaluarLicitación Evaluar Licitación

nombre: Sorteo *Justificación*: (HC3.2)		
Responsabilidades	*Colaboraciones*	*Contexto*
Definir Ganador del sorteo (HC3.2) generarOrdenSorteo (HC5) tomarPrimeroVálido (HC5)	Administradora, grupo adherente comunicación fehaciente	Sorteo

Como se mencionó en la definición de la clase Administradora los términos Licitación y Sorteo, Frases Verbales del LEL, quedaron asignados como responsabilidades de dicha clase, ya que aparecían en los impactos de la misma. Sin embargo en esta etapa, al analizarlos como clases candidatas, se decide modelarlos como clases independientes, ya que poseen un conjunto de propiedades y comportamiento que puede ser independizado. Estas nueva clases colaboran con clase primaria Administradora.

nombre: Plan de ahorro *Justificación:* (HC3.1)		
Responsabilidades	*Colaboraciones*	*Contexto*
Cambiar el bien tipo (HC4.1) cambiarValorCuota cambiar Bientipo Registrar NuevoGrupo(HC5) cambiarValorCuota(HC5)	Administradora, Grupo, Bien tipo Administradora, Grupo, Administradora, Grupo,	Cambiar Bien tipo ArmarGrupo Cambiar Bien tipo

nombre: Planillas de adhesión *Justificación*: (HC3.1)		
Responsabilidades	*Colaboraciones*	*Contexto*
RegistrarSolicitudIngreso (HC4.1) Dar Datos (HC5.2)	Solicitante, Administradora Administradora	Solicitar Ingreso Armar Grupo Asegurar Adherente
nombre: Seguro de bien tipo *Justificación*: (HC3.1)		
Responsabilidades	*Colaboraciones*	*Contexto*
Mantener información sobre el contrato (HC5.2)	Adjudicatario, Administradora, Aseguradora	Asegurar bien tipo

nombre: Seguro de vida *Justificación* (HC3.1)		
Responsabilidades	*Colaboraciones*	*Contexto*
Mantener información sobre el contrato (HC5.2)	Adherente, Administradora, Aseguradora	Asegurar Adherente

nombre: SobreLicitación *Justificación*: HC3.1		
Responsabilidades	*Colaboraciones*	*Contexto*
CrearSobre (HC5) DarDatos (HC5)	Adherente Administradora Licitación	Licita Evaluar Licitación

Como se indica en la heurística **HC8**, se utilizó el plural para indicar el conjunto de sobre de Licitación que es utilizado por la clase Licitación. En el caso de los adherentes, la colección sí tiene significado propio y es el Grupo definido como otra CRCs.

8.1.3.3 Aplicación del Modelo de Reglas del Negocio

Siguiendo con la estrategia presentada en el Capítulo 4, se analiza el modelo de reglas para modificar las CRCs o agregar nuevas. Consideremos las reglas relacionadas al pago de la cuota. (Reglas 4,7,10,12 y 40-44, 51, 66 y 72) que indican aspectos relacionados a la administración de pagos, como por ejemplo en donde debe hacerse los pagos en término y atrasados, y cómo se hacen los mismos bajo la política actual. El conjunto de reglas define un importante comportamiento para la Organización ya que se determinan la forma de realizar los pagos y el control de la morosidad del grupo de un individuo en particular.

Veamos un ejemplo de análisis de una regla, realizado durante la definición de las mismas:

La regla 72: "El solicitante debe realizar los pagos correspondientes a la solicitud de adhesión en la Administradora"

La Regla representa un límite del solicitante.
Origen: interno (en este caso no se determina el área porque sólo se modeló la Administradora local).
Arbitrariedad: fija el lugar para pagos. Analizando los símbolos del LEL involucrados en la regla (y los símbolos con los que se relacionan) determinamos que a pesar de ser fijo es estable ya que el solicitante realiza este pago al entregar la solicitud de adhesión en la Administradora y esta acción siempre debe realizarse en ese lugar, de otra forma la Administradora debería tener un control de todas las planillas que entrega para realizar visitas a domicilio (o que se pague en los Bancos). Como no todas las planillas que entrega se convertirán en clientes concretos, no tiene sentido registrar toda la información de cada persona y tener personal trabajando a domicilio para estos casos.
Impacto: bajo, no es una regla que represente una restricción significativa para el solicitante del plan (receptor de la regla), sólo indica un lugar de pago.
Complejidad: baja porque solo tiene una propiedad (lugar de pago) y como se determinó durante el análisis de arbitrariedad se considera con pocas posibilidades de cambiar de valor.

Para simplificar el ejemplo no se muestra todo el análisis de estabilidad de cada regla sino que se discute en forma general las características del conjunto de la regla. Son reglas de origen interno, ya que es la Administradora quien determina las políticas de pago. Afectan a los adherentes y a la forma de realizar sus pagos. Si bien no es crucial la forma de realizar los pagos, constituye un servicio para el cliente, por lo que los cambios están sujetos a distintas ofertas que les puedan hacer a los clientes y las negociaciones con los

Bancos. Si bien todo este comportamiento puede ser realizado por la Administradora, dada su importancia puede ser abstraído y realizado por otra entidad (clase) que colabore con la Administradora. De esta forma los cambios producidos en las políticas de pago son manejados por esta clase sin que la Administradora se vea afectada.

Nombre: AdministraciónDePagos
Justificación: refleja al conjunto de reglas(Reglas 4,7,10,12 y 40-44, 51, 66 y 72) que describen los aspectos relacionados a los pagos: definición, pagos demorados, modificaciones de la forma de pago, Heurística HC9.1

Responsabilidades	*Colaboradores*	*Contexto*
determinarFormaPago	adherente Administradora	(inicio)
CobrarCuotaSegunReglaVigente	adherente	Pagar Cuota
CobrarCuotaenBanco	Administradora Adherente Banco	Pagar Cuota
CobrarCuotaAdministradora	Adherente Adjudicatario cuota	Pagar Cuota Vencida
CobrarCuotaAtrasada	Administradora Adherente	Pagar Cuota Vencida
calcularInteresesPunitorios	Adherente	Pagar Cuota Vencida
DevolverListaBanco	Solicitante Administradora	SolicitarIngreso
CobrarDerechoAdmisión	Solicitante Administradora	SolicitarIngreso
Cobrar Adjudicación	Adjudicatario	Adjudicación
DeterminaIncumplimientoGrupo	Grupo Administradora	Disolver Grupo
DeterminarCumplimientoPago	Adherente Administradora	Licitar Expulsar Adherente
calcularHaber	Adherente Administradora	DejarPlan

A partir de la creación de esta nueva clase se producen cambios en algunas CRCs. Las siguientes CRCs sufren modificaciones:

- En la CRC Adherente las responsabilidades:
  Pagar Cuota mensual
  Pagar Cuota mensual vencida
  Pagar Adjudicación
  ElegirBanco
  colaboran con la clase AdminitraciónPagos en lugar de la Administradora

- En la CRC Administradora, desaparecen las responsabilidades:
  cobrar Cuota Mensual en caso de Vencimiento
  calcularHaber
  y serán responsabilidades de la nueva clase AdminitraciónPagos

- La responsabilidad de la CRC Banco:
  DevolverComprobante Pagos
  colabora con la nueva clase AdminitraciónPagos en lugar de la Administradora

El resto de las responsabilidades de la nueva clase se forman en base a las reglas asociadas siguiendo la heurística **HC9**.

8.1.3.4. Evaluación de las CRCs

Siguiendo la heurística **HC11** se procede a "ejecutar" los escenarios. Para esto se toma cada uno de los escenarios y se lo escribe desde la perspectiva de orientación a objetos, es decir, cada episodio será una colaboración entre objetos.

Por ejemplo, se toma el escenario EvaluarLicitacion (Anexo 1)

Evaluar Licitaciones
Objetivo: Adjudicación de un bien tipo a uno o más adherentes
Contexto: Se realiza en las dependencias de la administradora.
Restricciones: Por lo menos un adherente debe haber licitado.
El dinero recaudado debe alcanzar para comprar mas de un bien tipo.
Actores: Administradora, escribano
Recursos: Sobres con licitaciones Comunicación fehaciente.
Episodios:
La administradora comunica fehacientemente por medios de prensa de difusión nacional el día y lugar donde se llevará a cabo la evaluación de la licitación.
Se abren los sobres y se crea un ranking de adherentes ordenados por monto licitado.
Si existen mas de una licitación con el mismo importe, entonces se toma como orden el del sorteo.
Dependiendo de la cantidad de adjudicaciones a realizar se toman los primeros adherentes del ranking.
ADJUDICACION del bien tipo al adherente ganador.

El primer episodio, "La Administradora comunica ....." no puede ser resuelto a partir de las responsabilidades existentes ya que la responsabilidad Evaluar Licitación se encarga de la evaluación de los sobres y no de la convocatoria. Se define una nueva responsabilidad ConvocarLicitacion para esa clase y una responsabilidad en la clase secundaria ComunicaciónFehaciente para hacer la convocatoria.
De esta forma el escenario se describe como:

Nombre: EvaluarLicitación
Administradora. ConvocarLicitación (colabora con la clase ComunicaciónFehaciente
Administradora. EvaluarLicitación (a partir de las colaboraciones cubre los episodios 2,3 y 4)
Administradora. AdjudicarBienTipo

Ahora analicemos el siguiente escenario:

Solicitar ingreso
Objetivo: Una persona desea ingresar a un plan de ahorro.
Contexto: El solicitante pide información acerca de los planes de ahorro existentes.
Actores: Administradora Solicitante
Recursos: Planilla de adhesión Cuota de ingreso.
Episodios:
El solicitante pide la planilla de adhesión.
El solicitante completa dicha planilla y la entrega, abonando la cuota de ingreso.
Si la administradora rechaza la solicitud evaluada, **entonces** ésta devolverá al solicitante el importe correspondiente a la cuota de ingreso.
Si la administradora acepta la solicitud, **entonces** se debe ASEGURAR ADHERENTE.

El escenario puede ejecutarse a partir de las clases, responsabilidades y colaboraciones existentes:

Nombre: Solicitar ingreso
Solicitante.Solicitar planilla de adhesión
Solicitante.Entregar la planilla de adhesión completa
Administradora.cobrarCuotaIngreso
Si Administradora acepta
    Administradora.aceptarSolicitudAdhesión
Aseguradora.asegurar Adherente
sino
Administradora.devolverCuotaIngreso

De esta forma se ejecuta cada uno de los escenarios. La documentación generada corresponde a una evolución de desarrollo de los escenarios, conforme se explica en [Breitman'98].

8.1.4 Definición del Modelo Lógico

Una vez definidas las CRCs, se procede a definir el modelo lógico, siguiendo los pasos presentados en la Figura 5.1 y aplicando las heurísticas definidas en capitulo 5. Para simplificar el ejemplo sólo se explican algunas de las heurísticas aplicadas, en particular las relacionadas a la aplicación de las reglas de Negocios.

