CLASIFICACIONES DE TIPOS DE REQUISITOS PARA LA MEJORA DEL PROCESO DE DESARROLLO DEL SOFTWARE

Abril 2012

Alumno: Adelaida Ramírez Fernández
Tutora: Anabel Fraga Vázquez

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Objetivo

El estudio sobre un metamodelo de clasificación de tipos de requisitos así como su implementación, descrito en este proyecto, tiene como finalidad, la mejora del proceso de ingeniería del software durante el proceso de desarrollo de software.

El objetivo principal de este proyecto es el desarrollo de un metamodelo que permita el reuso de estructuras organizativas de requisitos. Para aprovecharlo en completitud, sería conveniente, que funcionase como parte de una herramienta case para la gestión de requisitos en un proyecto software.

El metamodelo de clasificación de tipos de requisitos pretende almacenar y reusuar, de manera efectiva, los distintos esquemas de clasificación de requisitos propuestos por los estándares estudiados para mejorar la especificación y gestión de requisitos. Esto conlleva una serie de beneficios tales como:

- Ayuda en la gestión de requisitos.
- Reuso de las estructuras organizativas de requisitos en futuros proyectos.
- Optimización de la herramienta CASE en la que sea incorporada la aplicación desarrollada.
- Mejora en el proceso de desarrollo del software en el campo de la ingeniería de requisitos.

1.2. Metodología

Para el desarrollo del metamodelo de clasificación de tipos de requisitos, en primer lugar, se muestran algunos conceptos básicos de la ingeniería de requisitos, considerada como una de las fases más complejas dentro del proceso del desarrollo del software.

A continuación, se hace un análisis de los principales estándares conocidos: Common Criteria, ESA, IEEE, Métrica y RM-ODP.

De los análisis realizados, se obtienen los distintos tipos de estructuras organizativas propuestos (representación de esquemas de clasificación de tipos de requisitos).

Además, se define e implementa un metamodelo capaz de gestionar todas las estructuras organizativas de los estándares mencionados y otros generados.

La definición del metamodelo se realiza mediante un diagrama de clases, el cual, se compone de las clases tipo (generalización de los subtipos de requisitos), subtipo (posibles tipos que son subtipos de otro tipo), relación (relación entre tipos) y clasificación (estándar para clasificación de requisitos).

La implementación de la aplicación, capaz de gestionar el metamodelo definido, se desarrolla en C# bajo la plataforma .NET.

Antes de la fase de codificación, es necesaria la descripción previa de los casos de uso que definen la funcionalidad del programa, una explicación de los aspectos de programación más relevantes y un diagrama de clases que describe la estructura de la aplicación.

En definitiva, con el metamodelo desarrollado, se pretende almacenar y reusar todas las estructuras organizativas dependiendo de las necesidades de cada proyecto software sobre el que se aplique. Esto conlleva una serie de beneficios tales como el aumento en la calidad del software, mejoras en el mantenimiento del mismo, reducción de costes, etc.

2. INGENIERÍA DE REQUISITOS

La ingeniería de requisitos, según Pressman [Pressman 05], es un conjunto de procesos, tareas y técnicas que permiten la definición y gestión de los requisitos de un producto de un modo sistemático. En definitiva, facilita los mecanismos adecuados para comprender las necesidades del cliente, analizando sus necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional.

La ingeniería de requisitos permite la gestión adecuada de los requisitos de un proyecto de desarrollo software. Además, mejora la capacidad para realizar planificaciones de los procesos de proyectos de desarrollo software puesto que el conocer qué se tiene que desarrollar permite una efectiva proyección de las actividades, recursos, costos, tiempos, etc. del proyecto. Según Sommerville [Sommerville 05], se puede considerar como el proceso de comunicación entre los clientes, los usuarios del software y los desarrolladores del mismo.

Llevar a cabo de manera adecuada el proceso de ingeniería de requisitos disminuye la probabilidad de fracaso de un proyecto. Esto es porque los requisitos bien definidos permiten conocer de un modo conciso que debe ser capaz de realizar el software a desarrollar y hace orientar las actividades, recursos y esfuerzos de manera eficiente permitiendo la disminución de costos y retrasos.

Todo ello provoca una mejora en la calidad del software puesto que los requisitos bien especificados podrán probarse de manera efectiva y, por tanto, cumplirse.

2.1. Fases en ingeniería de requisitos

La definición de las necesidades del sistema juega un papel fundamental en el proceso de desarrollo del software (SPD). Esta definición es un proceso complejo puesto que en él hay que identificar los requisitos que el

sistema debe cumplir para satisfacer las necesidades. Para realizar este proceso, no existe una única técnica estandarizada y estructurada que ofrezca un marco de desarrollo que garantice la calidad del resultado.

La delimitación de ingeniería de requisitos no es del todo clara puesto que incluso, hay autores que, dentro de la ingeniería de requisitos, generan modelos estáticos de clases que son entendidos por otros autores como tareas de una fase posterior.

Dentro de las posibles estructuras que se pueden definir en la fase de ingeniería de requisitos, la propuesta con mayor seguimiento es la establecida por Lowe y Hall [Lowe y Hall 99].

En ella, el proceso de tratamiento de requisitos está compuesto por tres actividades:

- **Captura** de requisitos.
- **Definición** de requisitos.
- **Validación** de requisitos.

Se puede representar este proceso de ingeniería de requisitos como un diagrama de actividades en UML, tal como se muestra en la ilustración 1.

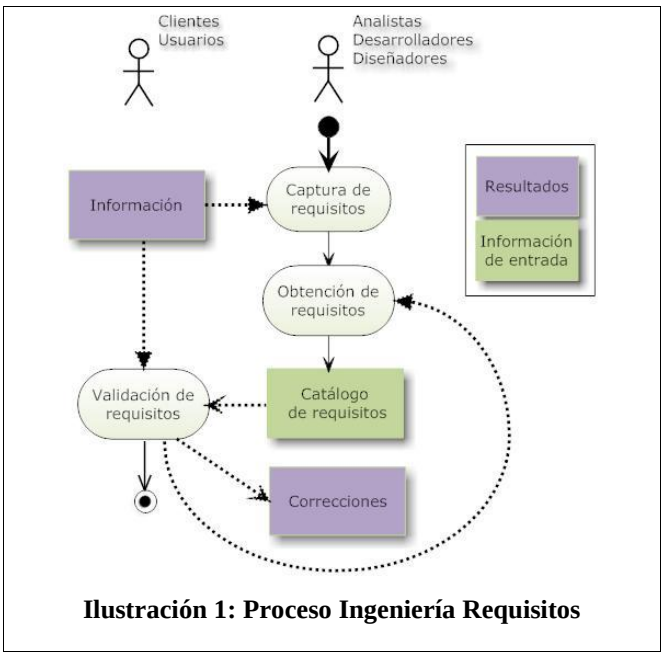

Ilustración 1: Proceso Ingeniería Requisitos

Es posible plantear el estudio de viabilidad y la gestión de requisitos en el proceso de tratamiento de requisitos.

El estudio de viabilidad permite definir de modo global la funcionalidad y ver si es factible su ejecución dentro del aspecto económico y del tiempo establecido, elaborando para ello un informe de viabilidad.

La gestión de requisitos permite las posibles actualizaciones y cambios que pueden sufrir los requisitos, dando lugar a versiones del documento de requisitos. Esta actividad se desarrolla a lo largo de todo el proceso de desarrollo software.

2.1.1. Captura de requisitos

La captura de requisitos es la actividad en la que un grupo especializado extrae, de cualquier fuente de información disponible (documentos, aplicaciones existentes, entrevistas, etc.), las necesidades de cubrir dicho sistema. El proceso de captura de requisitos puede resultar complejo, debido a esto existen un conjunto de técnicas que permiten hacer este proceso de una forma más eficiente y precisa, obteniéndose necesidades y modelos del sistema.

A continuación se enumeran un grupo de técnicas que son utilizadas para esta actividad:

- **Entrevistas**. Permiten tomar conocimiento del problema y comprender los objetivos de la solución buscada.
- **Desarrollo de conjunto de aplicaciones**. Es una práctica de grupo donde participan usuarios, analistas, administradores del sistema, y clientes y en la que se concretan las necesidades del sistema.
- **Tormenta de ideas**. Consiste en la mera acumulación de ideas sin evaluar las mismas. Ofrece una visión general de las necesidades del sistema pero sin ofrecer detalles concretos.
- **Mapa conceptual**. Grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos sirven para aclarar los conceptos relacionados con el sistema a desarrollar, ofreciendo una visión general de las necesidades del sistema.

- **Casos de uso**. Muestran el contorno (actores) y el alcance del sistema (requisitos expresados como casos de uso). Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. La ventaja principal de los casos de uso es que resultan muy fáciles de entender para el cliente, sin embargo pueden carecer de a precisión necesaria, es por ello que pueden acompañarse de una información textual.
- **Cuestionarios**. Recoge información del sistema de forma independiente de la entrevista.

2.1.2. Definición de requisitos

Para la definición de los requisitos son usados lenguajes naturales que, a pesar de poder ser ambiguos y/o extensos, su coste reducido hace que sean la alternativa común a los lenguajes formales.

Es necesario elaborar el documento de requisitos, en el cual, se definen los objetivos y necesidades del sistema. Existe un conjunto de técnicas propuestas para esta actividad:

- **Lenguajes naturales y formales**. Los lenguajes naturales es una técnica muy ambigua en contraposición a los lenguajes formales.
- **Ontologías**. Donde se definen los conceptos que intervienen en la definición del sistema así como las relaciones que existen entre ellos.
- **Patrones**. Define los requisitos mediante el lenguaje natural pero de una manera estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida.

Es necesario clasificar el conjunto de requisitos, pudiendo generarse un catálogo distinto dependiendo del estándar utilizado.

2.1.3. Validación de requisitos

Por último, se procede a la validación de requisitos, llevándose a cabo la valoración de los mismos, comprobando la veracidad, consistencia y completitud de requisitos. Las técnicas existentes para desarrollar esta actividad son:

- **Revisiones de verificabilidad**. Consiste en la lectura y corrección de la documentación o modelado de la definición de requisitos.
- **Auditorías**. Consiste en un chequeo de los resultados.
- **Matrices de trazabilidad**. Consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo que objetivos cubre cada requisito para detectar inconsistencias u objetivos no cubiertos.
- **Prototipos**. Permite que el usuario se haga una idea del sistema.

Los requisitos han de ser gestionados mediante el llamado plan de gestión de requisitos que establece las medidas, el método, soporte y técnicas para almacenar los requisitos, gestionar el cambio de éstos (debido a que pueden ser volátiles) y gestionar su trazabilidad.

2.2. Evolución y negociación de los requisitos

Desde el principio, se pretende especificar los requisitos de forma detallada pero es casi imposible conocer al principio del proyecto todos los detalles del futuro software. En la práctica, los requisitos son cambiantes dentro del proceso iterativo y evolutivo del software. Pueden existir muchos problemas en los proyectos software por la volatibilidad de los requisitos, muchos clientes desean que se añadan o modifiquen nuevos requisitos por el mismo precio y es tarea del ingeniero negociar esas nuevas funcionalidades no presentadas en el pliego inicial. Es por esto que para llevar con éxito un proyecto, es necesario el contacto continuado entre el cliente y el ingeniero software para conocer las

dificultades y necesidades de ambas partes y así buscar una solución de manera conjunta.

2.3. Necesidad del establecimiento y gestión de requisitos

La necesidad de modelar los requisitos dado un sistema tiene una importancia vital puesto que sin éstos el equipo no sabes cuales son las metas a lograr, ni puede inspeccionar y probar su trabajo de modo adecuado, ni puede controlar su productividad, ni puede obtener datos que se adecuen a las pruebas, ni pueden predecir el tamaño y esfuerzo del proyecto. En definitiva, no hay ingeniería completa sin requisitos escritos.

Muchos proyectos iniciados no llegan a finalizarse o aquellos que son terminados, incurren en mayores costes y tiempos o no incluyen la totalidad de los requisitos iniciales. Una de las causas de este problema es la deficiente captura y gestión de los requisitos.

Otro punto importante de la ingeniería de requisitos es la adecuada gestión de los requisitos que se puede considerar como el factor más relevante para el éxito o el fracaso de un proyecto. Esto es motivado por lo cambiante que son los requisitos.

El plan de gestión de requisitos define las medidas, métodos, soportes y técnicas para almacenar los requisitos, gestionar el posible cambio de éstos así como su trazabilidad.

La gestión de requisitos no es una tarea sencilla, Chiristel y Kang [Christel y Kang 92], plantean una serie de inconvenientes:

- Problemas de alcance. Esto es debido a que es complejo especificar los límites que abarcan los requisitos.
- Problemas de comprensión. Esto es debido a que los clientes y usuarios no suelen estar completamente seguros de lo que quieren.
- Problemas de volatibilidad. Esto es debido a que los requisitos pueden cambiar con el tiempo puesto que durante el proceso mismo del desarrollo los requisitos evolucionan.

2.4. Documentación

La documentación es el propio resultado del proceso de la ingeniería de requisitos y, por tanto, se convierte en unos de los activos del proceso del desarrollo software. Las herramientas existentes en el mercado para la gestión de requisitos generan los documentos así como las versiones evolutivas de requisitos para permitir la gestión de los mismos.

3. ESTÁNDARES DE REQUISITOS

A continuación, se analizan los principales estándares (Common Criteria, Métrica, ESA, IEEE, RM-ODP) que son fuentes necesarias para la implementación del metamodelo.

El estudio se fundamenta en la clasificación de requisitos que establece cada uno de dichos estándares.

3.1. Criterios para la evaluación de seguridad de tecnologías de la información (CC: Common Criteria for information technology security evaluation)

Common Criteria [Common Criteria 08] es un estándar de seguridad, que garantiza que los productos de Tecnologías de la Información cumplen con estrictos requisitos de seguridad.

En concreto, Common Criteria regula la evaluación objetiva, repetible y comparable de las propiedades de la seguridad para todo tipo de productos y sistemas de información, es decir, permite asegurar que el proceso de especificación, desarrollo y evaluación de un producto relacionado con la seguridad de la información ha sido ejecutado de forma consistente. Permitiendo evaluar tanto productos como sistemas relacionados con tecnologías de la información, incluyéndose sistemas operativos, redes, aplicaciones, etc.

Concretamente, la filosofía de CC es ofrecer grados de confianza en la seguridad de un producto o sistema de información.

3.1.1. Clasificación de requisitos y conceptos básicos de Common Criteria

3.1.1.1. Clasificación de requisitos

Se distinguen dos tipos de requisitos:

- **Requisitos funcionales de seguridad**. Definen el comportamiento de seguridad ante amenazas y el catálogo de

requisitos funcionales estándar de seguridad para los objetos de evaluación.

- **Requisitos de garantía de seguridad**. Describen el conjunto de requisitos de garantía estándar para garantizar la seguridad de los objetos de evaluación. Además se definen criterios de evaluación para perfiles de protección y para los objetivos de seguridad.

3.1.1.2. Conceptos básicos

- **Objetos de evaluación**. Parte del producto o sistema que cuenta con recursos (almacenamiento electrónico, periféricos, capacidad de computación) que es usada para procesar y almacenar información y que es sujeta de evaluación.
- **Perfil de protección**. Identifica los requisitos de seguridad relevantes a los usuarios para un propósito determinado donde se define clases de mecanismos de seguridad. Es decir, implementa de manera independiente requisitos de seguridad para dirigir amenazas que existen en un entorno específico. Estos perfiles de protección son necesarios cuando se ajusta el estándar a un tipo particular del producto.
- **Objetivos de seguridad.** Identifica las propiedades de seguridad del objeto de evaluación. Está formado por las amenazas de los objetos de evaluación, los requisitos de seguridad y un índice con la especificación de las funciones de seguridad y las medidas de garantía.
- **Entorno de seguridad**. Al igual que se establecen los requisitos, se define un entorno de seguridad donde el objetivo de evaluación será usado, documentando las amenazas y riesgos que pueden darse, los supuestos y las políticas de seguridad que se deben cumplir. También se establecen objetivos a conseguir, en concordancia a las políticas de seguridad para contrarrestar todas aquellas amenazas identificadas.

El conjunto de requisitos si son logrados, garantizan que el objeto de evaluación pueda cumplir sus objetivos de seguridad.

3.1.2. Organización jerárquica de requisitos

Los dos tipos de requisitos existentes, funcionales y de garantía, se organizan jerárquicamente en componentes, familias y clases.

- **Componente**. Conjunto específico de requisitos. Es el elemento más pequeño dentro de la jerarquía. Los componentes son construcciones de elementos. El elemento es el requisito de seguridad indivisible que puede ser verificado para su evaluación.
- **Familia**. Nivel intermedio de agrupamiento. Las familias comparten objetivos de seguridad pero pueden diferenciarse por el rigor de éstos. Las familias están compuestas por componentes.
- **Clase**. Nivel más genérico para agrupar requisitos, compuesto por familias que comparten un enfoque común aunque se diferencian en el alcance de los objetivos de seguridad. Las clases están compuestas por familias.

Existen paquetes que se componen de requisitos que cumplen un conjunto identificables de objetivos de seguridad, con la intención de ser reutilizables.

3.1.3. Explicación de clasificación de requisitos

3.1.4. Requisitos funcionales de seguridad

Los requisitos funcionales de seguridad describen el comportamiento de seguridad deseado por los objetos de evaluación, que debe tener el sistema de objeto de evaluación, expresados en perfiles de protección o con objetivos de seguridad. En concreto, este tipo de requisitos debe describir propiedades de seguridad que los usuarios pueden detectar por interacción directa con el sistema o por respuesta de éste a una serie de estímulos. Para los consumidores, estos requisitos, son usados como una

guía para el uso de las funciones de seguridad, sin embargo, los desarrolladores los usan para interpretar tales funciones de seguridad.

A continuación, se detallan una serie de clases, las cuales, agrupan familias que, a su vez, agrupan componentes que expresan los requisitos funcionales de seguridad:

- **Auditorías de seguridad**. Implican reconocimiento, registro, almacenamiento y análisis de la información relacionada con las actividades de seguridad. El resultado de estas auditorías puede ser examinado para determinar las actividades de seguridad relevantes que se dan lugar así como los responsables de las mismas. Las familias que pueden alojarse en esta clase:
    - Respuesta automática de auditorías de seguridad. Define las respuestas a tomar en el caso de detectar eventos que indiquen una violación en la seguridad.
    - Generación de datos de auditorías de seguridad. Son necesarios para registrar los eventos relevantes de seguridad que toman lugar bajo el control de las funciones de seguridad del propio objeto de evaluación.
- **Soporte criptográfico**. Es utilizado cuando el objeto de evaluación implementa funciones criptográficas para satisfacer diferentes objetivos de seguridad de mayor nivel, por ejemplo, aspectos relacionados con las identificaciones y autenticaciones, el no repudio, la separación de datos, etc. Estas implementaciones criptográficas pueden ser sobre el hardware, firmware o software. Las familias que pueden alojarse en esta clase:
    - Gestión de claves criptográficas. Describe el ciclo de vida de estas claves (generación, distribución, acceso y destrucción).
    - Operaciones criptográficas. Definen las operaciones que son ejecutadas mediante un algoritmo determinado junto con la clave criptográfica.
- **Comunicaciones**. Asegura la identidad de la parte que participa en intercambio de datos, tanto la identidad del origen que transmite la

información como quién la recibe. El origen no puede negar el envío del mensaje ni el destinatario negar el recibo de la misma, es decir, no debe existir repudio por ninguna de las partes. Las familias que pueden alojarse en esta clase:

- No repudio en el origen.
- No repudio en el destino.

- **Protección de datos de usuario**. Define aspectos durante la importación, exportación y almacenamiento de los datos, además de los atributos de la seguridad relacionados con la protección de los datos de usuario. Se especifican requisitos para las funciones de seguridad del objeto de evaluación así como políticas para proteger datos de usuario. Las familias que pueden alojarse en esta clase:
  - Políticas de control de acceso. Describen el alcance de las políticas que componen el control de acceso del objeto de evaluación.
  - Autenticación de datos. Provee de métodos para garantizar la validez de datos específicos y así verificar que no han sido fraudulentamente modificados.

- **Identificación y autenticación**. Asegura que los usuarios sean asociados con una serie de atributos de seguridad (por ejemplo, identidad, grupos, roles, seguridad o niveles de integridad). Necesario para la identificación no ambigua de usuarios autorizados y la correcta asociación de éstos con los atributos de seguridad. Una identificación no ambigua de usuarios autorizados es crítica para cumplir una correcta política de seguridad. Las familias que pueden alojarse en esta clase:
  - Fallos en la autenticación. Definen valores para situaciones de autenticaciones fallidas.
  - Definición de atributos de usuario que asocia los atributos de seguridad con los usuarios.

- **Gestión de seguridad**. Especifica la gestión de los datos, atributos y funciones de seguridad del objeto de evaluación, así como diferentes

tipos de roles y su interacción. Las familias que pueden alojarse en esta clase:

- Gestión de los datos de las funciones de seguridad del objeto de evaluación. Permite el control a los usuarios autorizados sobre la gestión de las funciones de seguridad del objeto de evaluación.
- Gestión de los atributos de seguridad. Permite el control a los usuarios autorizados sobre la gestión de los atributos de seguridad, etc.

- **Privacidad**. Los requisitos de privacidad proveen al usuario de una protección contra el uso indebido de su identidad (por parte de otros usuarios). Las familias que pueden alojarse en esta clase:
  - Anonimato. Provee de protección a la identidad del usuario.
  - Pseudoanonimato. Asegura el uso, por parte del usuario, de un recurso o servicio sin descubrir la identidad del mismo, etc.
- **Protección de las funciones del objeto de evaluación**. Define la protección de los datos de las funciones de seguridad del objeto de evaluación. Se definen los requisitos que relacionan la integridad y gestión de los mecanismos que proveen las funciones del objeto de evaluación y la integridad de los datos de los mismos. Las familias que pueden alojarse en esta clase:
  - confidencialidad de datos exportados de las funciones de seguridad del objeto de evaluación.
  - integridad de los datos exportados de las funciones de seguridad del objeto de evaluación.
- **Utilidad de recursos**. Provee la disponibilidad de recursos requeridos, por ejemplo, habilidad de procesamiento, capacidad de almacenamiento, tolerancia a fallos, prioridad de servicio, etc. Las familias que pueden alojarse en esta clase:
  - Tolerancia a fallos. Provee protección contra capacidades no disponibles causadas por un fallo del objeto de evaluación.

- Prioridad en el servicio. Asegura que los recursos van a ser localizados para las tareas más críticas y no deben ser monopolizados por tareas de baja prioridad.

- **Acceso al objeto de evaluación**. Cuenta con requisitos funcionales específicos para la identificación, autenticación y control del establecimiento de sesión por parte del usuario. Estableciendo limitación para sesiones concurrentes, limitación en el acceso de ciertos atributos, etc. Las familias que pueden alojarse en esta clase:

  - Limitación en sesiones concurrentes. Define el límite en el número de sesiones concurrentes para un mismo usuario.
  - Sesiones cerrojo. Proveen sesiones con cerrojo o sin él.

- **Canales de confianza**. Establece canales de comunicación de confianza entre el usuario y las funciones de seguridad del objeto de evaluación. Estos caminos de confianza tienen una serie de características: son construidos usando canales de comunicación internos y externos que aíslan un subconjunto identificado de datos y comandos, su uso puede ser iniciado por cualquiera de los dos lados del canal, asegurando que el usuario está comunicando con la correcta función de seguridad del objeto de evaluación y garantiza el no repudio con respecto a la identidad de ambos lados. Las familias que pueden alojarse en esta clase:

  - Canales de confianza.
  - Caminos de confianza.

3.1.4.1. Requisitos de garantía de seguridad

Los requisitos de garantía de seguridad definen la confianza en la exactitud de la implementación y en la efectividad de las funciones de seguridad alegadas. En concreto, definen un criterio de evaluación para perfiles de protección y objetivos de seguridad y presentan los denominados niveles de evaluación de garantía. Para los consumidores, estos requisitos, son usados como una guía para determinar los niveles

de garantía requeridos, sin embargo, los desarrolladores los usan para interpretar tales niveles de garantía.

A continuación, se detallan una serie de clases, las cuales, agrupan familias que, a su vez, agrupan componentes que expresan los requisitos de garantía de seguridad:

- **Gestión de configuración**. Define la integridad del objeto de evaluación mediante la disciplina y el control en los procesos de refinamiento y modificación del mismo. Las familias que pueden alojarse en esta clase:
    - Automatización de la gestión de configuración. Establece el nivel en el uso de automatización para el control de configuración.
    - Habilidad en la gestión de configuración. Define las características del sistema de gestión de configuración.
- **Distribución y operaciones**. Define la medida, procedimientos y estándares para la distribución de seguridad, así como instalación y uso operacional del objeto de evaluación. Las familias que pueden alojarse en esta clase :
    - Distribución. Cubre los procedimientos usados para mantener la seguridad durante la transferencia del objeto de evaluación al usuario.
    - Instalación, generación e inicio. Provee, al administrador, de confidencialidad los parámetros de configuración del objeto de evaluación.
- **Desarrollo**. Define el refinamiento de las funciones de seguridad del objeto de evaluación, así como un mapeo de los requisitos de seguridad gracias a una representación al más bajo nivel. Las familias que pueden alojarse en esta clase:
    - Especificación funcional. Describe las funciones de seguridad del objeto de evaluación y deben ser una instanciación exacta de los requisitos funcionales de seguridad.

- Diseño a alto nivel. Refina la especificación funcional de las funciones de seguridad.

- **Documentos de apoyo**. Describe el uso de operaciones seguras tanto por usuario como por administradores. Las familias que pueden alojarse en esta clase:
  - Guía de administrador.
  - Guía de usuario.

- **Soporte del ciclo de vida**. Incluye la definición de un modelo de ciclo de vida (incluyendo los pasos de desarrollo del objeto de evaluación), herramientas, políticas, técnicas y medidas de seguridad usadas para proteger el entorno de desarrollo. Las familias que pueden alojarse en esta clase:
  - Seguridad de desarrollo. Cubre las medidas de seguridad, personales, físicas, etc usadas en el ambiente de desarrollo.
  - Los remedios de defecto. Aseguran que los defectos descubiertos por los usuarios del objeto de evaluación son seguidos y corregidos mientras que el objeto de evaluación es respaldado por el administrador.

- **Tests**. Demuestra que las funciones de seguridad satisfacen los requisitos funcionales de seguridad del objeto de evaluación. Las familias que pueden alojarse en esta clase:
  - Cobertura. Define que funciones son testada.
  - Profundidad. Define el nivel de detalle de los tests desarrollados para el objeto de evaluación.

- **Valoración de vulnerabilidad**. Incluye requisitos que identifican la vulnerabilidad en la explotación, es decir, las que aparecen en la construcción, operación, mal uso o la configuración incorrecta del objeto de evaluación. Las familias que pueden alojarse en esta clase:
  - Análisis de canales secretos. Analiza el descubrimiento y el análisis de canales de comunicaciones explotados para violar las políticas de seguridad del objeto de evaluación.

- Uso incorrecto. Analiza si el objeto de evaluación puede ser configurado o usado de un modo inseguro.

3.1.4.2. Niveles de evaluación de garantía de seguridad

Para los requisitos de garantía de seguridad, se establecen niveles de evaluación de garantía que definen una escala para medir la garantía de los objetos de evaluación así como un criterio de evaluación de los perfiles de protección y los objetivos de seguridad.

En concreto, se define una escala que sopesa el nivel de garantía obtenido mediante el coste y la viabilidad del grado de garantía adquirido. Un alto nivel de evaluación de garantía no implica necesariamente una mejor seguridad, sólo quiere decir que la garantía de seguridad alegada ha sido validada más extensamente. Se definen siete niveles de evaluación.

- **Nivel 1**: Testado funcionalmente. En concreto, provee de un nivel básico de garantía, es aplicable cuando un bajo nivel de confianza es requerido en ciertas operaciones y las amenazas a la seguridad no son consideradas como serias.

- **Nivel 2**: Testado estructuralmente. En concreto, requiere un testeo del desarrollo, análisis de vulnerabilidades y un testeo independiente basado en detalles de la especificación del objeto de evaluación.

- **Nivel 3**: Testado y chequeado metódicamente. En concreto, es aplicable cuando desarrolladores o usuarios requieren un nivel moderado para asegurar la seguridad.

- **Nivel 4**: Diseñado metódicamente, Testado y revisado. En concreto, provee garantía gracias al uso de controles del entorno de desarrollo y una gestión de la configuración del objeto de evaluación mediante la automatización.

- **Nivel 5**: Testado y diseñado semiformalmente. En concreto, provee de descripciones de diseño semiformales, una implementación completa, una arquitectura estructurada y un análisis de los canales secretos.

- **Nivel 6:** Testado, diseñado y verificado semiformalmente. En concreto, es aplicable cuando el desarrollo de seguridad del objeto de evaluación es denotado como una situación de riesgo alto, lo que justifica costes adicionales. Provee de un análisis comprensivo, una representación estructurada de la implementación, una arquitectura estructurada, análisis independientes de vulnerabilidades, identificación sistemática de canales secretos, gestión de la configuración y controles en el entorno de desarrollo.
- **Nivel 7**: Testado, diseñado y verificado formalmente. En concreto, es aplicable cuando el desarrollo de seguridad del objeto de evaluación es denotado como una situación de riesgo extremadamente alto, lo que justifica altos costes adicionales. Provee de un proceso de desarrollo estructurado, controles en el entorno de desarrollo, gestión en la configuración del objeto de evaluación, todo ello gracias al uso de representaciones y correspondencias formales y un exhaustivo testeo.

3.2. Métrica

Métrica versión 3 [Métrica versión 3 08] es una Metodología de Planificación, Desarrollo y Mantenimiento de Sistemas de información. Esta metodología está basada en el Modelo de Procesos del Ciclo de vida de desarrollo (ISO/IEC 12207) así como en la norma ISO/IEC 15504 SPICE.

El objetivo del estándar es establecer un conjunto de tareas a realizar, técnicas y productos a obtener para desarrollar sistemas de información con una mayor calidad, productividad y satisfacción de los usuarios para facilitar su mantenimiento posterior.

3.2.1. Clasificación de requisitos y conceptos básicos de Métrica

3.2.1.1. Clasificación de requisitos

Se distinguen cinco tipos de requisitos:

- **Requisitos funcionales.** Representan funcionalidades que el sistema debe cubrir, mediante la descripción de casos de uso en los que los diferentes actores utilizan los diferentes servicios proporcionados por el sistema. Cada requisito funcional llega a identificarse con el evento de activación, las pre y post condiciones, así como los pasos que componen el caso de uso junto con sus excepciones.

- **Requisitos de rendimiento.** Definen las limitaciones de funcionamiento del software. Especifica aspectos tales como la velocidad de operaciones, rangos de precisión, uso de memoria, etc.

- **Requisitos de seguridad.** Definen los criterios de seguridad para el software. Especifica aspectos tales como restricciones de acceso, confidencialidad, integridad, etc.

- **Requisitos de implantación.** Relacionados con la formación, infraestructura e instalación para preparar y organizar los recursos necesarios para la implantación e instalación del sistema.

- **Requisitos de disponibilidad del sistema.** Definen los niveles de disponibilidad del software, el uso del sistema cuando esté operativo, su capacidad mínima y media disponible, etc.

3.2.1.2. Conceptos básicos

La metodología, utilizada en métrica, descompone el desarrollo completo del sistema software en procesos, éstos en actividades y éstas en tareas.

Los procesos principales son **planificación, desarrollo y mantenimiento** del sistema de información.

En el análisis del sistema de información (perteneciente al proceso de desarrollo) se suceden una serie de pasos claves en la generación de los requisitos.

En la actividad de definición del sistema, se genera un catálogo de requisitos generales a partir del catálogo de requisitos generado en el estudio de viabilidad del sistema.