- **Adherente** (HM1)
  - ⇒ Atributos
    - #Solicitante (HM1.1)
    - #Adherente (HM1.1)
    - #Grupo (HM1)
    - bancoElegido (HM1)
    - cantidadRechazos (HM1)
      cantidadRechazos<=Administradora.cantidadRechazosPermitidos (HM5.1)
    - #SeguroVida (HM1)
    - fechaUltimaCuotaPaga (HM1)
  - ⇒ Métodos
    - Pagar cuota mensual (cupón) (HM2.1/ HM2.3)
      bancoElegido.CobrarCuota (cupon, self)
      mesUltimaCuotaPaga= MesActual
    - Pagar cuota mensualAtrasada (cupón, mes) (HM2.1/ HM2.3)
      Administradora. CobrarCuotaAtrasada (self , cupón, mes)
      mesUltimaCuotaPaga= mes
    - Transferir plan(nuevoAdherente) (HM2.1/ HM2.3)
      Administradora.Transferencia
      (ComunicaciónFehaciente.notificar(self, nuevoAdherente,transferencia"))
    - Licitar (HM2.1)
      Administradora.Licitar (SobreLicitación.crear)
    - Rechazar bien tipo (HM2.1/ HM2.3)
      " post=pre@cantidaRechazo=Administradora.cantidaRechazoPermitido implies adherenteGrupo -> includes (adherente) = false (HM5.3) "
      cantidadRechazos +1

Administradora.AnalizarRechazo(self)

- Renunciar al plan (HM2.1/ HM2.3)
  com: ComunicaciónFehaciente.CrearForRenuncia(self)
  Administradora.eliminarAdherente(com)

- ElegirEntidadBancaria (HM2.1)
- PagarAdjudicación (Adj) (HM2.1/ HM2.3)
- bancoElegido? (HM2.4)

El atributo cantidadRechazos de la Administradora es afectado por la Regla 39 "Un adjudicatario puede rechazar hasta tres veces el bien tipo" que indica un límite en la cantidad de Rechazos. Se agrega la restricción correspondiente. Esta regla afectará métodos de la clase Administradora como se verá más adelante.

- **Adjudicatario** (HM1)
  - ⇒ Atributos
    - #adherente
    - Fecha de adjudicación
    - BienTipoElegido
    - SeguroTipo
  - ⇒ Métodos
    - Cambiar el bien tipo adjudicado
    - Elegir aseguradora para el bien tipo

- **Administradora** (HM1)
  - ⇒ Atributos
    - Razón social
    - Dirección
    - Planes de ahorro (HM1.1)
    - MontoDerechoAdmisión(HM3.1)
    - CantidadRechazosPermitidos (HM5.1)
    - MontoMinimoGrupo
  - ⇒ Métodos
    - Transferencia(cft) (HM2/ HM2.3)
      cft.solicitud?.cambiarDatos(cft)
      self.eliminarAdherente

    - analizarRechazoBienTipo(adh)
      Si (adh.cantidadRehazos?.<= cantidadRechazos)
      EliminarAdhrente
      BuscarSiguienteOrdendel Sorteo

    - AceptarSolicitudAdhesión (solAdh)
      pre: hoy() <= solAdh.fecha +10 or aceptarSolicitudAdhesion=true (HM5.3)

    - Devolver el valor de la cuota de ingreso
    - EliminarAdherente (adh) (HM2/ HM5.3)
      adh.grupo? eliminar (adh)
      Si adh.grupo? .cantAdh? <= CantidadMinimaGrupo
      Si adh.grupo?.planAhorro? eliminarGrupo(adh.grupo?.)
    - Determinar cuota mensual

- EvaluarLicitación
- Determinar seguro de vida
- Entregar bien tipo(adjudicatario)
  pre: hoy() <= adjudicatario. fechaAdjudicación +60 (HM5.3)
- CobrarAdjudicación (adh, Noadj)
  adh.grupo? agregarAdjudicatario(Adjudicatario.Crear(adh, No.Adj))
- Armar grupo (grupo)
  pre:grupo.adherentes->size>=grupo.planAhorro.tamañominimoGrupo (HM5.3)
- Disolver grupo
- Determinar cambio de bien tipo para un grupo
  (ComunicaciónFehaciente.new) notificar (self, grupo,"CambioBiernTipo") (HM5.3)
- CalcularHaber
- ConvocarAsamblea
- RealizarNuevoContrato
- LiquidarGrupo
  (ComunicaciónFehaciente.new) notificar (self, grupo,"CierreGrupo) (HM5.3)
- CalcularGastosAdministrativos

Como se indicó anteriormente, la Regla 39 determinó la creación de un atributo, CantidadRechazosPermitidos de la Administradora. Asimismo, la regla modifica el método analizarRechazoBienTipo ya que si el Adherente supera esa cantidad la Administradora lo puede eliminar. El método EliminarAdherente es afectado por la Regla 67: "La cantidad de miembros de un grupo es el doble de la cantidad de cuotas del plan elegido" que limita la cantidad de integrantes de un grupo. Como se puede observar los métodos de esta clase son afectados por varias reglas, ya que es la clase que representa el comportamiento del negocio que se ve afectado por las diferentes políticas de la Agencia.

- **Aseguradora** (HM1)
  - ⇒ Atributos
    - Razón social
    - Dirección
    - Seguros
  - ⇒ Métodos
    - Asegurar adherente
    - Asegurar bien tipo
    - Pagar el seguro de vida
    - Pagar el seguro de un bien tipo

- **BienTipo** (HM1)
  - ⇒ Atributos
    - Marca
    - Modelo
    - Tipo
  - ⇒ Métodos
    - Adjudicarse
    - SerEntregado

- **ComunicaciónFehaciente** (HM1)
  - ⇒ Atributos
    - Remitente

- Destinatario
- Contenido
- Tipo

⇒ Métodos

- Notificar (emisor, receptor, tipoNotificación) (HM2.5/ HM2.3)

Siguiendo la sub-heurística de factoreo de responsabilidades, **HM2.5**, se definió un solo método para representar todas las diferentes comunicaciones que en la CRC correspondiente Comunicación Fehaciente figuraban como diferentes responsabilidades. De esta forma se obtiene una clase más simple, con un método que se puede parametrizar. Cualquier incorporación de una nueva forma de comunicación no altera la interfase de esta clase. Los parámetros de esta nueva clase se determinan siguiendo la sub-heurística **HM2.3** analizando los escenarios en donde aparece la comunicación fehaciente.

Fabricante (HM1)

⇒ Atributos

- Razón social
- Dirección
- Bienes tipo

⇒ Métodos

- Entregar bien tipo
- Avisar cambios de precios de un bien tipo
- Avisar a la administradora la discontinuación de un modelo

- **Banco** (HM1)

⇒ Atributos

- Nombre
- Dirección
- Número de sucursal

⇒ Métodos

- Cobrar las cuotas mensuales
- Devolver cupones de pago

- **Grupo** (HM1)

⇒ Atributos

- Número (HM5.1)
- FechaInicio
- Plan de ahorro
- Adherentes
- Adjudicatarios

⇒ Métodos

- Aceptar el cambio de un bien tipo
- Rechazar el cambio de un bien tipo

- **PlanAhorro**(HM1)

⇒ Atributos

- Bien tipo
- CantidadCuotas
- ValorCuotaMensual=ValorCuotaPura+Adminitradora.CalcularGastosAdministrativos (HM5.1)
- ValorCuotaPura

- Grupos
- tamañominimoGrupo= self.cantidadcuotas*2 (HM5.1)

⇒ Métodos
- CambiarBienTipo
- Transferir

Los atributos de esta clase se ven afectado por dos reglas no funcionales. El valorCuotaMensual es afectado por la Regla 70: "La cuota mensual se calcula como la cuota pura más los gastos de administración" mientras que se define un atributo tamañominimoGrupo para reflejar la Regla 67: "La cantidad de miembros de un grupo es el doble de la cantidad de cuotas del plan elegido".

- **Licitación** (HM1)

⇒ Atributos
- Fecha
- Lugar
- Sobres

⇒ Métodos
- Convocar
- DarDatos
- CrearRankingAdherentes

- **PlanillaAdhesión** (HM1)

⇒ Atributos
- Número (HM5.1)
- Fecha
- solicitante
- Plan de ahorro
- Firma
- Contrato

⇒ Métodos
- Registrar la solicitud de ingreso

Siguiendo la sub-heurística **HM5.1** se define un atributo Número para representar la Regla 71: "La solicitud de adhesión aceptada tiene un número de orden dentro del grupo"

- **SeguroBienTipo** (HM1)

⇒ Atributos
- Número
- Bien tipo
- Importe
- Características

⇒ Métodos
- devolverDatos

- **SeguroVida** (HM1)

⇒ Atributos
- Número
- Persona
- Importe

- Características

⇒ Métodos

- devovlerDatos

- **Solicitante** (HM1)

⇒ Atributos

- Número de solicitante
- Nombre y apellido
- Edad
- DNI
- Dirección
- TEL

⇒ Métodos

- SolicitarPlanillaAdhesión
- EntregarPlanillaAdhesiónCompleta

Esta clase será modificada cuando se aplique los patrones análisis de Roles, en donde se definirá una clase `Persona` tendrá los datos personales, dejando que la clase `Solicitante` tenga los datos propios de su rol en la `Administradora`.

- **Sorteo** (HM1)

⇒ Atributos

- Fecha
- Lugar
- Método
- Lista de adherentes
- Escribano

⇒ Métodos

- DefinirGanadorSorteo

- **AdministraciónPagos**

⇒ Atributos

- listaBancos (HM1.1)

⇒ Métodos

- determinarFormaPago
- cobrarCuotaSegunReglaVigente
- CobrarCuotaenBanco
- CobrarCuotaenAdministradora
- CobrarCuotaAtrasada(adh, cupón, mesPago)
  self.actualizarIntereses (adh. mesUltimaCuotaPaga? , mesPago)
  cupón.sellar
- CalcularInteresePunitorios
- actualizarIntereses (adh. mesUltimaCuotaPaga? , mesPaga) (HM2.2)
- devolverListaBanco
- cobrarDerechoAdmisión
- cobrarAdjudicación
- DeterminaIncumplimientoGrupo
- DeterminarCumplimientoPago
- calcularHaber

8.1.4.1 Definición de los Diagrama de Asociaciones e Interconexión

Como se indica en la figura 5.1 del Capítulo 5, el siguiente paso es la definición de un diagrama de asociaciones como resultado de aplicar la heurística HM3 para modelar las relaciones entre clases. Por ejemplo, a partir de los términos del LEL de la figura 8.2 se puede realizar el siguiente análisis: en la primer noción de Adherente aparece el verbo ser conjugado, que es un verbo del tipo *bottom-up/top-down* [Juristo'98] que conecta al término adherente con un solicitante. Esto indica claramente la relación de herencia(HM3.1). La segunda noción tiene el sustantivo "integrante", que deriva del verbo integrar del tipo *content-composition-verb*. Y la cardinalidad está indicada en uno. De la misma forma en el término Grupo la primer noción sugiere explícitamente una relación de composición. De esta forma se establece una relación parte-de entre Grupo y Adherentes **(HM3.2)** . Las asociaciones comunes son obtenidas aplicando la heurística **HM3.3**. La figura 8.3 muestra el resultado final.

Adherente	**Grupo**
Nociones:	***Nociones:***
• Es un Solicitante aceptado por la administradora. • Integrante de un grupo.	• Conjunto de adherentes que desean obtener un mismo bien tipo, siguiendo un mismo plan de ahorro. • Inicialmente la cantidad de adherentes de un grupo es el doble de la cantidad de cuotas del plan de ahorro al que corresponde
Impacto:	***Impacto:***
• Pagar cuota mensual. • Transferir plan. • Licitar. • Rechazar bien tipo. • Renunciar al plan de ahorro.	• Disolver grupo. • Aceptar o rechazar el cambio de un bien tipo.