En la actividad de **establecimiento de requisitos** se lleva a cabo la definición, análisis y validación de los requisitos a partir de la información

facilitada por el usuario. El objetivo de esta actividad es obtener un catálogo detallado de los requisitos, a partir del cual se pueda comprobar que los productos generados en las actividades de modelización se ajustan a los requisitos de usuario.

Esta actividad se descompone en el siguiente conjunto de tareas:

- **Obtención de requisitos**. Se llevan a cabo sesiones de trabajo con los usuarios para definir qué es lo que debe cumplir el software. También se definen las prioridades que hay que asignar a los requisitos, considerando los criterios del usuario acerca de las funcionalidades a cubrir.

- **Especificación de casos de uso**.

- **Análisis de requisitos**. Donde se estudia la información capturada previamente para detectar inconsistencias, duplicidad o escasez de información. Además se analizan las prioridades establecidas por el usuario y se asocian los requisitos relacionados entre sí. El análisis de los requisitos y de los casos de uso asociados permite identificar funcionalidades o comportamientos comunes.

- **Validación de requisitos**. Donde se confirma que los requisitos especificados en el catálogo de requisitos, así como los casos de uso, son válidos, consistentes y completos.

En el diseño del sistema de información, se suceden una serie de pasos claves en la generación de los requisitos.

En la actividad de definición de la arquitectura del sistema. Donde se identifican los requisitos de diseño y construcción así como los requisitos de operación y seguridad.

En la actividad de establecimiento de requisitos de implantación. Donde se establecen los requisitos relacionados con la documentación que el usuario requiere para operar con el nuevo sistema y los relativos a la propia implantación del sistema en el entorno de operación. La incorporación de estos requisitos permite ir preparando los medios y recursos necesarios para que los usuarios, tanto finales como de

operación, sean capaces de utilizar el nuevo sistema de forma satisfactoria. En esta actividad aparecen dos tareas importantes, la de especificación de requisitos de documentación de usuario y la de especificación de requisitos de implantación.

3.2.2. Explicación de clasificación de requisitos

3.2.2.1. Requisitos funcionales

Los requisitos funcionales representan funcionalidades que el sistema debe cubrir. En concreto, describen los casos de uso en los que los diferentes actores utilizan los diferentes servicios proporcionados por el sistema. Cada requisito funcional llega a identificarse con un caso de uso donde se especifica la información relativa a:

- Descripción del escenario, es decir, cómo un actor interactúa con el sistema y cuál es la respuesta obtenida.
- Precondiciones y postcondiciones.
- Identificación de interfaces de usuario.
- Condiciones de fallo que afectan al escenario, así como la respuesta del sistema (escenarios secundarios).

3.2.2.2. Requisitos de rendimiento

Los requisitos de rendimiento definen las limitaciones del funcionamiento del software. Determinan los límites en el rendimiento (para aquellas aplicaciones donde existan) y los volúmenes de información que el software debe tratar.

3.2.2.3. Requisitos de seguridad

Los requisitos de seguridad se definen a partir de la arquitectura propuesta y las características del entorno tecnológico. Necesarios para garantizar la protección del sistema y minimizar el riesgo de pérdida, alteración o consulta indebida de la información. Los requisitos de operación son establecidos para los distintos elementos del sistema (módulos, clases, estructuras físicas de datos, sistemas de ficheros, etc).

3.2.2.4. Requisitos de implantación

Los requisitos de implantación son aquellos relacionados con la formación, infraestructura e instalación para preparar y organizar los recursos necesarios para la implantación e instalación del sistema. Además, teniendo en cuenta las particularidades del sistema de información, se determinan los conocimientos o aptitudes adicionales que requieren los usuarios finales para operar con el nuevo sistema

3.2.2.5. Requisitos de disponibilidad

Los requisitos de disponibilidad definen los niveles de disponibilidad del software. Por ejemplo, que la arquitectura del sistema sea tal que se maximice la alta disponibilidad del sistema, de tal forma que no existan puntos singulares de fallo en la misma.

3.3. Agencia espacial europea (ESA: European Space Agency)

Los estándares de ingeniería del software de ESA PSS-05-0 [ESA PPS-05-0 08] definen las prácticas de software que deben aplicarse en los proyectos de la agencia espacial europea.

Este estándar concierne todos los aspectos software del sistema, incluyendo sus interfaces, el hardware y resto de componentes. El software, definido en este estándar, son programas, procedimientos, reglas y toda la documentación asociada perteneciente a la operación de un sistema computarizado.

3.3.1. Clasificación de requisitos y conceptos básicos de ESA

3.3.1.1. Clasificación de requisitos

Se distinguen dos tipos de requisitos:

- **Requisitos de usuario**. Componen el proceso de organizar la información sobre las necesidades del usuario. Ofrecen al usuario una vista del problema pero no al desarrollador.

- **Requisitos de software**. Permite que el desarrollador construya un modelo de implementación que es necesario donde se muestra que es lo que el sistema debe hacer.

3.3.1.2. Conceptos básicos

Este estándar determina los primeros pasos del ciclo de vida del software como la definición de requisitos de usuario, la definición de requisitos software y la definición del diseño arquitectónico.

Durante la fase de requisitos de usuario, a partir de unas entradas informales, se compone el documento de requisitos de usuario. Éste es necesario durante la fase de requisitos de software para generar el documento de requisitos software. Este proceso se muestra en la

ilustración 2.

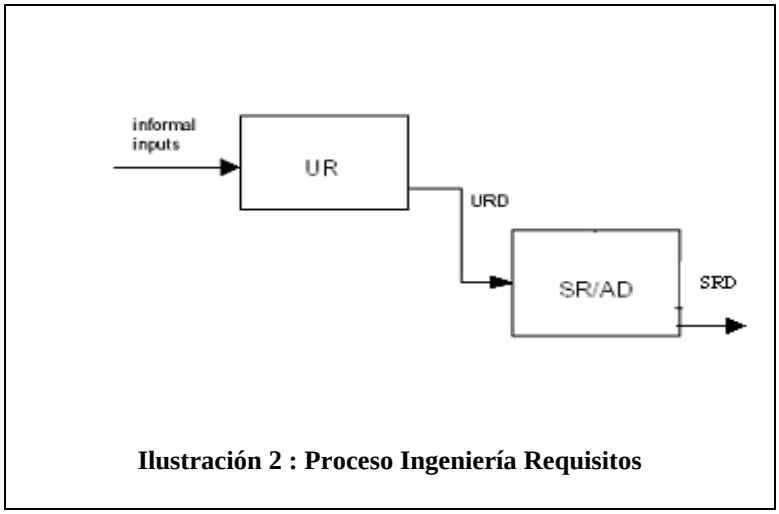

Ilustración 2 : Proceso Ingeniería Requisitos

Existen dos fases durante la recuperación de requisitos, la de requisitos de usuario y la de requisitos de software.

La fase de requisitos de usuario provee de una definición de los requisitos deseados el usuario, donde:

- La definición de los requisitos de usuario debe ser responsabilidad del usuario.
- Cada requisito de usuario debe ser identificado unívocamente.
- Para un desarrollo incremental, cada requisito de usuario debe incluir una índice de prioridad del mismo.

- Cada requisito de usuario debe ser verificable.
- La fase de obtención de estos requisitos de usuario compondrán el documento de requisitos de usuario.
- El documento de requisitos de usuario es generado antes de que el proyecto software sea iniciado.
- En la fase de requisitos software se describe que es lo que el software tiene que hacer (no cómo se tiene que hacer) y donde:
- Se debe ser tener en cuenta el documento generado en la fase de requisitos de usuario.
- Cada requisito software debe ser identificado unívocamente.
- Para un desarrollo incremental, cada requisito de software debe incluir una índice de prioridad del mismo.
- Cada requisito software debe ser verificable.
- La fase de obtención de estos requisitos software compondrán el documento de requisitos software.
- El documento de requisitos software debe cubrir todos los requisitos enumerados en el documento de requisitos de usuario.
- Debe existir una correspondencia entre los requisitos de usuario y los requisitos software.
- La construcción de un prototipo puede ayudar a clarificar los requisitos software que son necesarios.
- La terminología relacionada con la implementación debe ser omitida del documento de requisitos software.

3.3.2. Explicación de clasificación de requisitos

3.3.2.1. Requisitos de usuario

- **De capacidad**. Son aquellos que describen lo que los usuarios quieren hacer, es decir, una operación o secuencia que el software es capaz de ejecutar. En la fase de definición de requisitos software, los requisitos de capacidad serán analizados para producir un conjunto de

requisitos funcionales. Si ocurre una duplicación el los requisitos de capacidad, deben reemplazarse por un requisito funcional. Las descripciones cuantitativas deben formar parte de la especificación de un requisito de capacidad, es por ello que éste debe ser calificado con los siguientes valores:

- Capacidad. Describe cuanta capacidad es necesaria en cualquier momento. (Por ejemplo, número de usuarios soportados, número de terminales soportados, cantidad de datos a almacenar, etc).
- Velocidad. Describe cómo de rápido debe ejecutarse la operación o secuencia de operaciones. (Por ejemplo, número de operaciones hechas por intervalo de tiempo). Todos los requisitos 'respuesta' deben declarase como requisitos de interacción persona-ordenador.
- Exactitud. Describe la exactitud de una operación que es medida mediante la diferencia entre lo que es deseado y lo que realmente sucede cuando es ejecutada. La exactitud tiene en cuenta tanto errores sistemáticos como aleatorios.

- **De restricción.** Son aquellos que describen restricciones con las que se encuentra el usuario. Los usuarios pueden dar lugar restricciones en el software relacionadas con el interfaz (por ejemplo, cómo es la comunicación con otros sistemas, que hardware es usado, compatibilidades entre software, etc), la calidad del producto final (adaptabilidad, disponibilidad, portabilidad y seguridad), los recursos y las escalas de tiempo.
  - Interfaces de comunicación. Especifica las redes y los protocolos de red que son usados.
  - Interfaces hardware. Especifica todo o parte del hardware donde el software es ejecutado. Denotando limitaciones físicas (por ejemplo, tamaño, peso, etc), consideraciones ambientales (humedad, temperatura y presión) que afecten a la elección del

hardware, aspectos relacionados con la ejecución (por ejemplo, velocidad, memoria, etc).

- Interfaces software. Especifica las compatibilidades software (sistemas operativos, lenguajes de programación, compiladores, etc).

- Interacción persona-ordenador. Especifica cualquier aspecto relacionado con la interfaz de usuario. Por ejemplo, estilo, formato, tipo de mensajes, etc. El hardware en la interfaz de usuario, (por ejemplo, color del display), puede ser incluido tanto en este tipo de requisitos como en los requisitos de interfaz hardware.

- Adaptabilidad. Describe cómo el sistema debe acoplarse ante un cambio en los requisitos. Un sistema adaptable es necesario para sistemas de larga duración.

- Disponibilidad. Mide la capacidad del sistema para ser usado durante determinados períodos de operación. Cuando un sistema no es disponible, algunas o incluso ninguna de sus capacidades pueden ser usadas. Los requisitos de este tipo son descompuestos en requisitos de fiabilidad y mantenibilidad.

- Portabilidad. Mide la holgura con que el software puede ser movido de un ambiente a otro. El software portable tiende a una larga vida. La portabilidad puede ser medida, por ejemplo, en número de líneas de código que no han cambiado al portar el sistema a otro.

- Seguridad. El sistema puede necesitar seguridad contra ataques a su confidencialidad, integridad y disponibilidad. La seguridad puede ser descrita en términos de derecho de acceso, dependiendo del tipo de usuario.

- Prudencia. Define las capacidades de los usuarios para protegerse contra problemas potenciales tales como fallos de tipo hardware o software. Este tipo de requisitos define escenarios que el sistema debe manejar sin problema.

- Estándar. Referencia a los documentos aplicables que definen el estándar. Se definen dos tipos, de procesos y de productos.
- Recursos. Describe, por ejemplo, especificaciones de los recursos del ordenador (por ejemplo, memoria principal) necesario para operar con el software. La calidad y la sofisticación del producto software están limitados por los recursos elegidos para construirlo.
- Escalas de tiempo. Describe escalas de tiempo para el desarrollo y la producción del software.

3.3.2.2. Requisitos de software

- **Funcionales.** Específica las funciones que el sistema debe ser capaz de ejecutar. Éste debe definir qué procesos deben hacerse y no cómo implementarlos. Un requisito funcional debe:
  - Definir que procesos debe hacer pero no cómo se deben implementar.
  - Definir la transformación que se llevan a cabo sobre las entradas para generar las salidas.
  - Relacionarse con los requisitos de ejecución.
  - Ser rigurosos, sin usar, necesariamente, sentencias muy complejas para describirlos.
- **De ejecución**. Especifica, con valores numéricos, las variables de medición que son usadas en la definición de funciones (por ejemplo, frecuencia, capacidad, velocidad, etc). Este tipo de requisitos deben ser representados como un rango de valores, por ejemplo, valor aceptable (el mínimo nivel permitido), valor nominal (margen de seguridad sobre el valor aceptable) y valor ideal (nivel de ejecución más deseable).
- **De interfaz**. Especifica el software, hardware o los elementos de las bases de datos con los que el sistema debe interactuar o comunicarse. De este modo, los requisitos de interfaz pueden ser clasificados como: de software, de hardware o de comunicación. Este tipo de requisitos deberían definirse solamente con aspectos lógicos de una interfaz y no

con detalles físicos puesto que esto debe ser establecido en la fase de diseño.

- **De operación.** Especifica cómo el sistema va a ejecutarse y cómo comunicará con los usuarios (por ejemplo, mediante pantallas, teclados, etc). Este tipo de requisitos pueden describir aspectos físicos de la interfaz de usuario (diseño de pantallas), aspectos ergonómicos (niveles de eficiencia que el usuario debe ser capaz de conseguir). El usuario ha podido restringir la interfaz de usuario en el documento de requisitos de usuario.

- **De recursos**. Especifica los límites superiores en los recursos físicos, por ejemplo, memoria principal, espacio de disco, etc. Este tipo de requisitos deben describir las características sobre los recursos y no las restricciones a la hora de ser utilizados.

- **De verificación.** Determina cómo el producto es verificado, por ejemplo, mediante simuladores. Este tipo de requisitos restringen el diseño del producto.

- **De aceptación-testeo.** Es un tipo de requisitos de verificación. Este tipo de requisitos restringen el diseño del producto.

- **De documentación**. Referencia a la documentación necesaria durante el proyecto. Por ejemplo, el formato y el estilo de los documentos de control de la interfaz puede ser descrita en los requisitos de documentación.

- **De seguridad. (SECURITY).** Especifica requisitos para la seguridad contra ataques a la confidencialidad, integridad y a la disponibilidad. Deben describir el nivel y la frecuencia de acceso permitida a cualquier tipo de usuario del software que esté autorizado. Debe ser descrito el usuario no autorizado cuando es requerida la prevención contra este tipo de usuarios.

- **De portabilidad.** Especifica la movilidad para mover el software de un entorno a otro. Estos requisitos pueden reducir la ejecución del software e incrementar el esfuerzo requerido para construirlo.

- **De calidad.** Especifican los atributos del software que hacen que encajen para el propósito del mismo. Los atributos principales de calidad son fiabilidad, mantenibilidad y seguridad.

- **De fiabilidad.** Define la habilidad del sistema para realizar las funciones requeridas bajo unas condiciones establecidas para un período especificado de tiempo. Este tipo de requisitos pueden derivar de los requisitos de usuario de tipo disponibilidad.

- **De mantenibilidad.** Define el procedimiento con el que el software puede ser modificado para corregir fallos, para mejoras o para cambios de entorno. Este tipo de requisitos pueden derivar de los requisitos de usuario de tipo disponibilidad.

- **De seguridad**. (SAFETY) Define cualquier requisito para reducir la posibilidad de daño en el sistema que puede provocar un fallo de software. Estos requisitos pueden identificar funciones críticas cuyos fallos pueden proceder de personas o entornos dañinos. Este tipo de requisitos deben especificar qué debería suceder cuando falla una parte crítica del software.

3.4. Instituto de ingenieros eléctricos y electrónicos (IEEE: Institute Of Electrical and Electronics Engineers)

El estándar IEEE 830-1998 [IEEE 830-1998 98] fue desarrollado por el Instituto de Ingenieros Eléctricos y Electrónicos.

El software, definido en este estándar, puede contener toda la funcionalidad del proyecto esencialmente o puede ser parte de un sistema más grande.

3.4.1. Clasificación de requisitos y conceptos básicos de IEEE

Según este estándar, un requisito puede ser considerado como cualquier condición o capacidad que debe estar presente en un sistema o componente de sistema para satisfacer un contrato, estándar, especificación o cualquier otro documento formal.

3.4.1.1. Clasificación de requisitos

Se establece la siguiente clasificación de requisitos:

- Requisitos **de adaptación**. Especifican los rasgos que deben modificarse para adaptar el software a una instalación particular.

- Requisitos **de interfaces externos**. Describen, de modo detallado, todas las entradas y salidas del sistema.

- Requisitos **funcionales**. Describen las acciones fundamentales que deben lugar en el software.

- Requisitos **de ejecución**. Describen los requisitos estáticos y dinámicos que se dan lugar en el software.

- Requisitos **lógicos de las bases de datos**. Especifican los requisitos lógicos para cualquier información que es almacenada en un banco de datos.

- Requisitos **de restricciones de diseño**. Especifican los requisitos derivados de los estándares existentes.

- Requisitos **relacionados con atributos del sistema**. Existen una serie de atributos, tales como fiabilidad, disponibilidad, seguridad, mantenimiento y portabilidad.

La especificación de requisitos debe cumplimentar una serie de condiciones:

- Componer las especificaciones para un producto software en particular, programa, o juego de programas que realizan ciertas funciones en un ambiente específico.

- Especificar que funciones pueden ser ejecutadas sobre una serie de datos que producen resultados en localizaciones determinadas.

- Debe hablar del producto software, no de su proceso de producción. Requisitos del proyecto tales como su coste, su fecha de entrega no deben considerarse requisitos software.

- Deben ser escritos por uno o más representantes del proveedor, uno o más representantes del cliente, o por ambos.

3.4.2. Explicación de clasificación de requisitos

- **De adaptación**. Definen los requisitos para cualquier dato específico para un site concreto, misión o modo de operación, por ejemplo, aquellos relacionados con los límites de seguridad.

- **De interfaces externos**. Describen todas las entradas y las salidas del sistema software, además de completar la descripción de la interfaz. Se establece un patrón para cada ítem: nombre, descripción, fuente de entrada o destino de salida, rango válido, unidades de medida, tiempo, relaciones entre entradas y salidas, formatos de pantalla, formatos de ventana, formato de datos, formato de comandos, mensajes de fin, etc. Deben completar la descripción de interfaces de sistema, de usuario (describe las características lógicas de cada interfaz entre el software y sus usuarios), de hardware (describe las características lógicas de cada interfaz entre el software y el hardware), de software (describe el uso de otros productos software requeridos) y de comunicación (describe las interfaces de comunicación como los protocolos de red).

- **Funcionales.** Definen las acciones elementales que deben realizarse sobre el software aceptando y procesando un conjunto de entradas y procesando y generando un conjunto de salidas. Esto incluye chequeos válidos en las entradas, secuencia exacta de las operaciones, respuestas en situaciones anormales, efecto de los parámetros, relaciones de las salidas a las entradas, etc. Puede ser apropiado dividir los requisitos funcionales en subfunciones o subprocesos.

- **De ejecución**. Especifican los requisitos numéricos estáticos y dinámicos del software o la interacción humana con el software en conjunto.

    - Numéricos dinámicos: número de transacciones, tareas y cantidad de datos a ser procesados, incluyendo tanto los períodos de tiempo para condiciones normales como para picos de trabajo.

    - Numéricos estáticos: número de terminales, número de usuarios simultáneos, cantidad y tipo de información tratados.

- Requisitos **lógicos de las bases de datos**. Especifican los requisitos lógicos para cualquier información que se almacena en la base de datos (frecuencia de uso, habilidades de acceso, entidades de datos y sus relaciones, restricciones de integridad, retenciones de datos, tipos de información usada por varias funciones, etc).

- Requisitos **de restricciones de diseño.** Describe los requisitos relacionados con los estándares existentes o regulaciones.

- Requisitos **relacionados con atributos del sistema**. Existe una serie de atributos del sistema software que pueden considerarse requisitos como tal.

    - Fiabilidad. Describe los factores que establecen la fiabilidad requerida en el sistema.

    - Disponibilidad**.** Describe los factores que garantizan un nivel de disponibilidad definido para el sistema como un punto de control, la recuperación y al inicio.

    - Seguridad. Describe los factores que protegen al software del acceso accidental o malévolo, uso, modificación, destrucción o descubrimiento. Pueden incluirse el uso de criptografía, logs, chequear integridad de datos, restringir comunicaciones entre áreas del sistema, etc.

    - Mantenimiento. Describe atributos del software relacionados con la facilidad del mantenimiento de dicho software.

    - Portabilidad. Describe atributos del software relacionados con la facilidad de mover el software a otro servidor y/o sistemas operativos. Puede incluirse el uso de porcentaje de componentes con código cliente-servidor, uso de lenguaje portable probado, etc.

3.4.2.1. Fondo provisto para requisitos

Aparte de la definición y clasificación de los requisitos, una parte que cobra importancia en la especificación de requisitos, según la IEEE, es la descripción de los factores generales que afectan al producto software y a los propios requisitos. Es decir, una descripción donde no se declaran los

requisitos sino que se provee de un fondo para éstos. Para ello, se establecen una serie de puntos necesarios:

- Perspectiva del producto. Relaciones del producto con otros productos. En el caso de que el producto sea un componente de un sistema grande, se debe relacionar los requisitos de tal sistema a la funcionalidad del software.

- Funciones del producto. Especifica un índice de las funciones principales que el software puede realizar.

- Características del usuario. Describe las características generales de los usuarios a los que va destinado el producto (su experiencia, su nivel, etc).

- Restricciones. Se describe cualquier asunto que puede llegar a limitar las opciones del desarrollador (leyes, limitaciones de hardware, aspectos de seguridad, etc).

- Asunciones y dependencias. Se debe listar cada factor que afecta a los requisitos declarados en la especificación (por ejemplo, cambios que pueden afectar a los requisitos).

- Distribución de requisitos. Identifica aquellos requisitos que pueden ser retrasados hasta versiones futuras.

3.4.2.2. Beneficios aportados por estándar

Los beneficios aportados por la especificación de requisitos con este estándar son:

- Establecimiento de un una base sólida para alcanzar un acuerdo entre los desarrolladores y los clientes sobre lo que el producto debe hacer.

- Reducción del tiempo de desarrollo.

- Establecimiento de una base sobre la que llevar a cabo estimaciones es de costes y plan planificaciones, así como validaciones y verificaciones.

- Fácil transferencia del producto.

3.4.2.3. Características de requisitos

Los requisitos deben ser descritos con un nivel de detalle lo suficiente como para que los diseñadores hagan un sistema capaz de satisfacer tales requisitos. Cada requisito debe especificar, al menos, una descripción de cada entrada al sistema y cada respuesta de éste y todas las funciones ejecutadas en respuesta a una entrada o para dar soporte a una salida. Los requisitos declarados deben presentarse con las siguientes características:

- Correcto. Los requisitos declarados se encuentran en el producto software.
- No ambiguo. Los requisitos tienen una única interpretación.
- Completo. Los requisitos están relacionados con la funcionalidad, el desarrollo, las restricciones de diseño, los atributos y las interfaces externas y además se define todas las respuestas del software para todo tipo de entradas y en toda clase de situaciones.
- Consistente. Si el documento de especificación de requisitos contradice a algún documento de nivel mayor, entonces tal especificación no es consistente.
- Clasificado. Se debe delinear la importancia de cada requisito, mediante un ranking de necesidad (esencial, condicional y opcional) o mediante un grado de estabilidad (se refiere al número de cambios esperados a cualquier requisito).
- Verificable. Los requisitos declarados deben ser comprobables, es decir, debe existir un proceso que puede chequear que el producto software reúne tal requisito.
- Modificable. La estructura y el estilo deben ser tales que puede hacerse cualquier cambio a los requisitos fácilmente, completamente y de forma consistente conservándose la estructura y el estilo.
- Trazable. El origen de los requisitos debe ser claro y debe facilitarse la referencia de cada requisito en futuros desarrollos.

- Los requisitos no deben describir cualquier detalle de implementación o diseño ni imponer ninguna restricción adicional.

- Los requisitos deben referenciar a documentos que se han hecho previamente, deben ser identificados unívocamente, deben ser organizados para maximizar su lectura.

3.4.2.4. Organización de requisitos

Además de clasificar adecuadamente a los requisitos, éstos se deben organizar para facilitar la compresión del conjunto. Éstos pueden ser organizados atendiendo a distintos criterios:

- Organizados por modo del sistema. Un sistema puede tener diferente comportamiento dependiendo del modo de operación.

- Organizados por clases de usuario. Un sistema puede proporcionar distintas funciones dependiendo de la clase de usuario.

- Organizados por objeto. Son entidades del mundo real que tienen una contraparte dentro del sistema. Los objetos se definen mediante sus atributos y métodos. El sistema queda organizado por los objetos establecidos en el sistema.

- Organizados por rasgo. Un rasgo es un servicio externamente deseado por el sistema que puede exigir a una secuencia de entradas efectuar el resultado deseado.

- Organizados por estímulo. Un sistema puede organizarse describiendo sus funciones relacionándolo con estímulos.

- Organizados por contestación. Un sistema puede organizarse describiendo sus funciones relacionándolo con la generación de una contestación.

- Organizados por jerarquía funcional. Cuando ninguno de los esquemas orgánicos anteriores demuestra ser útil, la funcionalidad global puede organizarse en una jerarquía de funciones organizada por cualesquiera entradas comunes, rendimientos comunes, etc.

3.5. Modelo de referencia de procesamiento abierto distribuido (RM-ODP: Reference Model of Open Distributed Processing)

RM-ODP [RM-ODP 98] es un estándar publicado por ISO/IEC que provee un modelo de referencia para el procesamiento abierto y distribuido.

Este estándar proporciona una vista conceptual de la arquitectura para la construcción de sistemas distribuidos incrementalmente. Para ello, proporciona un marco de coordinación para la normalización del desarrollo de las aplicaciones distribuidas y abiertas, creando una arquitectura que soporta de modo integrado aspectos como la distribución, interoperabilidad o portabilidad de los sistemas, objetos y componentes.

3.5.1. Clasificación de requisitos y conceptos básicos de RM-ODP

3.5.1.1. Clasificación de requisitos

RM-ODP no establece una clasificación como tal de sus requisitos sino que puede describir el sistema a partir de cinco puntos de vista. Un punto de vista es una abstracción que genera una especificación de todo el sistema.

A continuación se enumeran los puntos de vista, cada uno de los cuales, está interesado en un aspecto particular del sistema.

- Empresa. Introduce los conceptos necesarios para representar un sistema en el contexto de una empresa en la que el sistema opera. El sistema es representado por una comunidad que es una configuración de objetos empresariales que persiguen un objetivo.

- Información. Describe la semántica de la información así como el tratamiento llevado a cabo sobre esa información. Un sistema puede ser descrito por objetos de información, relaciones y comportamientos.

- Computación. Permite una descomposición funcional del sistema. Las funciones son realizadas por objetos que interactúan gracias a sus interfaces.

- Ingeniería. Describe el despliegue y la comunicación del sistema con conceptos tales como canales, cluster, etc.

- Tecnología. Describe la implementación del sistema en términos de configuración de objetos tecnológicos representados por componentes hardware y software.

3.5.1.2. Estándares que especifican su estructura

El núcleo central del modelo de referencia RM-ODP se recoge en cuatro estándares que definen y especifican su estructura básica:

- **Visión de conjunto** (ISO/IEC 10764-1; ITU-T X.901). Norma que fundamenta la visión de conjunto de las motivaciones que ha de tener RM-ODP, donde se presenta el alcance, la explicación de los conceptos esenciales así como la descripción de la arquitectura.

- **Fundamentos** (ISO/IEC 10764-2; ITU-T X.902). Norma que describe la definición de los conceptos básicos de RM-ODP, así como un marco de análisis necesario para la descripción de los sistemas de procesamiento distribuidos y abiertos. Además, define los principios de conformidad y cómo éstos deben aplicarse sobre las normas RM-ODP.

- **Arquitectura** (ISO/IEC 10764-3; ITU-T X.903). Norma que especifica las características que ha de tener un procesamiento distribuido para que se considere abierto. Así como las restricciones que han de tener las normas que se integran en el sistema. Además establece una clara definición de los puntos de vista que pueden hacerse de un sistema desde diferentes perspectivas y que ayudan a su especificación global.

- **Semántica de la arquitectura** (ISO/IEC 10764-4; ITU-T X.904). Norma que formaliza los conceptos del modelo, usando para ello diferentes técnicas de descripción (por ejemplo, SDL, Z, etc.).

3.5.1.3. Transparencias de distribución

Las transparencias de distribución cobran una especial importancia en RM-ODP puesto que facilitan el establecimiento de una infraestructura para así ocultar las complejidades inherentes al sistema.

A continuación se enumeran una serie de transparencias, que dan lugar a una mayor capacidad de abstracción:

- De acceso. Permite el acceso a los objetos de información remotos de la misma forma que a los objetos de información locales.

- De fallos. Permite a los usuarios y programas de aplicación completar sus tareas a pesar de la ocurrencia de fallos en el hardware o en el software.

- De localización. Permite el acceso a los objetos de información sin conocimiento de su localización.

- De migración. Permite el movimiento de objetos de información dentro de un sistema sin afectar a los usuarios o a los programas de aplicación.

- De reubicación. Permite la reubicación de una interfaz donde estaba otra vinculada a la misma.

- De replicación. Permite utilizar múltiples instancias de los objetos de información para incrementar la fiabilidad y las prestaciones sin que los usuarios o los programas de aplicación tengan por que conocer la existencia de las réplicas.

- De persistencia. Permite la desactivación de un objeto y la activación de él mismo u otros objetos. Esto es utilizado para mantener la persistencia de un objeto cuando el sistema no es capaz de proveerlo con funciones de procesamiento, almacenamiento y comunicación continuamente.

- De transacción. Permite la coordinación, supervisión y recuperación de las transacciones entre objetos de forma externa y sin tener que involucrar a los propios objetos.

Estas transparencias se pueden llevar a cabo mediante una serie de funciones y estructuras definidas por RM-ODP. Este tipo de funciones se organizan en cuatro grupos: de gestión, coordinación, repositorio y seguridad. La existencia de estas funciones comunes ya definidas permite la construcción de aplicaciones distribuidas de una forma modular que ayuda a mejorar el tiempo de desarrollo y la fiabilidad del sistema.

3.5.1.4. Beneficios aportados por estándar

Los beneficios derivados del estándar RM-ODP se enumeran a continuación:

- Establece un marco de trabajo conceptual y una arquitectura capaz de integrar aspectos relacionados con la distribución, la interoperabilidad y portabilidad de los sistemas software. Todo ello consigue que la heterogeneidad del hardware, los sistemas operativos, lenguajes de programación y distintos sistemas de gestión sean transparentes al usuario.

- Establece un marco de coordinación para poder normalizar el procesamiento abierto y distribuido, aunando los estándares actuales así como el desarrollo de otros nuevos.

- Establece de forma clara aspectos que conforman el desarrollo de plataformas de componentes distribuidos, definiendo un vocabulario y un marco semántico común tanto para los desarrolladores como para el usuario final.

3.5.1.5. Otros normas de RM-ODP

Aparte de los cuatro estándares que definen el modelo de referencia, RM-ODP ofrece otros estándares que describen conceptos importantes tales como función de intermediación, esquema de asignación de nombres, lenguaje de definición de interfaces, referencias a interfaces y vinculación.

- Función de intermediación. Un intermediario (trading) es capaz de almacenar información acerca de posibles servicios para que sus potenciales proveedores puedan registrarse en él. A partir de ahí, los clientes envían sus peticiones a tales intermediarios quienes se encargan

de localizar a un proveedor de entre los que tiene registrados. Así, una vez localizado el proveedor, capaz de satisfacer las necesidades del consumidor, se envía la referencia del proveedor al cliente para que puedan interactuar entre sí directamente. Existen normas que describen esta función de intermediación donde se asegura que el servicio puede hacerse de forma federada entre diferentes funciones de intermediación distribuidas. Para especificar el comportamiento de esta función se usa un lenguaje de denotación formal.