Fig. 8.2: Términos del LEL correspondiente al caso de estudio Círculo de Ahorro

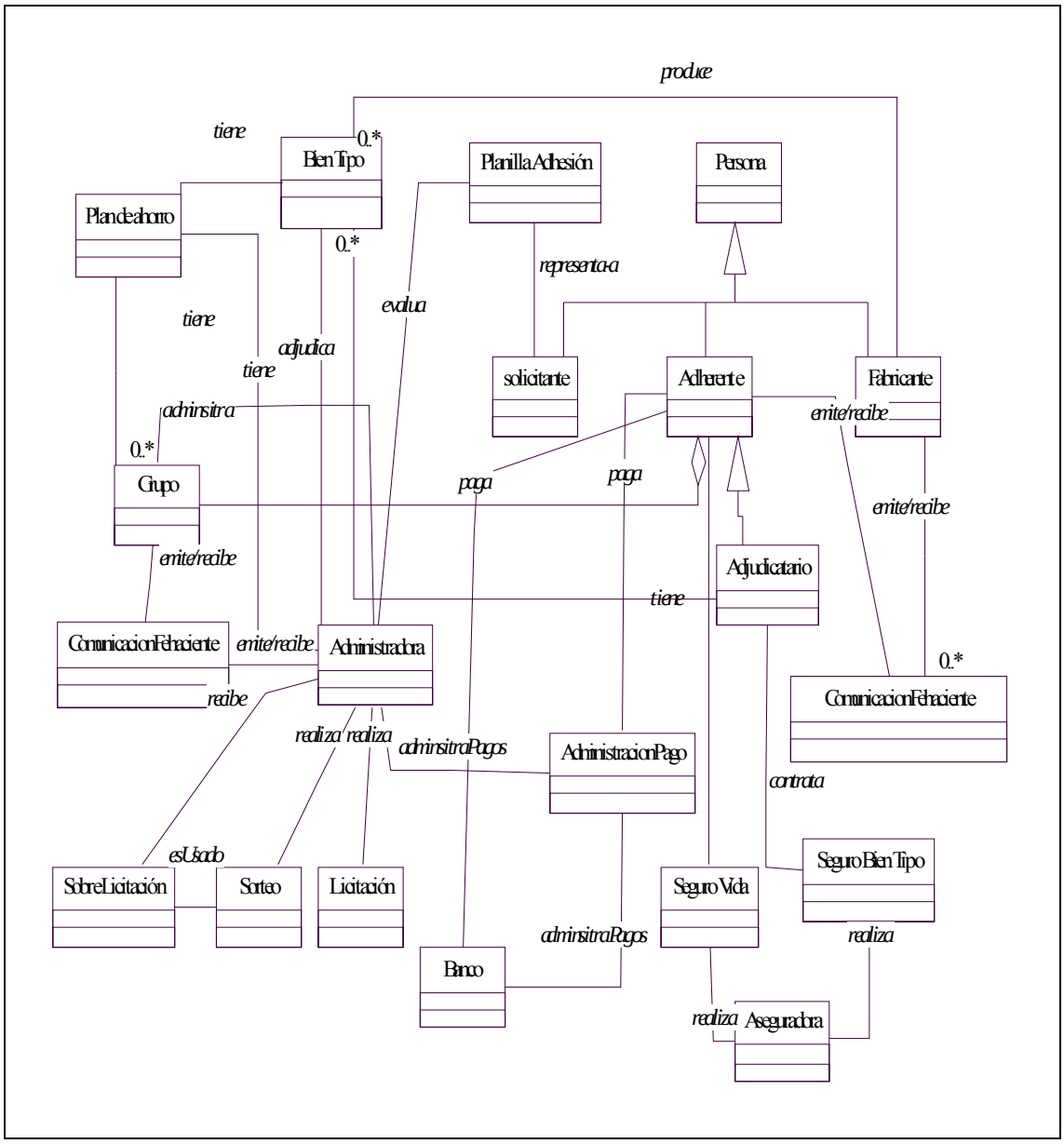


Fig. 8.3: Diagrama de Asociación

El siguiente paso es definir la comunicación entre los objetos, para completar la visión estructural del diagrama representado en la Figura 8.3. Se decide definir un diagrama de interconexión, para lo cual se deben establecer todos los datos entre las clases. Por ejemplo analizando la clase Administradora y los métodos "DeterminarCambioBienTipo" y "LiquidarGrupo"(entre otros) se observa que aparece Grupo como colaborador. Analizando los escenarios "Cambiar bien tipo", "Disolver Un Grupo" que son contextos de las responsabilidades asociadas a cada método, se encuentra que los datos que se pasan son: Administradora le envía el aviso Disolución del Grupo y Cambio del Bien Tipo y el Grupo le envía el Rechazo / aceptación del bien Tipo. Realizando el análisis completo de las clases se obtiene los siguientes parámetros:

M1 Disolución
M2 Mensaje
M3 Planilla de adhesión
M4 Aviso de cambio de bien tipo

M5	Rechazo/aceptación bien tipo
M6	Bien tipo
M7	Lista de aseguradoras
M8	Asegurar
M9	Datos personales
M10	Cupones de pago
M11	Ganador
M12	Modelo y tipo
M13	Lista de adherentes
M14	Ranking
M15	Rechazo
M16	Aviso de fecha y lugar
M17	Sobre con licitación
M18	Pagos
M19	Datos del seguro
M20	Cupón sellado
M21	Plan de ahorro
M22	Datos del plan de ahorro
M23	Cambio de precio
M24	Discontinuación del bien tipo

La figura 8.4 muestra el Diagrama de Interconexión obtenido con la aplicación de la heurística **HM6** para todas las clases del modelo.

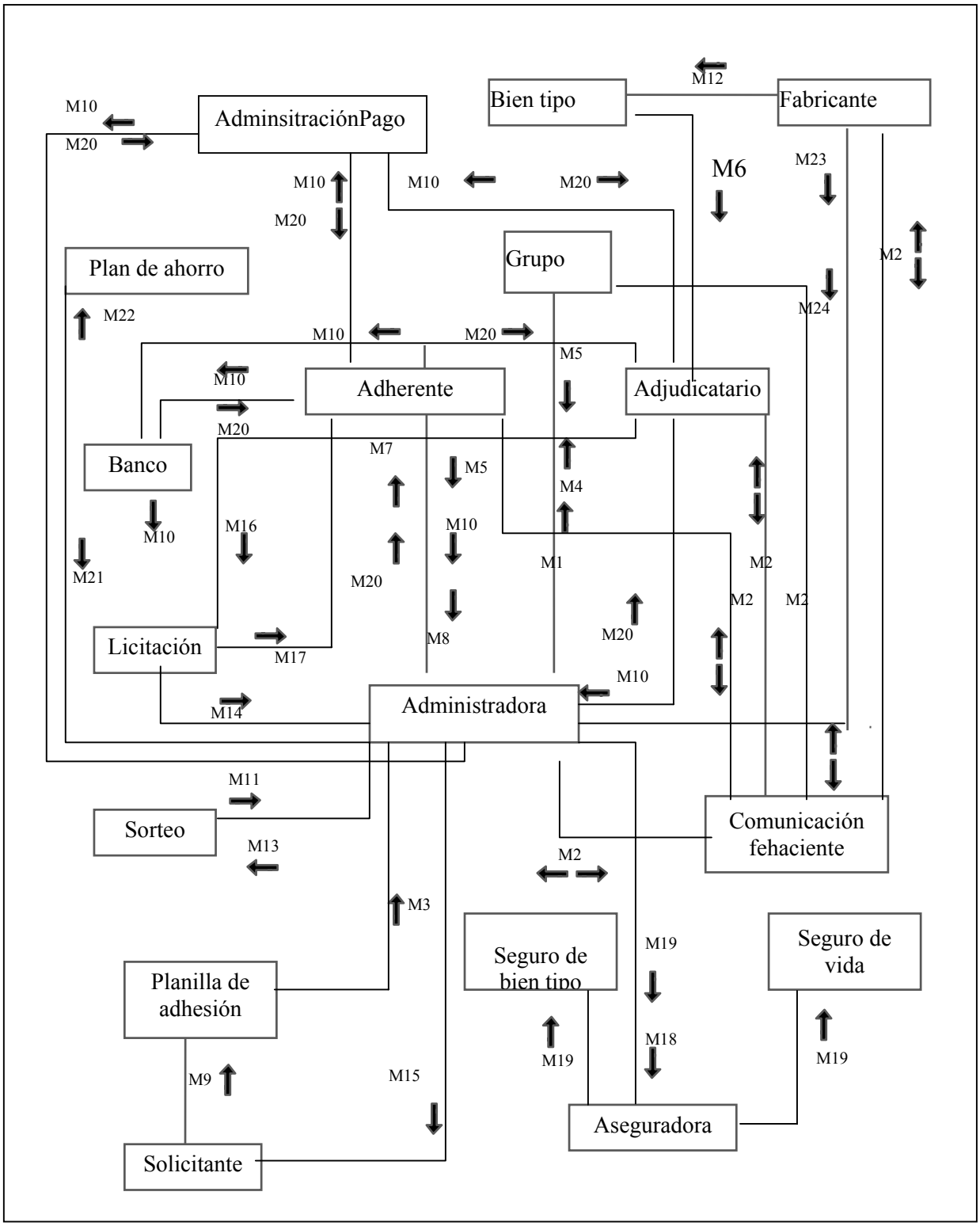


Fig. 8.4: Diagrama de Interconexión

8.1.4.2 Aplicación de patrones de Análisis

Para finalizar con la definición del modelo lógico, se intentan aplicar patrones de análisis con el objetivo de mejorar al modelo. Analizando algunas de las responsabilidades de la Administradora, se determina que tiene eventos recurrentes en ciertos días del mes. Todos los quintos días hábiles del mes la Administradora debe mandar una notificación fehaciente a los adherentes que todavía no han abonado su cuota mensual (realizado ahora por la clase AdministraciónPago con el método DeterminarCumplimientoPago). También debe sortear y licitar el bien tipo el primer día hábil de cada mes. A partir de esta situación, siguiendo la heurística **HM4.2** puede aplicarse el patrón "Scheduler " [Fowler'97] junto con el patrón "elemento de Scheduler". Siguiendo con el análisis y aplicando la heurística **HM4.1** se puede observar que el solicitante, adherente y adjudicatario, son roles que no pueden convivir en el tiempo para una misma persona, primero una persona es un solicitante, luego se transforma en adherente y finalmente en adjudicatario. Esta situación se dará si se considera que una persona sólo puede pertenecer a un grupo y se podría modelar con el pattern "*subtype role*". Pero, si consideramos que una persona puede pertenecer a varios grupos, entonces una persona puede estar en varios grupos con diferentes roles al mismo tiempo. Un patrón que ayuda a modelar esta situación es el pattern "*object role*" [Fowler'97]. En la figura 8.5 se muestra el patrón aplicado al modelo de objetos.

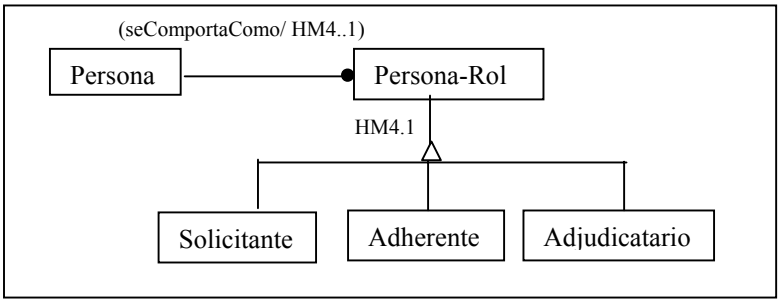


Fig. 8.5: modelo de objetos modificado por el patrón "object rol"

8.1.5. Determinado los límites del software

Básicamente el cliente es quien decide que se automatizará. Como se indicaron en las heurísticas el ingeniero de software puede darle estimaciones de tiempo a partir de los modelos de objetos. Sin embargo, es él quien finalmente decide.