- Esquema de asignación de nombres. Especifica cuál es la forma en que han de asignarse los nombres dentro de un contexto.

- Lenguaje de definición de interfaces de ODP. Define un lenguaje preciso, bien definido e independiente, el denominado lenguaje de definición de interfaces de ODP. Éste debe describir tanto las estructuras de datos que se manejan así como el perfil de las operaciones que definen sus servicios. Todo ello, para conseguir una ocultación necesaria en el desarrollo de aplicaciones abiertas.

- Referencias a interfaces y vinculación. Incluye la información necesaria para establecer vinculaciones entre objetos y soportar las transparencias de reubicación.

3.5.2. Explicación de clasificación de requisitos

Este estándar no compone su clasificación de requisitos mediante la representación de los mismos sino que RM-ODP proporciona un marco de referencia mediante el cual se puede examinar, describir y especificar un sistema desde distintas perspectivas, denominados puntos de vista.

La extensión y complejidad de un sistema de información impiden que una única persona pueda abarcar todos y cada uno de los aspectos. Por otro lado, cada persona tiene también sus propios intereses, necesidades y perspectivas distintas desde donde abordar y examinar las especificaciones de un sistema. Cada uno de estos puntos de vista trata de satisfacer a cada uno de los diferentes participantes. Además, asociado a cada uno de estos puntos de vista, se define un lenguaje

especializado que propone un vocabulario y una forma de expresarse concreta.

A continuación, se muestra en la ilustración 3, los puntos de vista necesarios para describir un sistema, según RM-ODP.

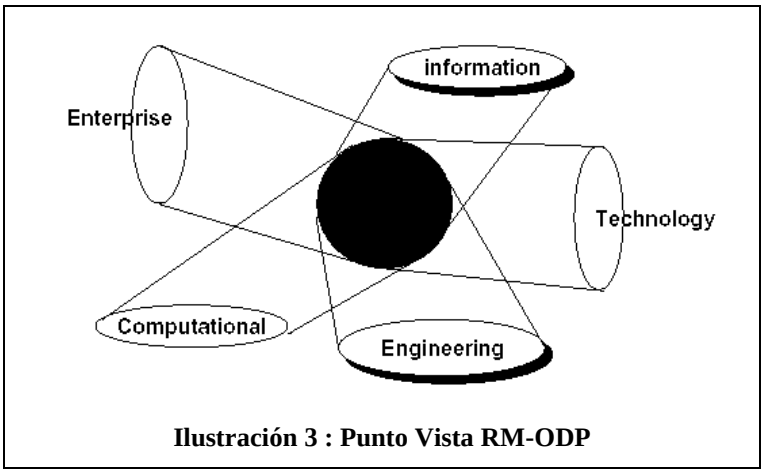

Ilustración 3 : Punto Vista RM-ODP

Los puntos de vista propuestos por RM-ODP son los siguientes:

- De la **empresa**. Propone los requisitos bajo la perspectiva del propio negocio así como el modo en que pretende satisfacerlos. Estableciendo la finalidad, el alcance, el entorno y las políticas que dirigen las actuaciones del sistema dentro de la organización de la que forme parte. Debe ser comprensible tanto para los clientes como para los usuarios. Además facilita la validación de la arquitectura software respecto a las necesidades de la empresa.

- De la **información.** Define el universo del discurso para el sistema de información, es decir, describe el tipo de información, la estructura de los datos y los posibles valores que va a usar el sistema. Deben establecerse clases de información, su semántica asociada y las restricciones que han de adoptarse en el tratamiento e interpretación de tal información. Se pueden establecer modelos orientados a objetos.

- De la **computación.** Establece la encapsulación de objetos a alto nivel (interfaces y comportamiento), es decir, describe la funcionalidad ofrecida por el sistema, su descomposición y organización funcional. Describiendo el sistema como un conjunto de objetos que interactúan entre sí a través de una serie de interfaces. Además define las fronteras entre los elementos software del sistema de información. Estas fronteras

cobran relevancia en la evolución para adaptar el software a nuevos requerimientos y cambios tecnológicos.

- De la **ingeniería.** Expresa la naturaleza distribuida del sistema, es decir, define la infraestructura que se necesita para soportar el procesamiento distribuido así como el modo de distribuir los datos y las operaciones que se llevan a cabo en el sistema para proporcionar las funcionalidades requeridas. El modelo de referencia de infraestructura distribuida puede usar canales para modelar todo tipo de conexiones middleware.

- De la **tecnología.** Define correspondencias de los objetos de ingeniería, con los estándares y la tecnología, es decir, describe la tecnología que soporta el sistema en base a la infraestructura definida en el punto de vista de la ingeniería (hardware, sistemas operativos, compiladores, etc.).

Debe existir una clara relación entre los puntos de vista puesto que todos ellos son complementarios y especifican un mismo sistema, es decir, se debe definir un marco común para garantizar la coherencia mutua entre ellos, identificando relaciones. De hecho, la segunda norma de RM-ODP establece un marco común que garantiza la coherencia mutua entre ellos.

Para la especificación del sistema desde los distintos puntos de vista, se usa el modelado de objetos común y técnicas de orientación a objetos, lo que proporciona una base común para los distintos lenguajes de los puntos de vista, además de abstracción y encapsulación.

3.5.2.1. Lenguajes asociados a cada punto de vista

Existen cinco tipos de lenguajes que definen los conceptos y las reglas para la especificación de un sistema ODP desde un determinado punto de vista, uno por cada punto de vista diferente.

Existen una serie de conceptos fundamentales para ilustrar el lenguaje desde un determinado punto de vista. A continuación, se enumeran algunos ejemplos:

- De la empresa. Comunidad, federación, etc.

- De la información. Esquema estático, esquema dinámico y esquema invariante.

- De la computación. Señal, operación, anuncio, interrogación, flujo, interfaz, plantilla, firma, etc.

- De ingeniería. Objeto ingeniero, cluster, cápsula, núcleo, nodo, canal, stub, protocolo, migración, desactivación, etc.

Un objeto es caracterizado por su comportamiento (conjunto de acciones con una serie de restricciones) o por su estado (condición de un objeto que determina el conjunto de acciones de las que puede tomar parte el objeto). Dependiendo del punto de vista, el énfasis a la hora de describir un objeto, puede ser dado a su comportamiento o a su estado. A la hora de identificar un objeto se requiere de unas conceptos de nombrado tales como nombre del objeto, nombre del dominio, nombre del contexto, etc.

Los lenguajes también pueden definir una serie de reglas que identifican comportamientos determinados.

4. SWREUSER

SwReuser [SwReuser 08] es una herramienta case usada durante el proceso de desarrollo del software.

La herramienta, basada en el paradigma de reutilización, utiliza la gestión de conocimiento apropiada. La clave fundamental de SwReuser es la de soportar distintas clasificaciones de tipos de requisitos.

SwReuser incluye:

- Gestión del vocabulario del problema y modelado del negocio.
- Gestión del versionado de cada requisito.
- Gestión de los cambios de estado de cada requisito.
- Gestión de la documentación de las decisiones tomadas.
- Establecimiento de prioridades.
- Gestión de las relaciones entre requisitos.
- Trazabilidad total con cualquier otro elemento del ciclo de vida de desarrollo. Es decir, es una técnica que se utiliza para proporcionar relaciones entre requisitos, elementos de diseño y elementos de implementación de un software para poder gestionar los cambios y asegurar el éxito del sistema.
- Conexión con los riesgos referentes a cada requisito.
- Gestión de casos de prueba por cada requisito.

El metamodelo implementado ha sido pensado para ser integrado en esta herramienta case, para aprovecharlo completamente y obtener una mejora en la organización de requisitos durante todo el ciclo de vida del software.

El metamodelo proporciona estructuras organizativas que han sido probadas con éxito para solucionar un problema recurrente en un proyecto software. Esto es la clave para su reuso en la búsqueda de soluciones en otros proyectos software.

5. REUSO SOFTWARE

Reuso, en la ingeniería del software, es definido por Krueger [Krueger 92] como el proceso de creación de sistemas software a partir de un sistema software existente.

Ezran [Ezran 00] lo define como el proceso sistemático de software construido a partir de un stock de bloques ya construidos. Así, las similitudes en requisitos y/o arquitectura entre aplicaciones pueden ser explotadas para conseguir destacados beneficios en productividad, calidad y acciones de negocio.

En sus inicios, el reuso software, se limitaba a la reutilización de código fuente, sin tener en cuenta muchos activos que surgen durante todo el proceso del desarrollo software. Sin embargo, gracias al avance en la búsqueda y recuperación de información sobre los repositorios donde se almacenan los activos reutilizables, se produjo una evolución del reuso a niveles mayores de abstracción.

Según Karlsson [Karlsson 95], un componente reusable es cualquier componente que es desarrollado específicamente para ser usado y, en efecto, es usado en más de un contexto. Esto no sólo incluye código sino otros productos del ciclo de vida del sistema, tales como especificaciones, diseños e incluso requisitos.

Cuanto más arriba se suba en el ciclo de vida, más valor tiene el activo a reutilizar, puesto que se ha requerido un mayor poder de abstracción para crearlo; y más independientemente se es de la plataforma tecnológica, lo que evita riesgos de obsolescencia antes de la reutilización.

El principal beneficio asociado a la reutilización es la mejora en la productividad del equipo, incrementando la calidad del producto software. Las ventajas, asociadas a la reutilización software, se enumeran a continuación.

- Reducción de los costes de desarrollo.
- Aumento de la calidad de los productos.

- Aumento de la productividad, mediante la mejora de los tiempos en los que se desarrollan los nuevos proyectos.
- Mejoras en las actividades de mantenimiento y soporte de aplicación.
- Mejoras en las actividades de control y planificación por la reducción de desviaciones en los desarrollos.

Los aspectos menos beneficios, a tener en cuenta, en la reutilización software son:

- Técnicas tradicionales implican una alta inversión inicial, lo que dificulta la amortización de la inversión.
- Nuevas herramientas y cambios en el proceso productivo avivan el 'temor al cambio', es decir, existe una gran falta de confianza en el reuso software puesto que muchos especialistas son detractores del reuso y es escasa la inversión propuesta para ello.

6. METAMODELO DE CLASIFICACIÓN DE TIPOS DE REQUISITOS

Las tareas de definición y administración de requisitos son la clave para el éxito de un proyecto. Es por ello que la ingeniería de requisitos juega un papel fundamental en el desarrollo de proceso software.

Existen varias clasificaciones de tipos de requisitos ofertadas por una serie de estándares, estudiados en este proyecto, tales como CC, ESA, IEEE, Métrica y RM-ODP. Por ello, es difícil encontrar una herramienta CASE que soporte distintas clasificaciones de tipos de requisitos para las necesidades de diferentes proyectos.

La solución propuesta a tal problema es un metamodelo [metamodelo 06] que pueda instanciarse en cualquier clasificación de tipos de requisitos para los estándares mencionados y otros generados. De esta forma podremos almacenar cualquier estructura organizativa de requisitos. Luego recuperar una estructura e integrarla a un proyecto específico con los correspondientes beneficios del reuso, estableciendo así una mejora en el proceso del desarrollo del software.

Esta representación de estructuras organizativas consta de un conjunto de tipos y subtipos de requisitos. Estos tipos y subtipos pueden tener diferentes niveles de jerarquía.

Los tipos de requisitos de una estructura, independiente del nivel jerárquico, podrán tener algún tipo de asociación entre ellos y podrán ser relacionados con elementos externos.

La implementación de un metamodelo de tales características trae consigo una serie de beneficios tales como ayuda en la gestión de requisitos, posibilidad de reutilización de los requisitos en futuros proyectos y, ante todo, una mejora en el proceso de desarrollo del software en el campo de la ingeniería de requisitos.

Los estándares usados para articular este metamodelo son **Common Criteria**, **ESA** (European Spacial Agency), **IEEE** (Institute of Electrical and

Electronics Engineers), **Métrica** y **RM-ODP** (Reference Model Distributed Processing).

El metamodelo puede ser aprovechado aún más, en la gestión de requisitos, si fuese integrado en una herramienta CASE como SwReuser [SwReuser 08].

6.1. Metamodelos asociados a cada estándar para la clasificación de tipos de requisitos

Los estándares estudiados en este proyecto (Common Criteria, ESA, IEEE, Métrica y RM-ODP) proponen distintos tipos de estructuras organizativas en la organización de requisitos.

A continuación, se muestran los metamodelos asociados a cada uno de los estándares asociados mediante diagramas UML.

6.1.1. Common Criteria

Common Criteria es un estándar orientado a la evaluación de seguridad en el campo de la tecnología de la información.

Presenta un conjunto de requisitos de seguridad (componentes funcionales de la seguridad) y un conjunto de requisitos de garantía de la seguridad (componentes de garantía de la seguridad). Todos estos requisitos se organizan de forman jerárquica en clases, familias y componentes.

Por tanto, el metamodelo asociado permite representar relaciones de generalización, tal como se muestra en la ilustración 4.

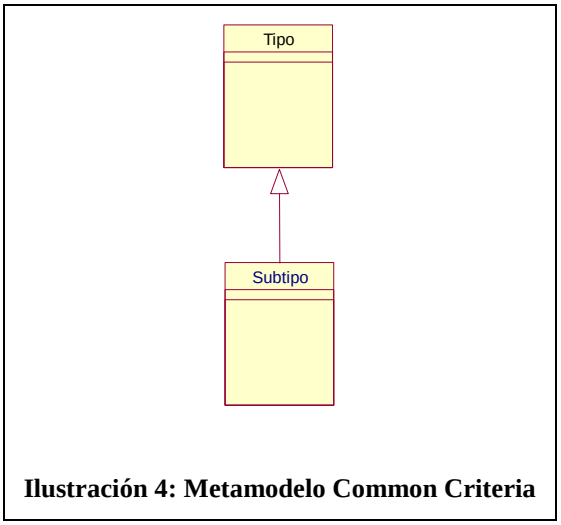

Ilustración 4: Metamodelo Common Criteria

6.1.2. ESA

ESA aplica una estructura organizativa compuesta de una serie de divisiones de los requisitos dependiendo de la naturaleza de los mismos.

Los requisitos de software se dividen en requisitos funcionales, de rendimiento, de interfaz, operacionales, de recursos, de verificación, de pruebas de aceptación, de documentación, de seguridad, de portabilidad, de calidad, de fiabilidad, de mantenimiento y de protección.

Por tanto, el metamodelo asociado, sólo permite representar tipos de requisitos mientras que las relaciones entre tales tipos no están permitidas, tal como se muestra en la ilustración 5.

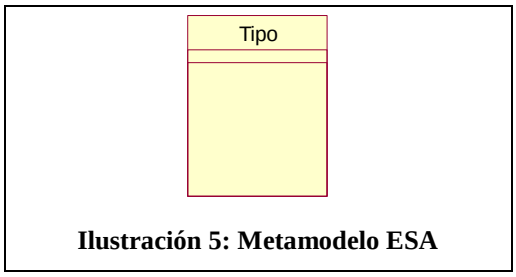

Ilustración 5: Metamodelo ESA

6.1.3. IEEE

IEEE es un estándar que define las prácticas recomendadas para la correcta especificación de requisitos software.

El documento de requisitos software debe contar con todos los requisitos a un nivel de detalle suficiente para que el diseño satisfaga los requisitos y para que, a la hora de realizar un test, el sistema satisfaga tales requisitos.

La clasificación de requisitos establecida por este estándar cuenta con requisitos de adaptación, de interfaces externos, funcionales, de ejecución, lógicos de las bases de datos, de restricciones de diseño y relacionados con atributos del sistema.

El metamodelo asociado sólo permite representar tipos de requisitos mientras que las relaciones entre tales tipos no están permitidas, tal como se muestra en la ilustración 6.

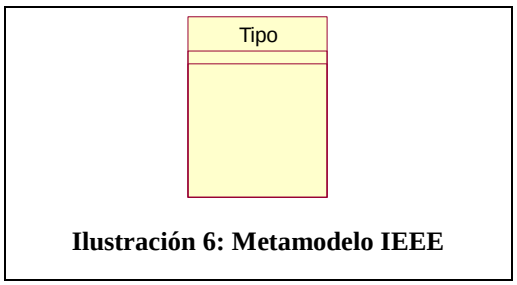

Ilustración 6: Metamodelo IEEE

6.1.4. Métrica

Métrica está basada en un modelo de procesos: planificación de sistemas de información, desarrollo de sistemas de información y mantenimiento de sistemas de información.

Cada uno de los procesos se descompone en actividades, y éstas, a su vez, en tareas.

La clasificación de requisitos establecida por este estándar cuenta con requisitos funcionales, de rendimiento, de seguridad, de implantación y de disponibilidad del sistema.

Métrica propone una organización de requisitos que consta sólo de un nivel jerárquico. Por tanto, el metamodelo asociado sólo permite representar tipos de requisitos mientras que las relaciones entre tales tipos no están permitidas, tal como se muestra en la ilustración 7.

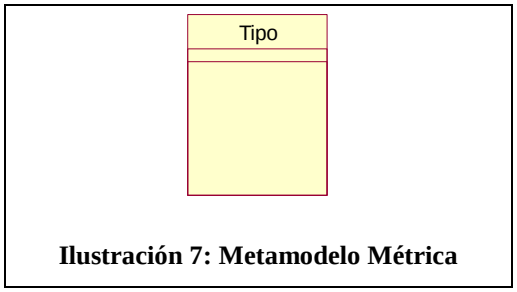

Ilustración 7: Metamodelo Métrica

6.1.5. RM-ODP

RM-ODP proporciona un marco de referencia mediante el cual poder examinar, describir y especificar un sistema desde distintas perspectivas, denominadas puntos de vista. Cada uno de estos puntos de vista trata de satisfacer a una audiencia distinta, cada una interesada en aspectos diferentes del sistema.

Los puntos de vista propuestos (computacional, de información, empresarial, tecnológico y de ingeniería) son usados como modo de clasificación de requisitos.

Por tanto, el metamodelo asociado sólo permite representar tipos de requisitos mientras que las relaciones entre tales tipos no están permitidas, tal como se muestra en la ilustración 8.

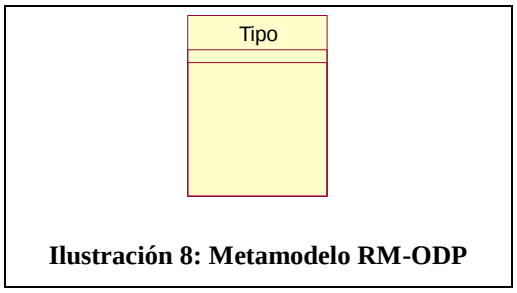

Ilustración 8: Metamodelo RM-ODP

6.1.6. Metamodelo de clasificación de tipos de requisitos

Para lograr el objetivo de este proyecto debemos crear un metamodelo que almacene las estructuras organizativas de tipos de requisitos propuestos por los estándares estudiados.

El metamodelo que envuelve o clasifica a todas las estructuras organizativas de los estándares mencionados y que forma parte del patrón de clasificación de tipos, se refleja, mediante un diagrama de clases, en la ilustración 9.

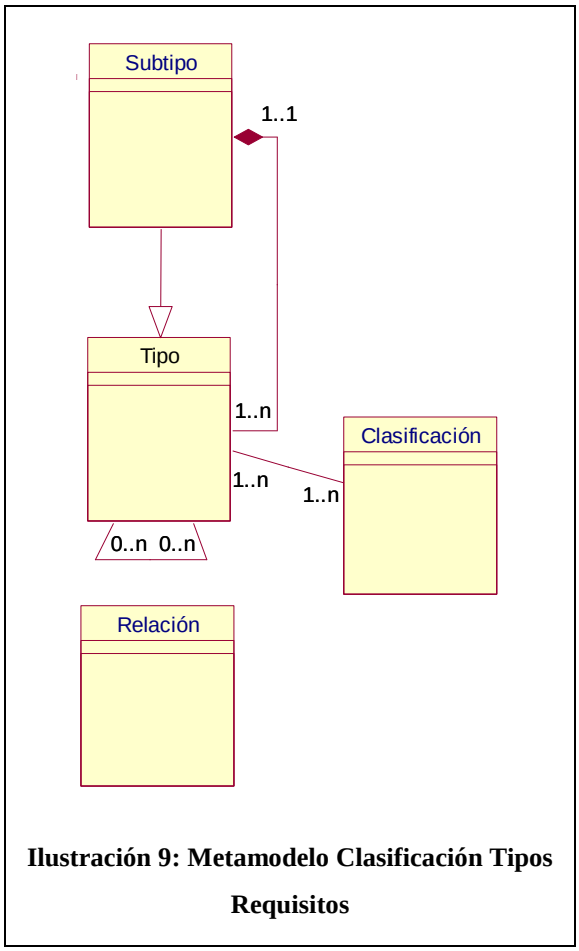

Ilustración 9: Metamodelo Clasificación Tipos Requisitos

- Clase **clasificación**. Representa todos los posibles estándares estudiados para la clasificación de requisitos: Common Criteria, ESA, IEEE, Métrica y RM-ODP. También puede representar a estructuras organizativas generadas por distintas personas o instituciones para luego reusarlas.

- Clase **tipo**. Representa a la generalización de los subtipos de requisitos.

- Clase **subtipo**. Representa todos los posibles tipos que en su momento son subtipos de otro tipo. Los requisitos están contenidos dentro de los subtipos o tipo determinado.

- Clase **relación**. Es una clase asociación que representa todas las posibles relaciones entre tipos de requisitos.

Se usa el patrón **composite** para describir una composición recursiva donde la clase subtipo se compone de objetos que, a su vez, tienen hijos

y son del mismo tipo. La clase abstracta tipo implementa un comportamiento común para los subtipos.

La intención es componer objetos en una estructura árbol para representar jerarquías del tipo “parte-todo”. Esto permite tratar objetos individuales y composiciones de objetos de manera uniforme.

Se puede aprovechar la idea de patrones, explicada en el apartado siguiente, para fundamentar la idea del metamodelo de clasificación de tipos de requisitos. Esto es, considerándolo como patrón con el objeto de transmitir conocimiento para la solución de problemas en el ámbito de la ingeniería de los requisitos.

En el capítulo siguiente, se desarrolla más detalladamente esta idea de definir nuestro patrón como un conjunto de tipos de requisitos, con posibilidad de relacionarse entre sí, junto con otros elementos externos, que han sido probados con éxito para solucionar un problema recurrente en un proyecto software.

7. DEFINICIÓN DE PATRÓN. PATRÓN DE CLASIFICACIÓN DE TIPOS DE REQUISITOS

7.1. Patrones

La idea de los patrones fue introducida por Christopher Alexander [Christopher Alexander 77] en el ámbito de la arquitectura civil, aunque hace ya años que la idea ha sido acogida en la ingeniería del software.

Cada patrón describe un problema que ocurre una y otra vez en un entorno dado, describiendo el núcleo de la solución al problema de tal forma que pueda ser empleada un millón de veces sin hacerlo dos veces de la misma forma.

El concepto de reutilización de software está bien relacionado con los patrones puesto que éstos proponen una forma de transmitir experiencias exitosas a problemas recurrentes, proporcionado soluciones que deben ser reutilizables para problemas similares en distintas circunstancias.

A la hora de describir un patrón se ha de hacer explícitamente para que pueda ser transmitido correctamente, por tanto, se ha de proveer una plantilla conceptual que muestra el núcleo de la solución de un problema específico, donde las explicaciones deben ser claras y expresadas de manera formal.

Además, la efectividad de la solución ha debido ser probada muchas veces por profesionales dentro de un contexto determinado en situaciones anteriores.

En definitiva, la ventaja asociada al concepto de patrón a la hora de mejorar las prácticas en el desarrollo software es la propia reutilización de la misma solución retocada para resolver problemas similares en diferentes situaciones.

7.2. Patrón de clasificación de tipos de requisitos

El ingeniero de software debe probar su efectividad resolviendo problemas de distinta índole, pudiendo reusar una misma solución adaptada para problemas similares. La idea de patrones pretende reusar las experiencias en la solución de problemas.

En concreto, en este proyecto, se considera la idea del metamodelo de clasificación de tipos de requisitos como patrón con el objeto de transmitir conocimiento para la solución de problemas del ámbito de los requisitos.

El patrón de clasificación de tipos pretende almacenar y reusuar, de manera efectiva, los distintos esquemas de clasificación de requisitos propuestos por los estándares estudiados para mejorar la especificación y gestión de requisitos.

A continuación, se detallan los atributos y valores que componen la descripción del patrón de clasificación de tipos.

En la tabla 1 se detallan todos los atributos de la clase clasificación.

Atributos clasificación	Valor
Identificador	Identificador de la clasificación.
Nombre	Nombre de la clasificación.
Autor	Nombre estándar u otra fuente
Descripción	Descripción general, ventajas y desventajas a la hora de usarla.
Fecha de creación	Fecha.

Tabla 1: Atributos Clasificación

En la tabla 2 se detallan todos los atributos de la clase tipo.

Atributos tipo	Valor
Identificador	Identificador del tipo.
Nombre	Nombre del tipo.

Descripción	Descripción el tipo.
Padre	Padre del tipo.
Nivel de jerarquía	Profundidad del tipo.
Relaciones	Posibles relaciones entre tipos: asociación, agregación, composición, generalización.
Elementos externos	Elemento externo asociado, por ejemplo, archivo, página web, etc.
Requisitos individuales	Ejemplos de requisitos pertenecientes a un subtipo determinado.

Tabla 2: Atributos Tipo

La solución propuesta tiene que representar cualquier clasificación estudiada en este proyecto, los tipos y subtipos pertenecientes a tales clasificaciones, las relaciones entre éstos y los elementos externos asociados, estableciendo así una mejora en el proceso del desarrollo del software.

En general, los beneficios proporcionados por este patrón son tales como:

- Ayuda en la administración de requisitos.
- Posibilidad de reutilización en los requisitos en futuros proyectos.

Como trabajo futuro, podría estudiarse completamente el beneficio obtenido en la aplicación de este patrón durante el proceso del desarrollo del software en el campo de la ingeniería de requisitos.

7.2.1. Ventajas de patrón de clasificación de tipos

El patrón de clasificación de tipos considera la representación de atributos descriptivos en el propio patrón que permita el continuo aprendizaje para su uso.

El proceso de ingeniería de requisitos puede ser mejorado mediante el patrón de clasificación de tipos aplicando el reuso de tipos requisitos.

Las posibles mejoras aportadas por el patrón son:

- Mejora en la especificación y organización de requisitos.
- Mejora en la gestión de requisitos durante todo el proceso de desarrollo software para combatir la volatibilidad de los mismos.
- Disminución en los tiempos de definición de requisitos con el consiguiente ahorro de recursos y sin perjudicar la calidad de los mismos, puesto que se usan activos software reutilizados con éxito en proyectos anteriores.
- Disminución del coste de proceso de desarrollo del software.
- Mejora en la trazabilidad para poder gestionar mejor los cambios y asegurar el éxito del sistema.
- Generación de activos reutilizables, optimizando el reuso efectivo de requisitos y mejorar así la gestión de requisitos.

8. APLICACIÓN METAMODELO CLASIFICACIÓN TIPOS DE REQUISITOS

Para la implementación del metamodelo se ha desarrollado una aplicación en C# bajo la plataforma .NET.

El programa posibilita la gestión de las estructuras organizativas necesaria en la organización de requisitos.

8.1. Requisitos para ejecutar la aplicación

El ejecutable de la aplicación, denominado MetaModeloTipoRequisitos.exe, se localiza en el directorio raíz del proyecto.

Para ejecutar la aplicación metamodelo de clasificación de tipos de requisitos se debe hacer doble clic sobre el ejecutable.

El único requisito necesario para la correcta ejecución del programa es tener el framework .NET instalado en la máquina.

El framework .NET, a partir del sistema operativo Windows XP, ya viene instalado por defecto. En cualquier caso, se puede descargar en:

http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5

8.2. Casos de uso

Los casos de uso definidos por Christerson [Christerson 92] son usados para identificar las funcionalidades de la aplicación.

A continuación, se muestra los casos de uso, agrupados en paquetes, que describen el comportamiento de la aplicación. Se enumeran a continuación.

- Paquete gestión tipos clasificaciones.
    - Paquete gestión tipos.
    - Paquete gestión clasificaciones.

- Caso de uso Guardar metamodelo.

- Paquete gestión clasificaciones.
  - Caso de uso Creación clasificación.
  - Caso de uso Búsqueda clasificación.
  - Caso de uso Modificación clasificación.
  - Caso de uso Eliminación clasificación.

- Paquete gestión tipos.
  - Caso de uso Creación tipo.
  - Caso de uso Modificación tipo.
  - Caso de uso Eliminación tipo.

El actor, que demanda funcionalidades, es todo aquel usuario capacitado para el manejo de la aplicación de gestión de patrones de tipos de requisitos.

8.2.1. Paquete gestión tipos clasificaciones

La ilustración 10 muestra el paquete gestión tipos clasificaciones q contiene el paquete gestión tipos, el paquete gestión clasificaciones y el caso de uso guardar metamodelo.

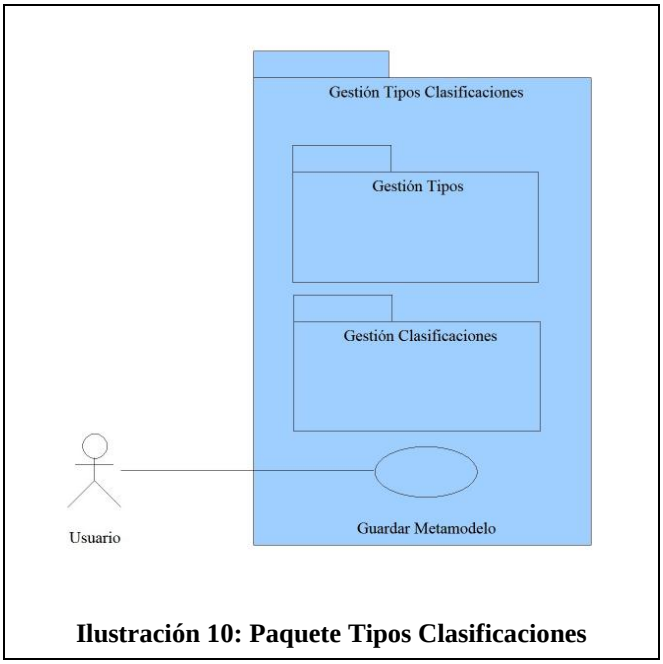

Ilustración 10: Paquete Tipos Clasificaciones

8.2.1.1. Caso de uso. Guardar metamodelo

El caso de uso Guardar Metamodelo se describe en la tabla 3, mostrada a continuación.

Nombre	Guardar Metamodelo.
Actores	Usuario.
Objetivo	Almacenamiento del metamodelo en un archivo con formato xml.
Precondiciones	El usuario ha entrado en la aplicación y ha creado, al menos, una clasificación.
PostCondiciones	Metamodelo guardado.
Escenario básico	Escritura de todas las clasificaciones pertenecientes al metamodelo mediante la escritura del mismo en un fichero xml.