¿Que va a hacer el sistema?

En nuestro ejemplo, "el cliente" por cuestiones legales debe hacer tanto el sorteo como la licitación de manera publica. Sin embargo el cliente quiere guardar los resultados de ambos, así como también las ofertas de licitación. Analizando las reglas se determina la funcionalidad del sistema.
Por ejemplo:

Regla 1: La Administradora entrega Bien Tipo por sorteo y licitación.

Si bien el análisis de sorteo y licitación se hará manualmente por cuestiones legales, se quiere guardar toda la información relacionada. A partir de ahí se analizan los escenarios en donde aparece esta regla a partir de las relaciones de trace y se comienza a trabajar con las clases del software.

Regla 2: La Administradora acepta solicitantes para sus planes de Ahorro

Se decide implementarla guardando y generando la planilla, para poder automatizar el control de los datos de los solicitantes.

Regla3: La Administradora debe ofrecer distintos planes de ahorro

Se decide implementarla guardando y generando la planilla.

¿ Cuales son las clases externas y cuales las internas?

Se analizan loa escenarios asociados para determinar las clases. Por ejemplo para la primer regla se identifican los escenarios Evaluar licitaciones y Sorteo.

Evaluar Licitaciones

.....

Episodios:

1. La administradora comunica fehacientemente por medios de prensa de difusión nacional el día y lugar donde se llevará a cabo la evaluación de la licitación.
   (Esto es resuelto por los métodos convocarLicitación de la clase Administradora y notificar de la clase comunicación fehaciente, la cual crea un documento impreso o un archivo.)
2. Se abren los sobres y se crea un ranking de adherentes ordenados por monto licitado.
   (Si bien estas operaciones se hacen manualmente, se guarda el ranking en la clase sorteo.)
3. **Si** existen mas de una licitación con el mismo importe, **entonces** se toma como orden el del sorteo.
   (Se usa información manual, si embargo el sistema guarda los datos.)
4. Dependiendo de la cantidad de adjudicaciones a realizar se toman los primeros adherentes del ranking.
   (Se usa información manual, si embargo el sistema guarda los datos.)
5. ADJUDICACION del bien tipo al adherente ganador.
   (El método adjudicar un bien tipo de la clase Administradora crea un nuevo objeto Adjudicatario. Se continúa el análisis para el escenario Adjudicación.)

Las personas / entidades físicas siempre serán externas, se deberá analizar si existe una clase de software que la refleje, en este caso Solicitante, Adherente, Adjudicatario, Compañía de Seguros, Banco, Fabricante. Claramente las tres últimas son externas a la organización y no se crearán clases asociadas. Para las tres primeras, se modelarán clases de software que permitan guardar la información de las personas dentro de la Administradora.

¿Cuales son las entradas/ salidas?

Las entradas y salidas se establecen a partir de las comunicaciones de las clases externas definidas anteriormente. A partir de las nuevas clases de software, se debe establecer que necesitan de las entidades externas El diagrama de capas de la figura 8.5 muestra las comunicaciones entre las entidades en todos los niveles establecidos. En el diagrama se observa las diferentes capas creadas en el *UofD*, la capa interna es la Administradora que contiene al sistema de software y al empleado que interactúa directamente con él, luego la capa del Círculo de Ahorros propuesto por la Administradora mostrando los distintos roles de la persona que solicita planes de ahorro (solicitante, adherente y adjudicatario). Finalmente, la última capa contiene las entidades externas al Círculo, es decir, los Bancos, Aseguradoras y el Fabricante, que no pertenecen a la Administradora pero se relacionan a través de las diferentes actividades que deben realizar para poner en marcha el círculo de Compras en el mundo externo siguiendo las actuales políticas de la Administradora y del Mercado.

Interfase

I1 Planilla de adhesión
I2 Datos Cupones de pago
I3 Datos Sobre con licitación
I4 Datos de los planes de ahorro

Flujos

F1 Planilla de adhesión vacía
F2 Lista de precios
F3 Aviso de cambio de bien
F4 Bien tipo
F5 Cupón de pago e importe
F6 Cupón sellado
F7 Rechazo de bien tipo
F8 Lista de aseguradoras
F9 Aviso de discontinuidad de un bien tipo
F10 Datos personales
F11 Ganador
F12 Rechazo bien tipo
F13 Datos de los planes de ahorro
F14 Datos del seguro
F15 Aviso de fecha y lugar del sorteo

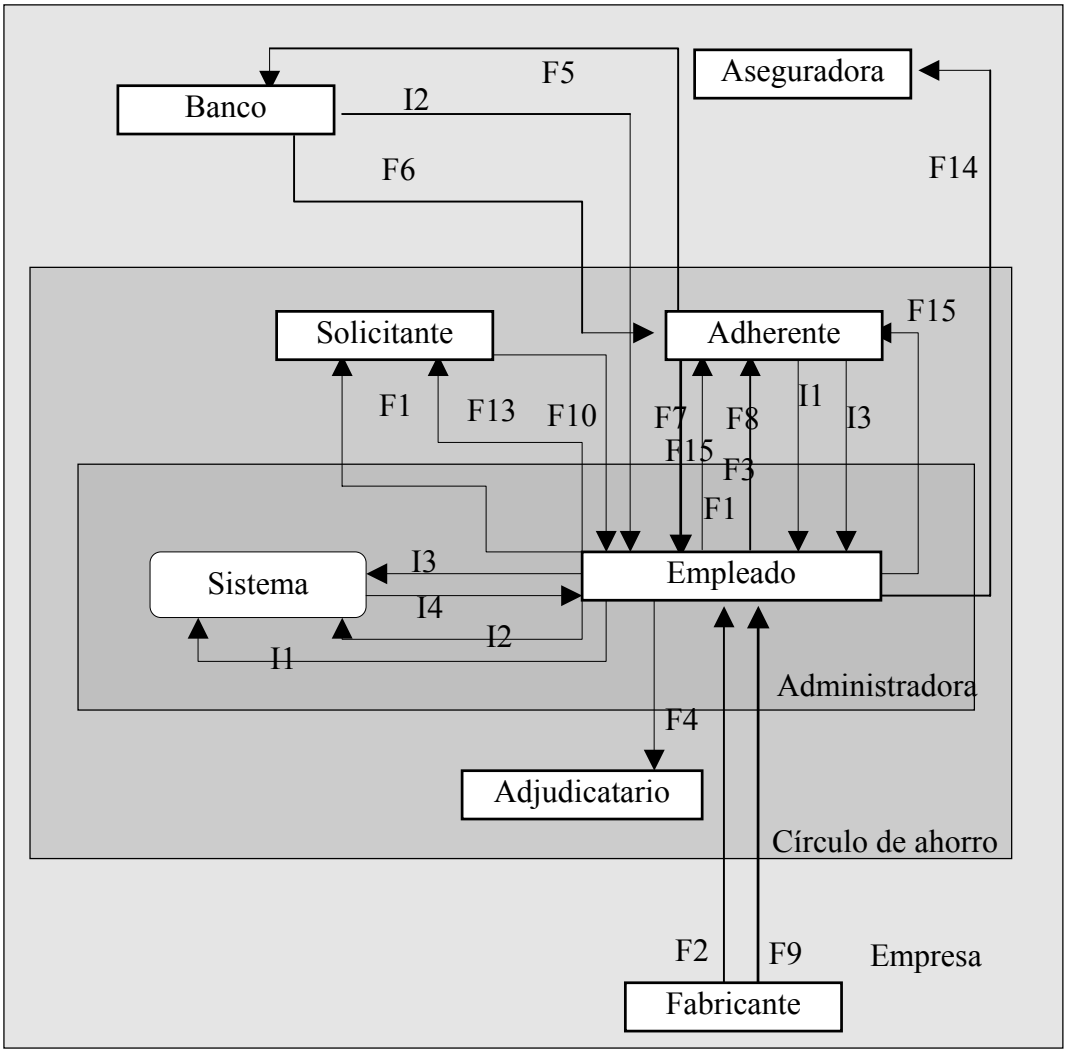


Fig. 8.5: Diagrama de capas

8.2 Otras Aplicaciones de la estrategia

Esta estrategia se utilizó para realizar la modelización conceptual de diferentes casos de estudio:

Sistema de pasaporte Argentino: representa al sistema de emisión de pasaporte realizado por la Policía Federal, única entidad autorizada para emitirlos (este sistema es el anterior al que actualmente está vigente) Los modelos LEL (34 símbolos) y escenarios

(24 escenarios) fueron producidos por el grupo de investigación de la Universidad de Belgrano [Leite´96]. Las CRCs resultantes se encuentran detalladas en el reporte de beca [Leonardi'96]

Meeting Schedulder: representa un organizador de reunión, basado la organización de Reuniones de la Universidad de Belgrano, cuyo grupo de investigación desarrolló los modelos del LEL y escenarios [Hadad´98]. Algunos de sus resultados se muestran en [Leonardi'97].

Sistema de alumnos PUC-Rio: describe el sistema de post-graduación de la PUC-Rio. Se obtuvieron 6 clases primarias 10 secundarias. Este caso de estudio continuó hasta la etapa de implementación. Todos los resultados se pueden obtener en www.inf.puc-rio.br\ ~dilbert.

Auto-aplicación de la propuesta de derivación de un modelo de objetos: se modelo esta estrategia de derivación de objetos. Este trabajo fue realizado por dos estudiantes de la Carrera de Ingeniaría de Sistemas de la UNCPBA para su tesis de grado [Garcia'00] dirigidos por el Ing. Doorn y la Lic. Leonardi. Se describieron 64 escenarios 78 términos del LEL para representar la construcción del modelo de escenario, LEL, y derivación de CRCs y modelo lógico de objetos.

Módulo de Administración Presupuestaria y Economía Financiera: Desarrollo de un Sistema Integrado De Información -SIDI- para apoyo a la gestión académico-administrativa realizado en la Universidad Tecnológica Nacional Facultad Regional Santa Fe [Ballejos'00].

Estos casos de estudio fueron desarrollados sin tener en cuenta las heurísticas que hacen referencia al modelo de Reglas de Negocio. La inserción de este modelo en la estrategia merece un estudio más profundo y es la principal actividad a desarrollar en los trabajos futuros (capítulo 9)

8.3 Conclusiones obtenidas a partir de los casos de estudio

Sobre la base de los resultados de los trabajos mencionados en la Sección anterior se puede concluir las siguientes ventajas de la estrategia:

- Dada su naturaleza orientada al cliente, tanto los modelos de escenarios, LEL y las reglas como también la estrategia que los manipula para obtener el modelo de objetos son fáciles de comprender. Luego de una corta explicación se puede comenzar a trabajar inmediatamente.
- El lenguaje natural facilita la validación con los stakeholders.
- Las heurísticas son útiles si la persona responsable para la definición del modelo de objetos no es la misma persona que desarrolló los modelos de requisitos ya que permiten manipular el gran volumen de información generado por estos modelos. En el caso que sea la misma persona la que desarrolla todo el trabajo, no es necesario seguir estrictamente las heurísticas (solo para validar) porque ya dispone de un profundo conocimiento sobre los símbolos, sus impactos y su comportamiento obtenido durante la definición del LEL, los escenarios y las reglas.
- El LEL y las reglas permiten analizar un objeto desde una perspectiva global, más general que la provista por las descripciones específicas dadas por los episodios de los escenarios.
- Desde el punto de vista de la Orientación a Objetos, es una mejora para la fase de Requisitos, incorporando técnicas centradas en los clientes

- Desde el punto de vista de los requisitos, esta estrategia extiende la tarea concerniente a la captura del conocimiento del cliente para definir un modelo de objetos que permitan al analista visualizar el macrosistema en términos de objetos, el mismo paradigma que se usará en la siguiente fase de diseño. De esta forma se logra que la transición entre el modelado conceptual y el diseño sea un proceso más natural [Kotonya'98].