Tabla 3: Caso Uso. Guardar Metamodelo

8.2.2. Paquete gestión clasificaciones

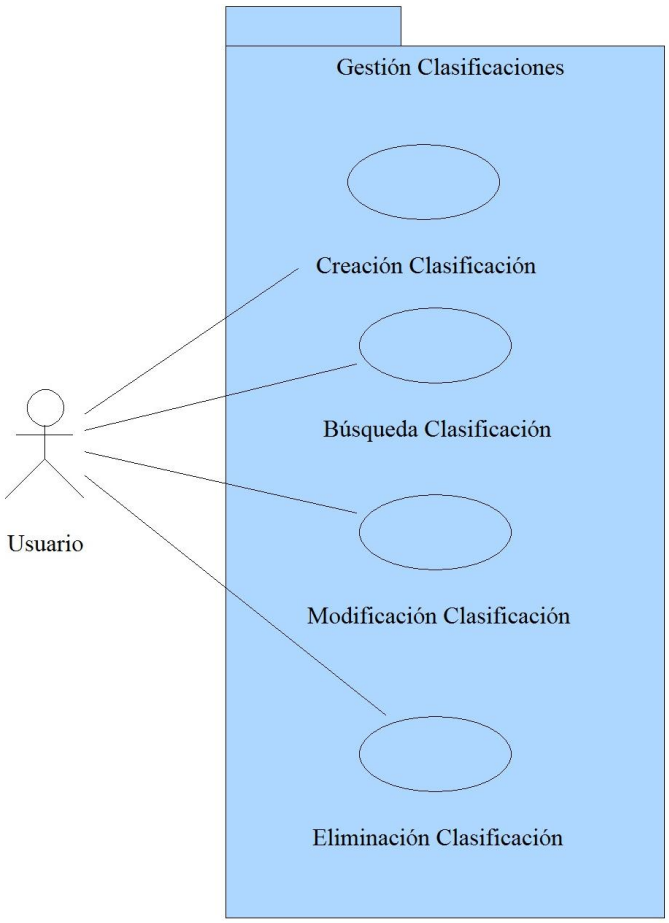

Ilustración 11: Paquete Gestión Clasificaciones

La ilustración 11 refleja el paquete gestión clasificaciones, el cual, contiene los casos de uso: creación clasificación, búsqueda clasificación, modificación clasificación y eliminación clasificación.

8.2.2.1. Caso de uso. Creación clasificación

El caso de uso Creación Clasificación se describe en la tabla 4, mostrada a continuación.

Nombre	Creación Clasificación.
Actores	Usuario.
Objetivo	Creación de una clasificación perteneciente al metamodelo.
Precondiciones	El usuario ha entrado en la aplicación.

PostCondiciones	Clasificación creada.
Escenario básico	Creación de la clasificación, introduciendo las propiedades de la clasificación.

Tabla 4: Caso Uso. Creación Clasificación

8.2.2.2. Caso de uso. Búsqueda clasificación

El caso de uso Búsqueda Clasificación se describe en la tabla 5, mostrada a continuación.

Nombre	Búsqueda Clasificación.
Actores	Usuario.
Objetivo	Búsqueda de una clasificación perteneciente al metamodelo.
Precondiciones	El usuario ha entrado en la aplicación y ha creado previamente, al menos, una clasificación.
PostCondiciones	Clasificación buscada.
Escenario básico	Introduciendo unos criterios de búsqueda, se obtiene un listado de clasificaciones coincidentes con los filtros introducidos.

Tabla 5: Caso Uso. Búsqueda Clasificación

8.2.2.3. Caso de uso. Modificación clasificación

El caso de uso Modificación Clasificación se describe en la tabla 6, mostrada a continuación.

Nombre	Modificación Clasificación.
Actores	Usuario.
Objetivo	Modificación de una clasificación del metamodelo.
Precondiciones	El usuario ha entrado en la aplicación y ha creado previamente, al menos, una clasificación.
PostCondiciones	Clasificación modificada.
Escenario básico	Selección de una clasificación en la pantalla principal. Modificación de cualquier atributo de la clasificación, a excepción de su identificador.

Tabla 6: Caso Uso. Modificación Clasificación

8.2.2.4. Caso de uso. Eliminación clasificación

El caso de uso Eliminación Clasificación se describe en la tabla 7, mostrada a continuación.

Nombre	Eliminación Clasificación.
Actores	Usuario.
Objetivo	Eliminación de una clasificación del metamodelo.
Precondiciones	El usuario ha entrado en la aplicación y ha creado previamente, al menos, una clasificación.
PostCondiciones	Clasificación eliminada.
Escenario básico	Selección de una clasificación en la pantalla principal. Eliminación de la clasificación.

Tabla 7: Caso Uso. Eliminación Clasificación

8.2.3. Paquete gestión tipos

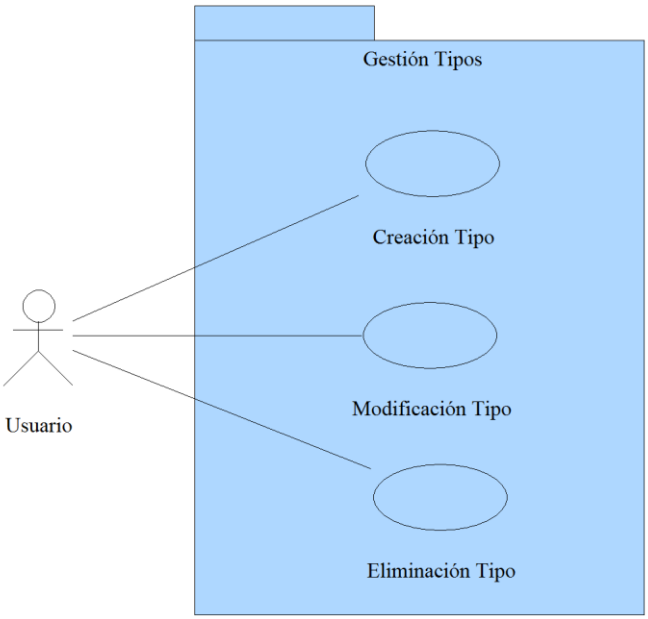

Ilustración 12: Paquete Gestión Tipos

La ilustración 12 muestra el paquete gestión tipos, el cual, contiene los casos de uso: creación tipo, modificación tipo y eliminación tipo.

8.2.3.1. Caso de uso. Creación tipo

El caso de uso Creación Tipo se describe en la tabla 8, mostrada a continuación.

Nombre	Creación Tipo.
Actores	Usuario.
Objetivo	Creación de un tipo perteneciente a una clasificación o a un nodo tipo padre.
Precondiciones	El usuario ha entrado en la aplicación y ha creado previamente, al menos, una clasificación.
PostCondiciones	Tipo creado.
Escenario básico	Selección de un nodo padre en la pantalla principal. Introducción de las propiedades del tipo. Gestión de los nodos relacionados con el tipo. Gestión de los elementos externos del tipo. Gestión de los subtipos del tipo, es decir, los hijos del nodo tipo creado.

Tabla 8: Caso Uso. Creación Tipo

8.2.3.2. Caso de uso. Modificación tipo

El caso de uso Modificación Tipo se describe en la tabla 9, mostrada a continuación.

Nombre	Modificación Tipo.
Actores	Usuario.
Objetivo	Modificación de un tipo perteneciente a una clasificación o a un nodo tipo padre.
Precondiciones	El usuario ha entrado en la aplicación y ha creado previamente, al menos, una clasificación y un nodo tipo.
PostCondiciones	Tipo modificado.
Escenario básico	Selección de un nodo padre en la pantalla principal. Modificación de las propiedades del tipo. Gestión de los nodos relacionados con el tipo. Gestión de los elementos externos del tipo.

	Gestión de los subtipos del tipo, es decir, los hijos del nodo tipo creado.

Tabla 9: Caso Uso. Modificación Tipo

8.2.3.3. Caso de uso. Eliminación tipo

El caso de uso Eliminación Tipo se describe en la tabla 10, mostrada a continuación.

Nombre	Eliminación Tipo.
Actores	Usuario.
Objetivo	Eliminación de un tipo perteneciente a una clasificación o a un nodo tipo padre.
Precondiciones	El usuario ha entrado en la aplicación y ha creado previamente, al menos, una clasificación y un tipo.
PostCondiciones	Tipo eliminado.
Escenario básico	Selección de un nodo tipo en la pantalla principal. Eliminación del nodo tipo.

Tabla 10: Caso Uso. Eliminación Tipo

8.3. Pantallas del programa

8.3.1. Pantalla principal. Plantilla para la gestión de patrones de tipos de requisitos

La pantalla principal, Plantilla para la gestión de patrones de tipos de requisitos, que aparece en la ilustración 13, muestra el árbol de clasificaciones guardado en el fichero xml de almacenamiento. En el caso de que no hubiese fichero xml, no aparecería ninguna clasificación. El árbol construido, mediante el control de .NET TreeView, muestra los identificadores de cada nodo (clasificación/tipo).

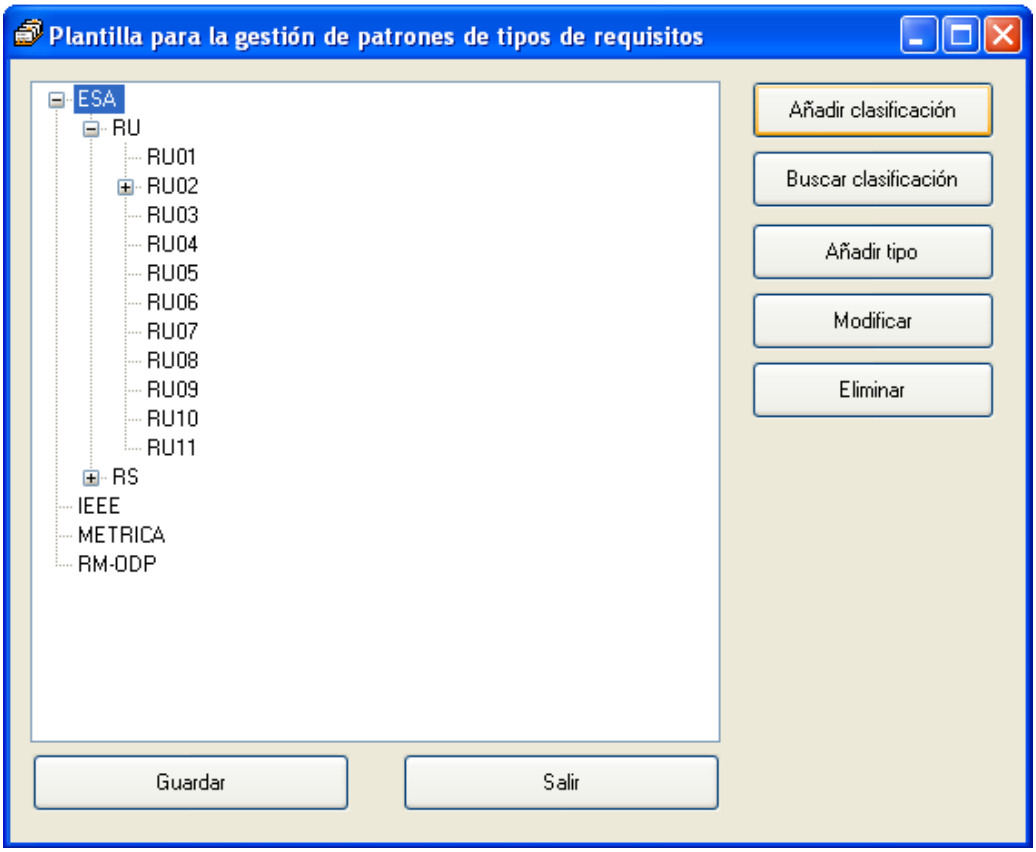

Ilustración 13: Pantalla Plantilla Gestión Patrones Tipos Requisitos

El menú de la derecha permite:

- Añadir una clasificación.
- Buscar una clasificación.
- Añadir un tipo a una clasificación o a un tipo “padre”.
- Modificar cualquier tipo de nodo (clasificación/tipo).
- Eliminar nodo (clasificación/tipo).

Mediante el botón derecho del ratón, también se puede llegar al menú de operaciones, mostrado en la ilustración 14. Esto puede implementarse gracias al menú contextual de .NET.

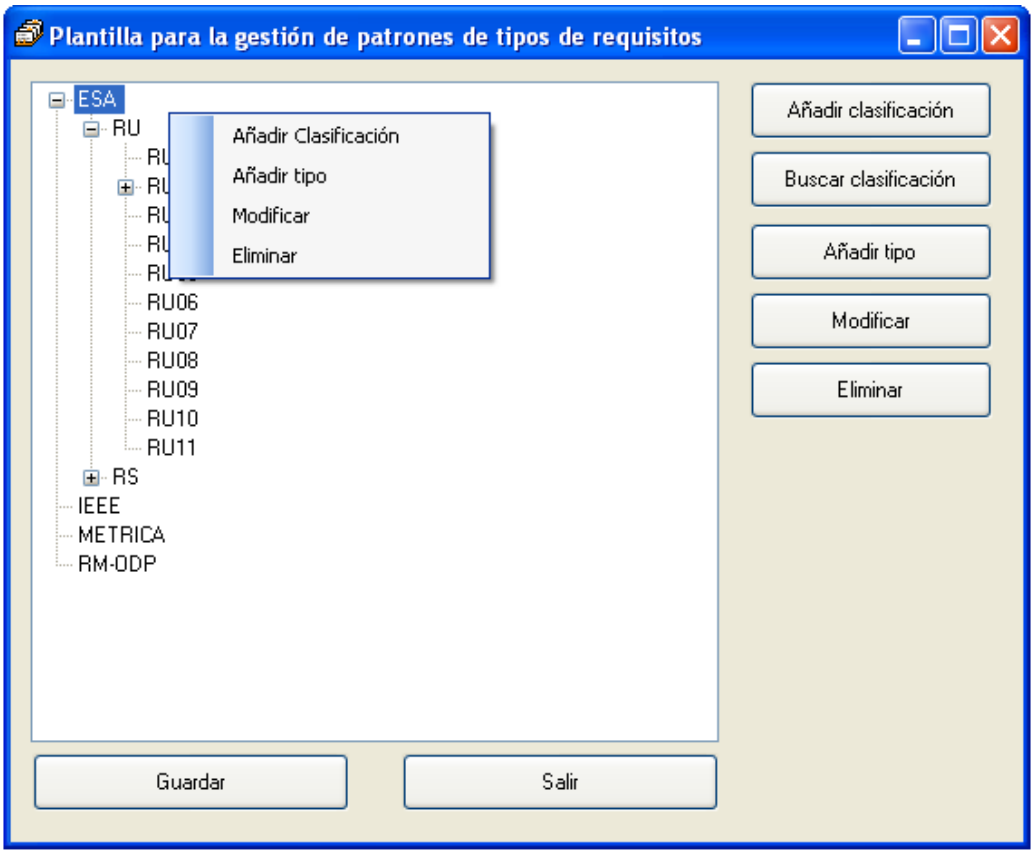

Ilustración 14: Menú Contextual Plantilla Gestión Patrones Tipos Requisitos

También mediante una serie de teclas, se puede interactuar con la aplicación.

- Tecla SUPR. Seleccionado un nodo del árbol, puede eliminarse éste pulsando sobre SUPR.

- Tecla ENTER. Seleccionado un nodo del árbol, puede verse toda su información pulsando sobre ENTER.

Los botones mostrados en la parte inferior de la pantalla.

- Guardar. Guarda el metamodelo construido y reflejado en el árbol. Esta acción construye un fichero xml con toda la información del metamodelo. Si previamente a salir, no se guarda, se perderán todos los cambios.

- Salir. Sale de la aplicación.

La aplicación contiene una serie de alertas informativas advirtiendo del resultado de la operación invocada en cada momento. Por ejemplo,

cuando se pincha sobre el botón guardar, se muestra este mensaje, tal y como se puede observar en la ilustración 15.