Las principales dificultades que se evidenciaron en el uso de la estrategia son:

- Uno de los problemas más significativos con que se lidió fue el del nivel de detalle al que se puede llegar para representar la realidad del macrosistema. Según el nivel de detalle se podían detectar más o menos clases.
- Un problema que surgió fue el de tratar de modelar el macrosistema en cuanto se refiere a las actividades intelectuales. Esto problema surgió en el caso del Auto-aplicación de la estrategia, donde se estaba describiendo un proceso que requiere habilidades intelectuales, desarrollados por el "ingeniero de requisitos", ya que, por ejemplo, cuando tiene que tomar conocimiento del macrosistema o de otras clases no hay una forma sencilla de representarlo. Se puede citar como ejemplo a la responsabilidad que dice: *lee la documentación intentando comprender el vocabulario que utiliza el cliente-stakeholder*. En este caso no existe forma de representarlo como una responsabilidad, pero si se elimina se pierde importante información. Es por esta razón que se decidió dejar expresadas este tipo de responsabilidades.
- También es necesario una cuidadosa identificación de la redundancia de los modelos de requisitos para que no se modele aspectos redundantes en el modelo de objetos.
- Finalmente, es necesario decidir cuales relaciones de trace se quieren rastrear, ya que se genera gran cantidad de información que es imposible mantenerla actualizada. Esta decisión es dependiente del sistema de soporte automatizado y de la facilidad que provea para la administración de trace.

Capítulo 9
Conclusiones

En esta tesis se presenta una estrategia que pretende reducir la brecha que existe entre los modelos de requisitos basados en Lenguaje Natural y una solución orientada a objetos. Para esto se definió un conjunto heurísticas y pasos que guían en la construcción de un modelo conceptual de objetos a partir de los modelos de LEL, escenarios y reglas de negocio, éste último introducido en esta tesis. Esta estrategia puede ser usada con cualquier metodología como una mejora para la etapa de adquisición de requisitos.

9.1 Comparaciones con otras estrategias

Como se mencionó en la introducción de esta tesis(capítulo 1 sección 1.3), existen varias metodologías que utilizan el concepto de escenarios o sus variantes para el desarrollo de un modelo de objetos. En esta sección se describe brevemente sus similitudes y diferencias con la estrategia presentada en esta tesis.

OBA[Rubin'92] es una de las primeras metodologías que propone una forma de escenarios, llamados *scripts*, para describir las secuencias de acciones del sistema. Nuestra estrategia difiere en el concepto de lo que es para nosotros un escenario: los *scripts* de OBA son de un nivel de abstracción más bajo, enfocando su atención en los eventos, estados y transiciones del sistema. De esta forma la dinámica de modelización que propone esta estrategia es típicamente bottom-up en lugar de top-down, lo cual puede ser útil en algunos casos pero que dificulta la obtención de una vista general de la dinámica del sistema. Los episodios de los escenarios utilizados en esta tesis describen más información que la brindada por una secuencia de comunicaciones entre iniciadores-participantes. Por otro lado, el diccionario usado en OBA describe las características básicas de una entidad (por ejemplo, si es iniciador o participante del *script*) sin tener en cuenta el impacto global del símbolo en el macrosistema. En nuestra estrategia, este problema se resuelve con el uso de LEL, cuya rica semántica permite tener un mayor conocimiento de los actores, acciones y recursos que forman al macrosistema.

Objectory [Jacobson'92] usa lenguaje natural para describir *Use-Cases*. Los *Use Cases* son derivados de las principales tareas realizadas por los actores, de forma similar a nuestra estrategia, pero en la propuesta de Jacobson, los *Use-Cases* son descripciones de la interfase entre los actores y el sistema software, por lo tanto, estas tareas describen acciones que afectan al sistema de software y desencadenan algún comportamiento. Desde el punto de vista orientado a objetos, Jacobson propone la construcción de dos modelos de objetos durante la etapa del modelo conceptual: el modelo de objetos del dominio y el modelo de objetos de análisis. El objetivo del primero es ayudar en el entendimiento del modelo de requisitos y el vocabulario y la comunicación con el stakeholder. Esta expresado por el nombre de los objetos, atributos lógicos y asociaciones estáticas. No se define ninguna heurística para encontrar esos objetos. Las principales diferencias con nuestra propuesta son: (a) nuestro objetivo es construir un primer modelo para ayudar a los ingenieros en las primeras etapas del desarrollo de un software

orientado a objetos luego que el comportamiento del macrosistema y su vocabulario hayan sido validados con el stakeholder. (b) nuestro modelo lógico describe los objetos del macrosistema incluyendo actores (c) Finalmente en nuestra estrategia los objetos son definidos a partir del comportamiento que se detecta en las nociones de los símbolos del LEL. El segundo modelo expresa al sistema a través de los objetos de interfase, entidades y control y provee algunas heurísticas para derivarlos. Este modelo ya considera aspectos de diseño, etapa no tratada en nuestra estrategia.

Las estrategias propuestas por *Objectory*, *OMT* [Rumbaugh'91] y Booch [Booch'94] confluyeron en el lenguaje *UML* y el *Unified Software Development Process*, conocido como *RUP/UML*. *RUP/UML* propone para las primeras etapas del desarrollo de software(previo a la etapa de definición de requisitos), la construcción de un Modelo de Negocios para comprender el contexto del Sistema. La filosofía es similar a la presentada en esta estrategia, es decir, comprender el contexto del sistema que implica entender el proceso de negocios y la organización. El modelo de negocios está compuesto por un modelo de *Use-Cases* y un modelo de objetos de negocio. Los pasos para su construcción son la identificación del modelo *Use-Cases* del negocio a partir de los actores de negocio y la identificación de los objetos para el modelo de *Use-Cases*. No provee heurísticas precisas para la identificación de los *Use-Cases* ni para la definición del modelo de objetos. Se menciona la importancia del uso de Reglas de Negocio para aplicar al modelo de objetos, pero no se provee estrategias para identificarlas ni para relacionarlas con los objetos. Al igual que toda la filosofía de desarrollo de *RUP/UML*, los objetos surgen principalmente de los *Use-Cases*, siendo ésta otra diferencia con nuestra estrategia, la cual utiliza principalmente el LEL para definir los objetos, con la ventaja de tener una visión global y más detallada del *UofD,* como se menciona en la introducción de esta tesis (capítulo 1, sección 1.3). *RUP/UML* utiliza este modelo para capturar los requisitos y definir posteriormente los modelos de análisis. A pesar de las diferencias de procedimiento en cuanto a la definición de los objetos, es importante destacar que, tal vez la metodología más utilizada en el contexto de un desarrollo orientado a objetos como lo es *RUP/UML*, hace hincapié en la importancia de definir un modelo de objetos que represente a la organización y su negocio como una etapa previa a la definición de los límites de software y el modelo de análisis.

RDD [Wirfs'90; Wirfs'95], y *RBM* [Cockburn'99]son estrategias que utilizan *CRCs* y *Use-Cases* para el modelado de objetos. Las principales diferencias con relación al modelo de *CRC* presentado en esta tesis son: la definición de un modelo de objeto que represente al *UofD* como una etapa previa a considerar los límites de software, el uso de modelos orientados al cliente para obtener el modelo de objetos y el énfasis de traceability. En particular *RBM* es la estrategia que más heurísticas ha desarrollado para la construcción de CRCs las cuales han servido como base para las heurísticas de definición del modelo CRCs de la estrategia aquí presentada (capítulo 4).

En los últimos años, existen propuestas que extienden el uso de los *Use-Cases* en el contexto del desarrollo orientado a objetos. En [Diaz'00] el concepto de *Use-Cases* es usado en el proceso de un modelado orientado objetos que hace hincapié en los cambios potenciales del sistema. Los *Use-Cases* constituyen un Baseline que permite la identificación de los cambios y en consecuencia, la identificación de los cambios en las responsabilidades, y relaciones entre objetos. Sin embargo, no se presenta un conjunto explícito de heurísticas para identificar los objetos desde los *Use-Cases*. Esta propuesta es muy interesante porque ataca el problema del proceso de reingeniería y en este sentido

permite completar nuestra estrategia, que si bien no se concentra en el mantenimiento del sistema, permite genera gran cantidad de información de trace como resultados de aplicar las heurísticas que pueden ayudar en esta etapa.

En [Mattingly'98] se presenta una estrategia interesante que separa el comportamiento interno del sistema del comportamiento externo con distintos tipos de *Use-Cases,* derivando los casos de test de los últimos. En nuestra propuesta, no existe un modelo de escenarios que reflejen el sistema de software, por lo que podría adaptarse esta propuesta, y descomponer los escenarios, teniendo en cuanta su parte externa e interna con respeto al sistema de software completando así la estrategia presentada en el capítulo 6. Sin embargo, este nuevo modelo de escenarios esta dentro del proceso de evolución de los escenarios [Breitman'98], tema que escapa al alcance de esta tesis.

Otra estrategia que integra *Use-Cases* con especificaciones orientadas a objetos es la propuesta de Dano et al. [Dano'97]. El objetivo de ese trabajo es producir especificaciones orientadas a objetos dinámicas, mientras que nuestra propuesta es concerniente a especificaciones estáticas. A pesar de esta diferencia fundamental, ellos utilizan el concepto de *Use-Cases* de la misma forma que Objectory, es decir, escenarios en los cuales se ha establecido un limite con el sistema. La notación que utiliza es más rigurosa, a través de tablas similares a los *scripts* de OBA y las redes de Petri. Este trabajo persigue un fin similar al anterior por lo que es también una alternativa complementaria al nuestro, que puede ser utilizado como paso posterior a la elaboración del modelo conceptual de objetos, una vez que los límites de software se hayan establecido y se puedan representar a los escenarios con una notación semi-formal para avanzar a la etapa de diseño.

En [Rosson'99] se propone una estrategia interesante que integra escenarios de interacción con modelado conceptual y diseño orientado a objetos. Al igual que el resto de las propuestas, la estrategia de Rosson involucra el modelo de escenarios en el proceso de desarrollo de software orientado a objetos. La principal diferencia con nuestra propuesta es que la estrategia de Rosson construye el primer modelo de objetos a partir de escenarios concretos, teniendo en cuenta las entidades instanciadas que aparecen en él. Define un modelo separado para cada escenario. De esta forma, comienza con un modelo concreto hacia el abstracto, definiendo un "punto de vista" para cada objeto, una versión centrada en el objeto para cada escenario de interacción en el cual aparece le objeto. A pesar que esta estrategia es útil para sistemas de tamaño pequeño o casos particulares en que los ingenieros necesitan analizar en detalle algún comportamiento particular de un objeto, se vuelve inmanejable a medida que crece el número de escenarios. Como en nuestra estrategia se define un primer modelo de objetos a partir del LEL, el análisis previo realizado para definir al léxico, brinda un profundo conocimiento de las entidades y procesos más relevantes y sus impactos en el ambiente. De esta forma se obtienen las responsabilidades y colaboraciones más generales de un objeto y luego se las refina analizando el modelo de escenario y reglas. La *object network* de la propuesta de Rosson es similar a los diagramas de asociación e interconexión de nuestra propuesta, a pesar que la *object network* es una instanciación siguiendo el espíritu de la estrategia. A pesar que en nuestro trabajo no se tratan aspectos de diseño, encontramos muy útiles los *usability claims* para permitir el análisis de las consecuencias negativas y positivas de diferentes alternativas de diseño de los objetos, mejorando el aspecto de racionalidad de los objetos. Creemos que la propuesta de Rosson sirve como un complemento a la nuestra, muy conveniente en el análisis de objetos particulares para darle mayor racionalidad.

Como en nuestra propuesta, la de Rosson relaciona los objetos con escenarios dando importante información para la administración de traceability.

Con respecto a propuestas basados en lenguaje natural, un trabajo de [Juristo'98] presenta una estrategia para generar especificaciones Orientadas a Objetos a partir de modelos en lenguaje natural, enfatizando, al igual que en nuestra estrategia, la importancia de trabajar sobre la relación entre los modelos lingüísticos y el mundo conceptual. Las heurísticas presentadas en esta tesis utilizan la clasificación de los verbos presentada en esta propuesta para determinar las relaciones estructurales entre clases (capítulo 5, heurística **HM3**). La estrategia de Juristo presenta dos tipos de patrones lingüísticos para modelar el Universo de Discurso, considerando ya en esta etapa al sistema de software. El primer tipo (SUL) describe los conceptos del Universo de Discurso de una forma más estructurada que el LEL, mientras que el otro tipo (DUL) describe el comportamiento del sistema de software comenzado con los *Use-Cases*. La propuesta presenta un conjunto de reglas para definir un modelo de objeto basadas en una correspondencia entre los patrones lingüísticos y los patrones conceptuales. Se modelan aspectos estáticos y dinámicos siguiendo estrategias fundamentalmente sintácticas. En nuestra estrategia se propone un conjunto heurísticas informales que guían en la definición de un modelo de objetos sin tener patrones de objetos predefinidos a los cuales deben ajustarse los objetos definidos (los patrones de análisis son introducidos para mejorar el modelo). Por otro lado, nuestra estrategia tampoco se puede automatizar totalmente ya que los aspectos semánticos están profundamente involucrados en la definición de los objetos, por lo que es necesario la participación del stakeholder. Por ejemplo, en la propuesta de Juristo, una clase puede aparecer como resultado de una correspondencia entre los patrones lingüísticos y conceptuales y desaparece por cuestiones de cardinalidad. En nuestra propuesta, esta entidad no seria definida como objeto si es que su comportamiento (determinado por los impactos) no es relevante para el sistema. Creemos que la eliminación de posibles objetos debido a aspectos de cardinalidad, tiene problemas potenciales, porque no considera la riqueza de las relaciones que se encuentran en el mundo real y que deben ser representadas en un modelo conceptual de objetos.

Finalmente, con respecto a trabajos de reglas de negocio y objetos, se puede mencionar el trabajo de Diaz et al [Diaz'97]. Esta propuesta considera las reglas de negocio desde un punto de vista de Base De Datos Orientadas a Objetos (con una estructura de evento-condición-acción propia de esta visión). En este sentido ellos extraen las políticas de modelos de especificación de eventos, representados como diagramas de estados, luego que los *Use-Cases*, los objetos y los límites de software han sido definidos. De este trabajo hemos adoptado la noción de estabilidad para aplicarla en nuestra estrategia de identificación de reglas así como también la visión de representar una regla o un conjunto de ellas como un objeto independiente.

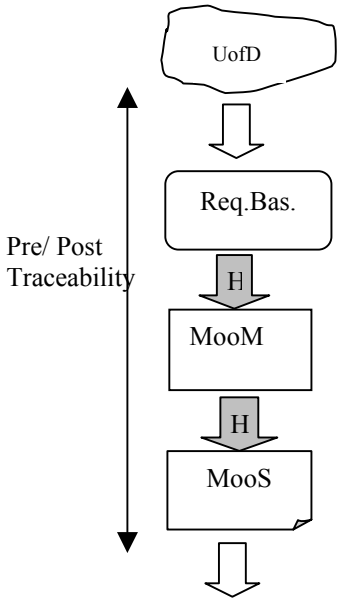
9.2 Contribuciones

Este trabajo de tesis presenta una estrategia de modelado conceptual orientada a objetos que utiliza modelos de requisitos definidos en Lenguaje Natural. La estrategia fue probada en diversos casos de estudio, de los cuales se desarrolló uno en particular, presentado en la sección 8.1, siendo los demás mencionados en la sección 8.2. A partir de estos se determinaron las *ventajas y desventajas* de la misma, también presentadas en la sección 8.3 Como ya se mencionó en el capítulo 6 esta estrategia ha sido adaptada en el

contexto de modelización de requisitos no-funcionales en orientación a objetos [Neto´00]. En esta sección se presentan las principales contribuciones de este trabajo:

- *Definición de una estrategia de definición de un modelo preliminar de objetos que puede ser utilizada en cualquier metodología de Orientación a Objetos como etapa preliminar.*

Desde el paradigma de Orientación a objetos esta tesis permite contar con una estrategia de modelado conceptual que mejora la definición del modelo de objetos, ya sea desde el punto de vista del comportamiento del *UofD* a partir de las CRCs(capítulo 4) como también los aspectos estructurales del mismo (capítulo 5). El modelo obtenido sirve de base para el desarrollo de software orientado a objetos (capítulo 6). En general, como ya se mencionó en la sección anterior, las metodologías de orientación a objetos carecen de etapas de requisitos bien definidas, las más actuales y utilizadas no tienen etapas especificas de requisitos y definen los objetos a partir de los *Use-Cases* o técnicas similares como los *User-Stories* sin utilizar modelos complementarios. Asimismo, el paradigma puro de orientación a objetos no es adecuado para la etapa de modelado conceptual ya que su abstracción principal, los objetos, corresponde más a una abstracción de diseño e implementación que a un concepto de modelado conceptual útil para los clientes [Jacskon'95]. La utilización de los modelos de escenarios, de reglas y LEL permite una mejor comunicación con el cliente porque se acercan a la forma en que él piensa y organiza su trabajo. Esta estrategia si bien utiliza un concepto similar a los *Use-Cases* a través de los escenarios, utiliza otros modelos, el LEL y el modelo de reglas, que permiten dar un mayor grado de abstracción y globalidad a las heurísticas. El conjunto de heurísticas es simple y su objetivo es guiar en la construcción de un modelo de objetos. No son reglas, solo ayudan a manipular la gran cantidad de información generada por los modelos de requisitos. Las heurísticas se basan en cuestiones semánticas más que sintácticas aunque también se hace uso de esta última. En cada etapa de la estrategia se intenta aplicar aspectos de calidad de orientación a objetos que mejoren al modelo conceptual. La aplicación de las heurísticas genera información que permite mejorar la traceability de los modelos(capítulo 7). Esta estrategia puede ser utilizada con cualquier metodología de desarrollo orientada a objetos, como por ejemplo *RUP/UML*. Si bien dentro del proceso de desarrollo de software, esta estrategia consume tiempo y recursos, esto se ve compensado por un mayor conocimiento del Universo de Discurso en donde estará inserto el sistema de software. La figura 9.1 resume la contribución de este trabajo:



Modelos	Beneficios
LEL, escenarios Reglas	- cercano al stakeholder - fácil aprender / usar - fácil validación
Modelos conceptuales OO del Macrosistema	- Ing. software visualiza el Macrosistema en términos de objetos - modelos reusable para distintos sistemas de software del Macrosistema - "puente" entre Requisitos y modelos de Solución
Modelos conceptuales OO del sistema de software	- modelo base para el diseño Orientado a Objetos

Fig. 9.1: Motivación de la estrategia

Desde la perspectiva de Ingeniería de Requisitos, la estrategia permite la aplicación de la *Requirements Baseline*, más concretamente de los modelos de LEL, escenarios y Reglas para el desarrollo de un modelo posterior a esta etapa. De esta forma, modelos que ya han sido probados y aceptados dentro de la comunidad de requisitos, pueden ser utilizados para la definición de un modelo conceptual siguiendo con las heurísticas. La falta de una estrategia de manipulación de estos modelos creaba una brecha entre la *Requirements Baseline y* los modelos de Solución, en este caso un modelo de objetos. Las heurísticas permiten extraer información de una manera más directa y concreta, de esta forma su utilidad se ve ampliada, mas allá del beneficio propio que cualquier modelo de requisitos permite al ingeniero de Software comprender mejor el Universo de Discurso en donde estará inserto el sistema de software que debe desarrollar. Como se menciona en el Capítulo 2, la *Requirements Baseline* es una estructura que contiene modelos orientados al cliente escritos en lenguaje natural, cuyo objetivo es describir al macrosistema, siendo este uno de los principales puntos que aun faltan investigar en el Área de Ingeniería de Requisitos [Zave'97]. Sin embargo, esto crea un modelo orientado al cliente y creemos que desde el punto de vista del proceso de desarrollo de software, es insuficiente. Se necesitan aun más modelos dentro de la etapa de modelización del macrosistema y requisitos. Concordamos con Fowler **[**Fowler'96] cuando afirma que *" la etapa de análisis involucra analizar detrás de los requisitos para formar un modelo conceptual de que está pasando en el problema."* Por esta razón es útil desde el punto de vista de un desarrollo orientado a objetos, realizar un modelo conceptual orientado a objetos, ya que permite al ingeniero visualizar el macrosistema en términos de objetos y responsabilidades como el modelo base para las siguientes etapas del desarrollo de software.

- Definición un modelo de reglas como un modelo complementario de la Requirements Baseline

En esta tesis se presentó un modelo de reglas de Negocio con el objetivo de capturar de manera independiente las políticas de la organización. Las reglas de negocio son parte del corazón de cualquier organización [Gottesdiener'97], por lo tanto es vital su

identificación y modelización en forma explícita. Por esta razón se incorporó una nueva vista que sirve de interfase entre un modelo de la organización y los modelos de *Requirements Baseline*. Este modelo es utilizado junto con el LEL y los escenarios en la estrategia de derivación de objetos, definiendo heurísticas que mejoran el modelo de objetos, ya sea modificando métodos, asociaciones o creando nuevas clases en caso de ser necesario. La incorporación de un modelo de reglas permite que se analice y diseñe una solución que se adapte más fácilmente a los cambios de la organización [Blaze'99]. Como se explicó en la Sección 3.4 del Capítulo 3, es muy importante la inserción de este modelo en la estrategia presentada en esta tesis, por lo cual dicha inserción debe ser profundizada, como se detalla en la siguiente sección.

9.3 Futuro Trabajo

La estrategia presentada en esta tesis se encuentra en un continuo proceso de desarrollo. A partir de su difusión y posterior utilización en diversos casos de estudio se pretende mejorar la estrategia, sobre la base de los resultados y feedback obtenidos. En particular se pretende:

- *Escalar a casos reales*

Esta estrategia fue utilizada en varios casos de estudios reales (capítulo 8) algunos de los cuales han llegado hasta la etapa de implementación. Sin embargo, algunos de estos casos han sido simplificados (como el caso de estudio presentado en esta tesis o el del Sistema de Pasaporte) o perteneciente a un UofD conocido por los investigadores de esta estrategia (por ejemplo la auto-aplicación de esta estrategia o el Sistema de Post-Graduación de Puc-Rio). Sería óptimo probar la estrategia en una organización donde la estructura y el proceso de negocios sean complejos y poco conocido para determinar la efectividad de la estrategia dentro de un proceso de desarrollo de software.

- *Profundizar el estudio del modelo de Reglas de Negocio y su inserción en el modelo de objetos*

Se pretende profundizar la integración del modelo de reglas en el desarrollo de un modelo conceptual de objetos y darle una mayor participación que la que actualmente le da la estrategia en su estado actual. En esta tesis se han propuesto heurísticas para su incorporación en el modelo de objetos. Sin embargo, creemos que así como es útil modelarlas en forma explícita, se deberá estudiar en mayor detalle su posible modelización como objetos independientes o reforzando las relaciones de traceability, para permitir que los modelos de solución se adapten a los cambios de la organización [OMG'97]. De esta forma, la estrategia pretende dar una mayor importancia a este modelo en el contexto del desarrollo de los modelos de objetos, aspecto que todavía no está profundamente estudiado en las metodologías de objetos más difundidas [Gottesdiener'98]. Existen varias estrategias que resaltan la importancia de los objetos de negocio [OMG'97; Ehnebuske'97; Hung'98]. En estos trabajos se definen a los objetos conceptuales y las reglas de negocio relacionadas en un mismo objeto (clase). Creemos que dada la importancia del modelo de reglas en la organización, y que éstas afectan generalmente a más de un objeto, se podría adaptar y profundizar la estrategia propuesta por [Diaz'97] desde el área de Base de Datos Orientadas a Objetos hacia la modelización conceptual. De esta forma bajo ciertas circunstancias, existirán objetos que representen a reglas. Por otro lado, el capturar las reglas de negocio y su modelización sobre los modelos de objetos en casos de estudios de la misma naturaleza puede ayudar a identificar modelos de la solución recurrentes a situaciones particulares, pudiendo definir

patrones de análisis dependientes del dominio [Fowler'96]. Un posterior uso de estos patrones podría permitir la construcción de framework orientados a negocios [Keefer'94].

- *Estudiar impacto de los Patrones de Escenarios en esta estrategia*

Como se mencionó en el capítulo 4, existe una librería de patrones de escenarios que se incorpora en el proceso de construcción de los escenarios mejorando la calidad final de los mismos. En el contexto de esta tesis, se pretende estudiar si existe una relación entre los patrones de escenarios y esquemas básicos de comunicación de objetos, (posiblemente modelados con diagramas de interacción y colaboraciones de *UML*) De la misma forma que existen patrones de análisis independientes del dominio [Fowler'96] que definen aspectos estructurales, se pretende investigar si es posible la definición de patrones de comportamiento de objetos relacionados con los patrones de escenarios.

- *Extensión de BaselineMentor*

Como se mencionó en diferentes capítulos de esta tesis, se ha desarrollado una herramienta llamada *BaselineMentor* [Antonelli´99; Antonelli´00] que implementa parte de la estrategia. Concretamente la herramienta permite:

- Construcción de los modelos de escenarios y LEL
- Derivación automática el modelo de CRCs sin edición siguiendo parte de estas heurísticas
- Navegación entre los modelos de LEL, escenarios y CRCs.
- Manejo de versiones y Traceability de estos modelos

Esta herramienta puede ser extendida para implementar las nuevas heurísticas definidas en esta tesis, en particular se debería incorporar capacidades de edición gráfica para la modelización del modelo lógico de objetos(capítulo 5). También puede incorporarse las relaciones de traceability y su soporte propuestas en el capítulo 7 de esta tesis.

Bibliografía

[Antonelli'99] Antonelli L, Rossi G, Oliveros A "Baseline Mentor, An Application that Derives CRC Cards from Lexicon and Scenarios". Anais de WER'99: Workshop en Requerimientos. 28 JAIOO, SADIO, pp.5-16.

[Antonelli'00] Antonelli L "Traceability en Requirements Baseline y Baseline Mentor Workbench (BMW)" Jornadas sobre ingeniería de requisitos. Universidad De Belgrano, 1 de Diciembre del 2000. http://www.ub.edu.ar/catedras/jor_ing_req/index.htm

[Arsanjani'00] Arsanjani Ali "Design and Implementation of Business Rules in Distributed Object-Oriented Applications" . ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications Minneapolis Convention Center Minneapolis, Minnesota USA October 15-19, 2000 . http://oopsla.acm.org/ap/tutorials/tut59.html

[Ballejos'00] Ballejos L " Módulo de Administración Presupuestaria y Economía Financiera " Universidad Tecnológica Nacional Facultad Regional Santa Fe Proyecto FOMEC SPU Nº 10011)" Jornadas sobre ingeniería de Requisitos. Universidad de Belgrano, 1 de Diciembre del 2000. http://www.ub.edu.ar/catedras/jor_ing_req/index.htm

[Beck'89] Beck K., Cunningham W, "A Laboratory For Teaching Object-Oriented Thinking:" From the OOPSLA'89 Conference Proceedings October 1-6, 1989, pp. 1-6.

[Beck'00] Beck K "Extreme programming explained: embrace change" Addison-Wesley 2000.

[Blaze'99] Blaze Sopftware "Architecture for change: The Rule Engine Propostion" White Paper. www.blazesoft.com.

[Booch'94] Booch G "Object-Orienetd Analysis and desing with Applications" Redwood City, CA: Benjamin/Cummings, 1994.

[Booch'98] Booch G, Rumbaugh J, Jacobson I "The Unified Language User Guide". Addisson Wesley 1998.

[Breitman'98] Breitman K.K, Leite J.C.S.P "A framework for scenario evolution "Proceedings of the Interantional Conference on Requirements Engineering" Colorado Springs, USA –1998, pp. 214-221.

[Buschmann'96] Buschmann Frank et al. "Pattern-Oriented Software Architecture, Volume 1: A System of Patterns " John Wiley & Son Ltd; Vol 1 August 8, 1996.

[Castro'94] Castro R "Script: Uma linguagem de Representação para Modelagem de sistemas de Informação"Tese de Mestrado, Departamento de Informática, PUC-Rio, 1994.

[Coad'95] Coad Peter. "Object models: strategies, patterns and applications" Prentice Hall,1995.

[Cockburn'99] Cockburn Alistair "Responsibility-based Modeling" Technical Memo HaT TR-99.02. http://members/aol.com/humansandt/techniques/responsabilities.htm

[Chidamber'94] Chidamber, Shyam. Kemerer, Chris. "A Metrics Suite for Object Oriented Design". IEEE Trans. on Software Engineering, June 1994, pp. 476-493.

[Dano'97] Dano B, Briand H, Barbier F. "An Approach Based on the Concept of Use Case to produce Dynamic Object-Oriented Specification" Proceedings of IEEE Third International Requirements Engineering Symposium, IEEE Computer Society Press, 1997, IEEE Computer Society Press, pp. 54-64.

[Davis'93] Davis A "Software Requirements: Objects, Functions and States" Englowood cliffs, Nes Jersey: Prentice-Hall.

[Diaz'97] Diaz, O., Iturrioz, J., Piattine, M., "Promoting business policies in object-oriented methods" Sesión Trabajos ya publicados: publicado en The Journal of Systems and Software, 1998. Actas de II Jornadas de Ingeniería de Software, JIS97, Dpto. de Informática , Universidad del país Vasco, San Sebastián, España, 1997, pp. 384-400.

[Diaz'00] Diaz O, Rodriguez J "Change Case analysis" Journal of Object Oriented Programming February 2000, pp.9-15.

[Doorn'98] Doorn J, Kaplan G, Hadad G, Leite JCP "Inspección de escenarios" Anais WER98: Workshop de Engenerhia de Requisitos, Departamento de Informática. Puc-Rio, pp. 57-69.

[Dömges'98] Dömges R, Pohl K "Adapting traceability environments to project-specific needs" Communications of the ACM, December 1998/Vol. 41 No.12, pp. 54-62.

[Ehnebuske'97] Ehnebuske D., Mc KeeB., Rouvellou I., Simmonds I. "Business Objects and Business Rules" Business Object Workshop III ACM OOPSLA'97, Atlanta Usa.

[Elmasri'89] Elmasri and Navathe, S. "Fundamentals of Data Base Systems", Benjamin/Cummings Publishing Comp. Inc, 1989.

[Fiorini'97] Fiorini, S., Leite, J.C.S.P., Macedo-Soares, T., "Integrando Processos de Negocio a Elicitacao de Requisitos" Revista de Informática Teorica e Aplicada, Instituto de Informática da Universidade Federal do Rio Grande do Sul, Vol. IV, N. I, pp. 7-48.

[Fiorini'98] Fiorini S., Leite J.C.S.P., Lucena C, "Organizando Processos de Requisitos" Anais WER98: Workshop de Engenerhia de Requisitos, Departamento de Informática. Puc-Rio, pp. 1-8.

[Fowler'96] Martin Fowler. "Analysis Patterns: reusable object models" Addison-Wesley,1996.

[Fowler'97] Martin Fowler. "New Analysis Patterns" www.martinfowler.com|articles

[Gamma'94] Gamma E., Helm R., Johnson R., Vlissides J. "Design Patterns. Elements of Reusable Object Oriented Software" Addison Wesley, 1994.

[Garcia'00] Garcia O, Gentile C "Escenarios del proceso de construcción de escenarios Autoaplicación de la metodología" Tesis de grado. Depto De Computación y Sistemas, UNCPBA, Argentina.

[Guide'96] Guide Business Rules Project, "Defining Business Rules - What are they are really", www.guide.org/pubs.htm, 1996.

[Gotel'94] Gotel O, Finkelstein A."An analysis of the Requirements Traceability Problem", International Conference on Requirements Engineering,1994, pp.94-101.

[Gottesdiener'97] Gottesdiener "Business RULES Show, Power, Promise" Application Development Trends, vol 4, nro. 3 1997. Extraído de EBG Consulting: http://www.ebgconsulting.com.

[Gottesdiener'98] Gottesdiener E " OO Methodologies: Process & Product Patterns" Component Strategies, Vol. 1,No. 5. November, 1998 Extraído EBG Consulting: de http://www.ebgconsulting.com.

[Hadad'97] Hadad, G., Kaplan, G., Oliveros, A., Leite, J.C.S.P., "Construción de Escenarios a partir del Léxico Extendido del Lenguaje" JAIIO'97, SADIO Buenos Aires, 1997, pp. 65-77.

[Hadad´98] Hadad, G., Kaplan, G., Leite, J.C.S.P "Léxico extendido del lenguaje y escenarios del Meeting Scheduler", Technical Report #13, Departamento de Investigación, Universidad de Belgrano, Buenos Aires, 1998.

[Hadad'99] Hadad et al. "Enfoque middle-out en la construcción e integración de escenarios". Anais WER'99. 28 Jornadas de Informática e Investigación Operativa" 199.9, pp.79-94.

[HendersonSellers'99] Henderson-Sellers B., F.Barbier "What Is This Thing Called Aggregation" Proceedings of Technology of Object-Oriented Languages and Systems Europe'99, Nancy, France, IEEE Computer Society Press, pp. 236-250, June 7-10, 1999.

[Herbst'96] Herbst H."Business Rules in systems analysis: a meta-model and repository system". http:/ ie.iwi.unibe.ch/public/documents/papers.

[Høydalsvik'93] Geir Magne Høydalsvik and Guttorm Sindre "On the purpose of Object-Oriented Analysis" OOPSLA'93 Proceedings", Conference on Object-Oriented Programming Systems, Languages, and applications, Washington. DC. USA, September 1993, pp. 240-255.

[Hung'98] Hung K, Simons t, Rose T. "The Truth Is Out There?: A Survey of Business Object" 5th International Conference on Object-Oriented Information systems (OOIS'98), 9-11 September 1998 at the University of La Sorbonne, Paris, France.

[Jacobson '92] Jacobson I. et al "Object Oriented software engineer: A Use- Case driven approach" Addison Wesley, 1992.

[Jacobson '95] Jacobson I, Ericsson M(Contributor), Jacobson A (Contributor). "The Object Advantage: Business Process Reengineering With Object Technology" Addison-Wesley Object Technology Series. 1995.

[Jacobson'99] Jacobson I Booch G, Rumbaugh J "The Unified Software Development Process" Addison Wesley 1999.

[Jackson 95] Jackson, M. "Software Requirements & Specifications" Addison-Wesley-1995.

[Juristo'98] Juristo N, Morant J, Moreno A "A formal approach for generating OO specifications from Natural language" The Journal of systems and Software. 48 Elsevier. 1998, pp. 139-153.

[Keefer'94] Keefer P, "An object-Oriented framework for accouinting systems" Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 1994. Urbana, Illinois.

[Kotonya'98] Kotonya G, Sommerville I. "Requirements Engineering" J. Wiley and Sons, 1998.

[Leite'95] Leite J.C.S.P, A. Pádua Albuquerque Oliveira. "A Client Oriented Requirements Baseline" Proceedings of the Second IEEE International Symposium on Requirements Engineering , IEEE Computer Society Press, 1995, pp.108-115.

[Leite´96] Leite J.C.S., Oliveros, Rossi, Balaguer, Hadad, Kaplan, Maiorana " Léxico extendido del lenguaje y escenarios del sistema nacional para la obtención de pasaportes", ", Technical Report #7, Departamento de Investigación, Universidad de Belgrano, Buenos Aires, 1996.

[Leite'97] Leite J.C.S, G. Rossi et al. "Enhancing a Requirements Baseline with Scenarios" Proceedings of RE 97': IEEE Third International Requirements Engineering Symposium, IEEE Computer Society Press, 1997, IEEE Computer Society Press, pp. 44-53.

[Leite'97b] Leite J.C.S. "Ingeniería de Requisitos". Notas de Aula. Facultad de Cs. Exactas, UNCPBA, Tandil, Argentina, noviembre de 1997.

[Leite'98] Leite J.C.S.P, Leonardi Ma. Carmen "Business Rules as organizational Policies" IEEE IWSSD9: Ninth International Workshop on Software Specification and Design, IEEE Computer Society Press, 1998, pp. 68-76.

[Leite'00] Leite J.C.S.P, Hadad G, Doorn J, Kaplan G. "A Scenario Construction Process" Requirements Engineering Journal, Springer-Verlag, 2000. Vol.5 N1, pp.38-61.

[Leonardi'96] Leonardi C. "Informe Beca Estudio CIC" Provincia de Buenos Aires.1996

[Leonardi'97] C. Leonardi, V. Maiorana, F. Balaguer "Una Estrategia de Análisis Orientada a Objetos Basada en Escenarios"Actas de II Jornadas de

Ingeniería de Software, JIS97, Dpto de Informática, Universidad del pais Vasco, San Sebastián, España, 1997, pp. 87-100.

[Leonardi'98[a]] Leonardi Carmen, Leite J.C.S.P, Petersen Laura "Integración de un Modelo de Reglas a la definición de Requisitos" Actas de la Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software, Universidad Federal do Rio Grande do Sul, Instituto de Informatica, 1998, pp. 273-285.

[Leonardi'98[b]] Leonardi Carmen, Leonardi Carmen, Leite J.C.S.P, Rossi G "Estrategias para la identificación de Reglas de Negocio" Anais de Sbes98 "Simposio Brasilero de Engenharia de Software" Sociedade Brasilera de Computacao, Maringa, Brasil, 14-16 de Octubre de 1998, pp. 53-67.

[Leonardi'98[c]] Leonardi Carmen, Rossi G, Leite J.C.S.P "Un modelo de hipertexto para la especificación de Requisitos"Anais Workshop de Engenerhia de Requisitos. Departamento de Informática. Puc-Rio 1998. 1998, pp. 119-128.

[Leonardi'00] Leonardi Carmen, Leite J.C.S.P Rossi G, "Un modelo orientado a objetos para el rastreo de requisitos" Actas de las Terceras Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes Software - 2000. Resumen, pp. 492-493.

[Lorenz'94] Lorenz, Mark. Kidd, Jeff. "Object-Oriented Software Metrics: A Practical Guide". Prentice Hall. 1994.

[Mattingly'98] Mattingly L, Rao H "Writing Effective Use Cases and Introducing Collaboration Cases" Journal of Object Oriented Programming. October 1998, pp. 77-84.

[Martin'00] Martin R, "Design Principles and Design Patterns" http://www.objectmentor.com/publications.

[Meyer'97] Meyer Bertrand "Object Oriented Construction" Prentice Hall Object oriented Series.1997.

[Morgan'98] Morgan Gareth. "Imágenes de la Organización" traducción autorizada, Alfaomega Grupo Editor, 1998.

[Neto´00] Neto J. "Integrando Requisitos não Funcionais à Modelagem Orientada a Objetos" Tesis Maestrado. Depto informática. Pontificia Universidade Católica do Rio de Janeiro, 2000.

[OMG'97] Object Managenment Group "Business Object DTF Common Business Objects" OMG Document bom/97-12-04. Version 1.5 1997.

[OOPSLA'97] Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings editado por J Sutherland, D Patel, C Casanave, G Hollowell, J Miller. Springer Verlag; 1997.

[OOPSLA'98] Business Object Design and Implementation : Oopsla '96, Oopsla '97, and Oopsla '98 Workshop Proceedings editado por Workshop on Security for Object-Oriented Systems, D. Patel, J. Sutherland, J. Miller Springer Verlag. 1998.

[OOPSLA'99] Business Object Design and Implementation III : Oopsla '99 Workshop Proceedings editado J. Sutherland. Springer Verlag. 1999.

[OOPSLA'00] Best–practices in Business Rule Design and Implementation" OOPSLA 2000, . ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications Minneapolis Convention Center Minneapolis, Minnesota USA October 15-19, 2000. http://oopsla.acm.org/oopsla2k/fp/workshops/08.html

[Palmer'97] Palmer J.D "Traceability" Software Requirements, 2nd. Edition. Ed. R. H. Thayer and M. Dorfman. IEEE Press, 1997, pp. 364-374.

[Petersen'99] Petersen L., Tornabene S., "HeaR: Una Herramienta de Adquisición de Requisitos". Trabajo de Graduación, Facultad de Cs. Exactas, UNCPBA, Tandil, Argentina, diciembre de 1999.

[Petersen'00] Petersen L. and Tornabene S. Leonardi C, Doorn; "HeaR: Una Herramienta de Adquisición de Requisitos" Anais III Workshop en Engenharia de Requisitos, Rio de Janeiro, Julio del 2000, pp. 38-53.

[Pinheiro'96] Pinheiro Francisco de Asis Cartaxo , "Design of a Hyper-environment for Tracing Object-Oriented Requirements" A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy at the University of Oxford, Trinity Term, 1996.

[Pirotte'99] Pirotte A "Modelización Conceptual"curso de post-grado. Depto. de Sistemas, Fac. de Ciencias Exactas, UNCPBA. 1999. Tandil.

[Pohl'96] Pohl Klaus "Pro-Art: Enabling Requirements Pre-Traceability". Technical Report 96-04. http://www-i5.informatik.rwth-aachen.de/lehrstuhl/publications/index.html

[Potts'94] Colin Potts, Kenji Takahashi, Annie Antón, "Inquiry-Based Requirements Analysis" IEEE Software, March 1994, pp. 21- 32.

[Potts'95] Colin P "Using Schematic Scenarios to Understand User Needs" DIS'95, pp. 247-256.

[Ramesh'98] Ramesh Balasubramanuan "Factors Influencing Requirements Traceability Practice" Communications of the ACM, Dec.1998/Vol.41 No.12, pp.37-44.

[Ridao'00] Ridao M, Doorn J, Leite J "Uso de Patrones en la construcción de escenarios" Anais III Workshop en Engenharia de Requisitos, Rio de Janeiro, Julio del 2000, pp.140-157.

[Rivero'98] Rivero L, Doorn J, del Fresno M, Mauco V, Ridao M, Leonardi C. "Una Estrategia de Análisis Orientada a Objetos basada en Escenarios: Aplicación en un Caso Real" Actas Workshop de Engenerhia de Requisitos, Departamento de Informática. Puc-Rio 1998, pp. 79-88.

[Rolland'98] Rolland C, Ben Achour C, Cauvet C, Ralyté J, Sutcliffe A, Maiden M, Jarke M, Haumer P, Pohl K, Dubois E, Heymans P "A proposal for a Scenario Classification Framework" Requirements Engineering Journal, Vol. 3, No 1 1998, pp 23-47.

[Rosca'97] Rosca, D., Greenspan, S., Feblowitz, M., Wild, C., "A Decision Making Methodology in support of the business rules Lifecycle" Proceedings of RE 97': IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 236-246.

[Ross'80] Ross D "Structured Analysis(SA): a language for Communicating Ideas." Tutorial on design Techniques, Freeman and Wasserman, Eds. IEEE Computer Society Press, Long Beach, Ca 1980, pp.107-125.

[Ross'97] Ross R "The Business rule Book" Database Research Group INC, 1997

[Rossi'96] Rossi Gustavo "An object Oriented Method for Designing Hypermedia Applications" PhD Thesis, Departamento de Informative, PUC-Rio, Brasil, 1996.

[Rosson'99] Rosson M.B "Integrating development of Task and Object models" Commmunciations of The ACM Vol 42 Number 1, pp 49-56.

[Rubin'92] K. Rubin, A.Goldberg "Object Behavior Analysis" Communication of ACM Vol. 35 No. 9 September 1992, pp. 48-60.

[Rumbaugh'91] J. Rumbaugh., M. Blaha et al. "Object-oriented Modeling and Design", Prentice Hall, 1991.

[Rumbaugh'99] Rumbaugh J, Jacobson I, Booch G "The Unified Modelling Language Reference" Manual Addison Wesley 1999.

[Schmauch'95] Schmauch, Ch., "ISO 9000 for software Developers", revised Edition, ASQC, Quality Press, 1995.

[Sommerville'97] Sommerville I, Sawyer P "Requirements Engineering: A good practice guide" J. Wiley and Sons, 1997.

[Warner'99] Warner J , Kleppe. "The Object Constraint Language: Precise Modeling with UML" Addison-Wesley. 1999.

[Wirfs'90] Wirfs-Brock Rebecca, B. Wilkerson, L. Wiener "Designing Object-Oriented Software", Prentice Hall, 1990.

[Wirfs '95] Wirfs-Brock R. "Designing Objects and their interactions" En Scenarios Based Design edit by JohnCarrol. john Willey @ Sons.INC, 1995, pp. 337-359.

[Yu'95] Yu Eric, "Models for Supporting the Redesign of Organizational Work" Proceedings Conference on organizational Computing Systems, August 13-16, 1995 Milipitas, California, USA. Extraído de ftp://ftp.cs.toronto.edu/pub/eric.

[Zave'97] P. Zave, M. Jackson "Four Dark Corners of Requirements Engineering" ACM Transactions on Software Methodology, Vol. 6, N.1,1997, pp. 1-30

[Zachman'96] Zachman John "Enterprise Architecture: The Issue of the Century" extraído de www.zifa.com.

[Zelkowitz'98] Zelkowitz M, Wallace D. "Experimental Models for Validating Technology" IEEE Computer, May 1998, pp.23-31.

[Zorman'95] Zorman L., 1995. "REBUS: Requirements Envisaging By Utilizing Scenarios" A Dissertation presented to the Faculty of the Graduate School University of Southern California, USA. In partial fulfillment of the Requirements for the Degree Doctor of Philosophy (Computer Science).