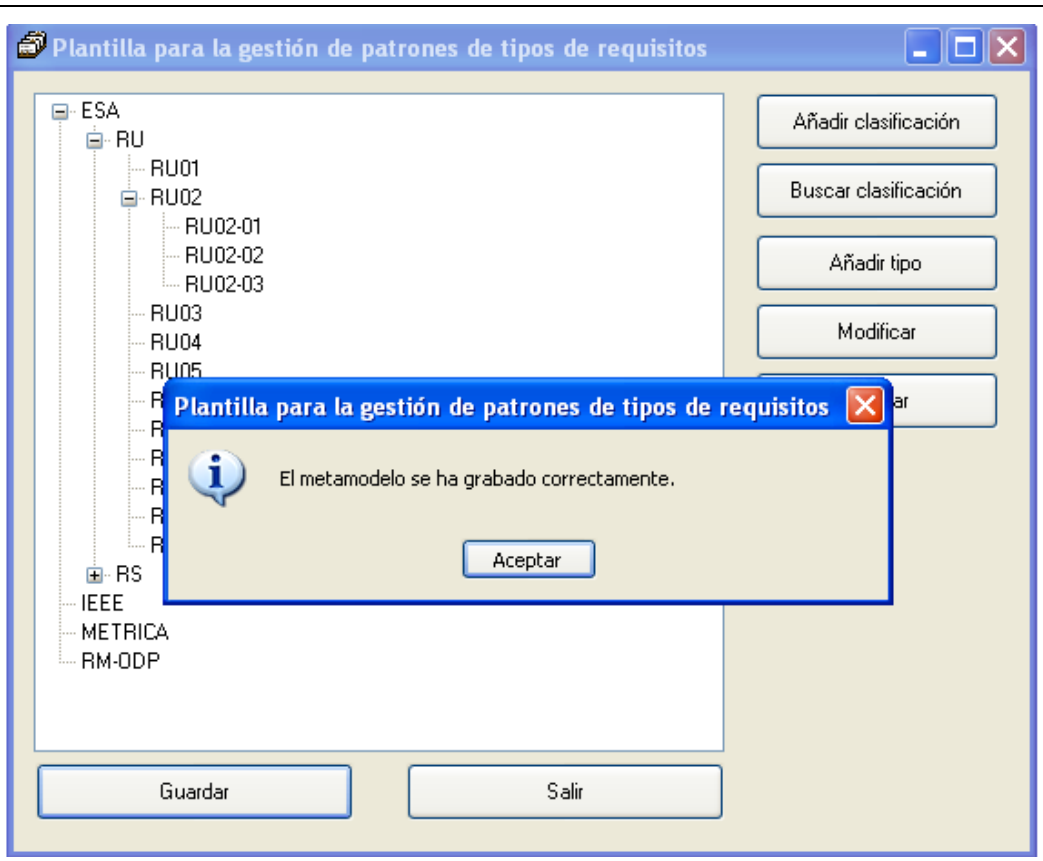

Ilustración 15 : Mensaje Plantilla Gestión Patrones Tipos Requisitos

8.3.2. Pantalla nodo clasificación

Pantalla que aparece cuando se pincha sobre el botón añadir clasificación o modificar (si previamente se ha seleccionado un nodo clasificación) en la pantalla *Plantilla para la gestión de patrones de tipos de requisitos*.

En la parte izquierda de la pantalla aparecen los campos descriptores del nodo.

- Identificador. Identificador unívoco de una clasificación, es por esto que no pueden existir dos clasificaciones con idéntico identificador.
- Nombre. Nombre asociado a la clasificación.

Los atributos obligatorios son el nombre y el identificador, marcados con un asterisco, es por esto que deben ser completados antes de guardar.

- Autor. Empresa y/o persona creadora de la clasificación.

- Descripción. Breve aclaración acerca del enfoque de la clasificación, así como cualquier relevancia referente a ésta.

- Fecha de creación. Fecha, en la que tuvo lugar, la creación de la clasificación.

En la parte derecha de la pantalla aparecen los siguientes botones.

- Botón guardar. Añade una clasificación al metamodelo del árbol de la pantalla principal. La escritura sobre el fichero xml se produce únicamente cuando se pulsa sobre el botón guardar de la pantalla *Plantilla para la gestión de patrones de tipos de requisitos*.

- Botón salir. Cierra el formulario.

A continuación, se muestra la ilustración 16 que representa la pantalla *Nodo Clasificación*, con todos los componentes que se han explicado anteriormente.

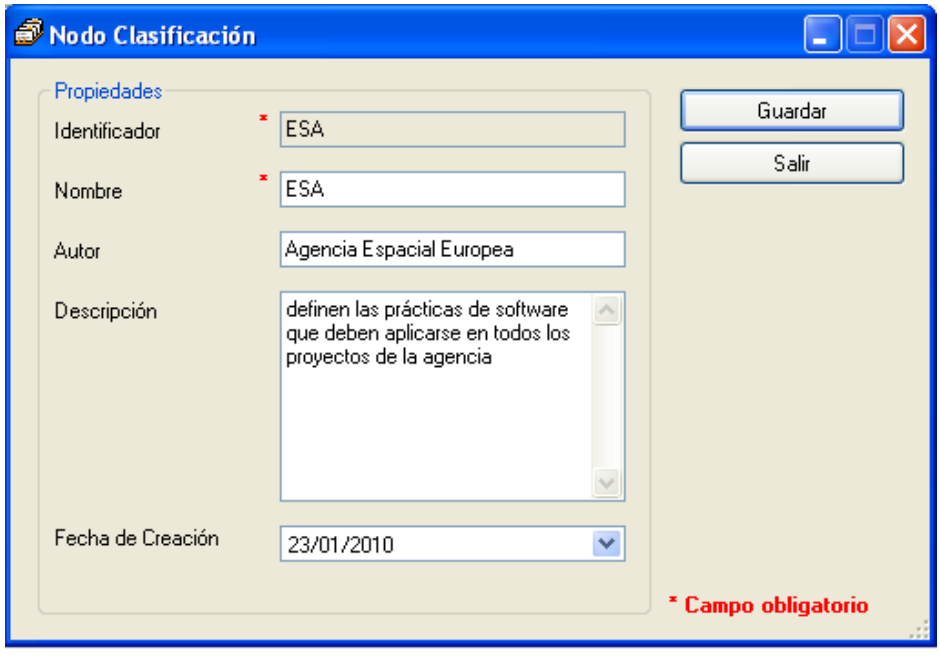

Ilustración 16: Pantalla Nodo Clasificación

Una restricción implementada, es el no poder crear dos clasificaciones con el mismo identificador, advirtiendo de esta situación mediante la alerta 'La clasificación ya existe', reflejada en la ilustración 17. Cuando el usuario desea modificar una clasificación, el campo Identificador es de lectura para evitar duplicados.

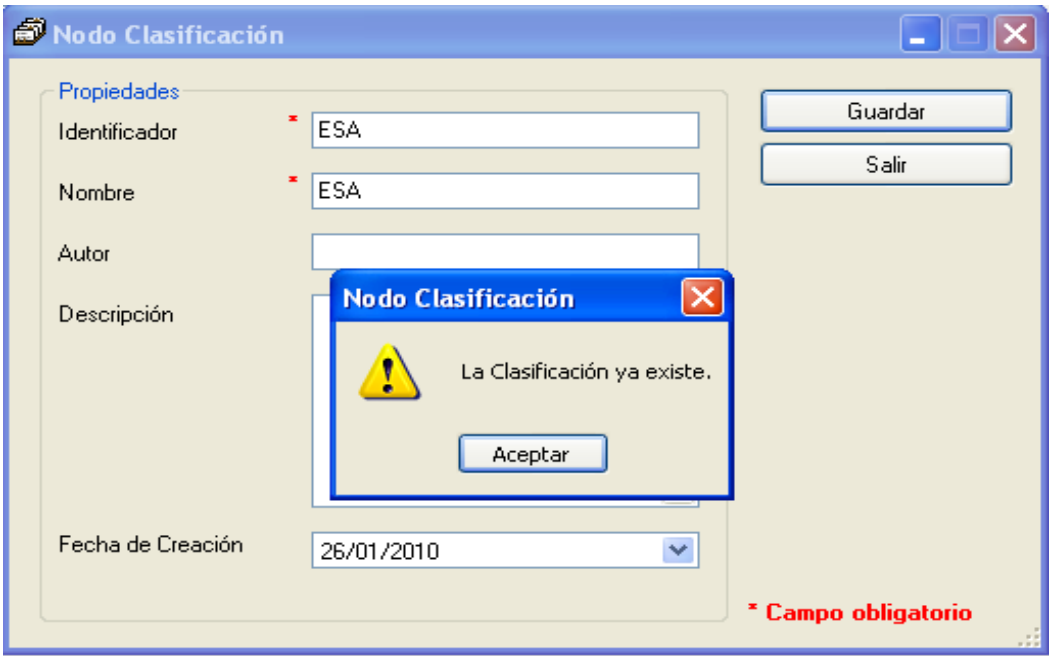

Ilustración 17: Alerta Clasificación Existente

La aplicación contiene una serie de alertas informativas advirtiendo del resultado de la operación invocada en cada momento. Por ejemplo, cuando se pincha sobre el botón guardar de la pantalla *Nodo Clasificación*, se muestra el mensaje reflejado en la ilustración 18.

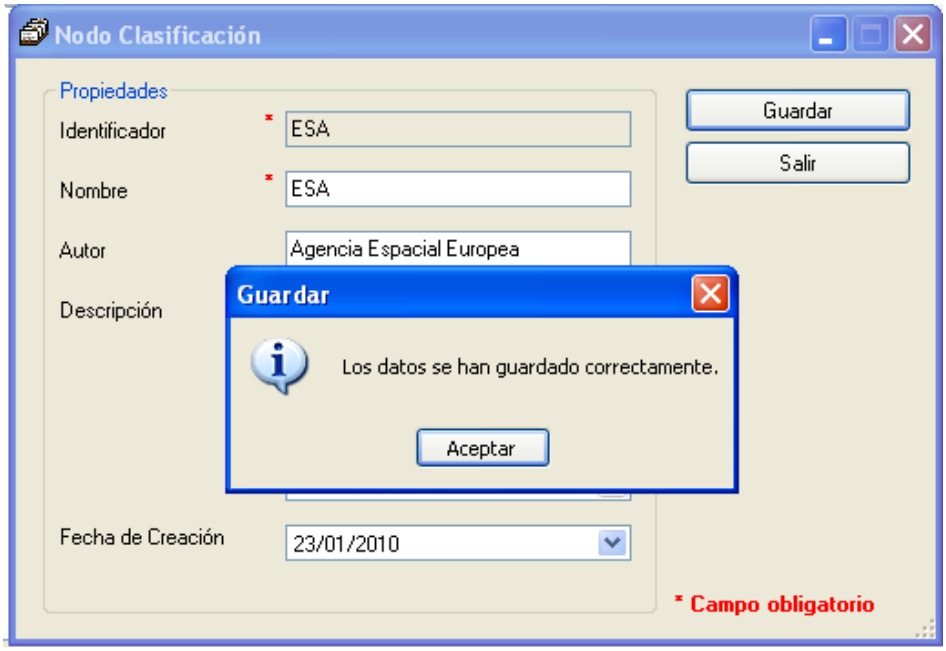

Ilustración 18: Aviso Nodo Clasificación

8.3.3. Pantalla nodo tipo

Pantalla que aparece cuando se pincha sobre el botón añadir tipo o modificar (si previamente se ha seleccionado un nodo tipo) en la pantalla *Plantilla para la gestión de patrones de tipos de requisitos*.

En la pantalla *Nodo Tipo* se muestran todos los campos asociados a dicho nodo (son nodos tipo los asociados a una clasificación y aquellos subtipos pertenecientes a un tipo).

Es imprescindible seleccionar un nodo padre en el árbol para añadir un nodo tipo asociado a éste.

En la parte derecha de la pantalla aparece información relevante del nodo.

- Identificador del padre. Identificador del nodo padre.
- Nivel de jerarquía del tipo. Profundidad que tiene el nodo actual en la clasificación a la que pertenece.
- Botón guardar. Añade un tipo al nodo “padre” del árbol seleccionado en la pantalla principal.
- El botón salir. Cierra el formulario.

En la parte izquierda de la pantalla aparecen los campos asociados al nodo.

- Identificador. Identificador del nodo tipo.
- Nombre. Nombre asociado al tipo.

Los atributos obligatorios son el nombre y el identificador, marcados con un asterisco, es por esto que deben ser completados antes de guardar.

- Descripción. Breve definición del nodo tipo, así como cualquier información relevante referente a éste.
- Botón Gestión Relaciones. Permite relacionar el nodo tipo con otros nodos tipo de la misma clasificación, con una pequeña descripción de esta relación. En el apartado siguiente, se explica más detalladamente, la pantalla asociada a la gestión de relaciones.

- Botón Elementos Externos. Permite relacionar el nodo tipo con archivos adjuntos junto con una pequeña descripción. En el apartado siguiente, se explica más detalladamente, la pantalla asociada a los elementos externos.

- Requisitos individuales. Permite añadir requisitos individuales. Los requisitos están contenidos dentro de los subtipos, los cuales, pertenecen a un tipo determinado.

A continuación, se muestra la ilustración 19 que representa la pantalla *Nodo Tipo*, con todos los componentes que se han descrito anteriormente.

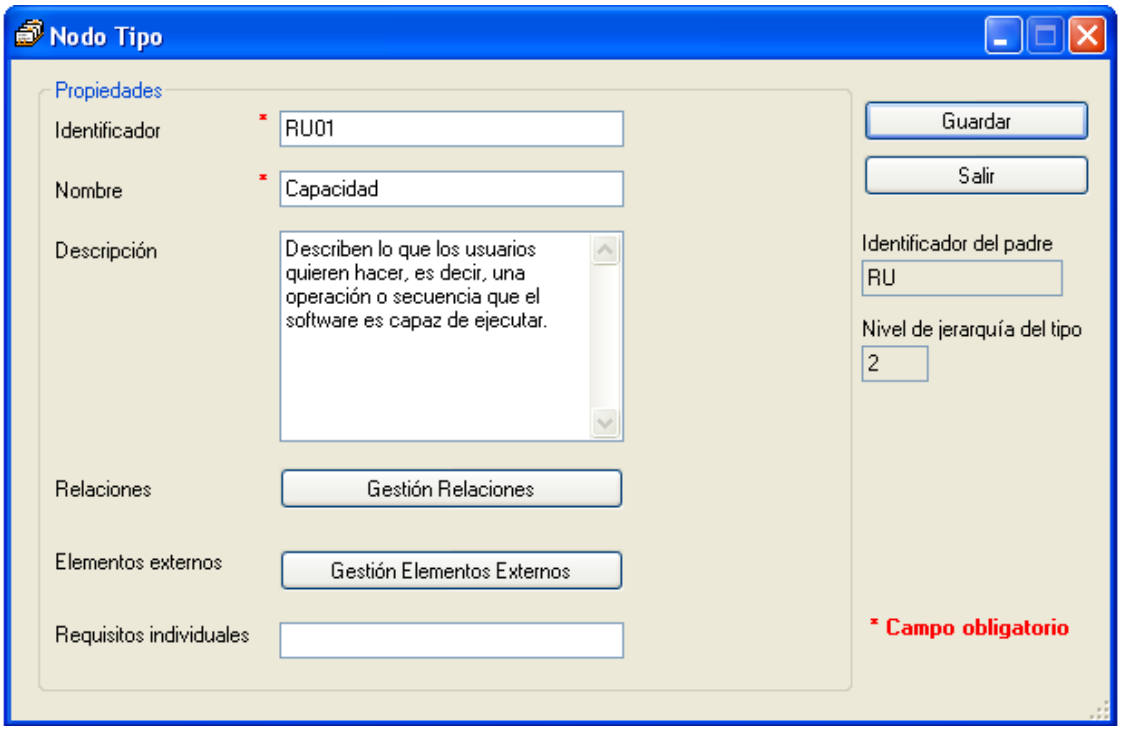

Ilustración 19: Pantalla Nodo Tipo

8.3.4. Pantalla gestión relaciones

Cuando se pincha sobre el botón Gestión Relaciones de la pantalla *Nodo Tipo*, se permite relacionar el nodo tipo visualizado con otros nodos tipo de la misma clasificación mediante la pantalla de *Gestión Relaciones.*

En la parte superior, aparece un árbol donde se muestran todos los nodos tipo que penden de la clasificación a la que pertenece el nodo tipo visualizado.

En la parte inferior, se muestra una tabla con todas las relaciones almacenadas, donde aparecen el identificador del nodo tipo relacionado y una breve descripción (opcional).

En la parte derecha de la pantalla aparecen los botones necesarios para añadir o eliminar una relación.

- Añadir una nueva relación. Se selecciona cualquier nodo del árbol, excepto el nodo que identifica a la clasificación, y luego se pulsa sobre el botón añadir. En ese momento, se insertará un nuevo registro en la tabla.
- Eliminar alguna relación, previamente establecida. Se debe seleccionar el registro o registros de la tabla, que se deseen quitar, y pulsar la tecla SUPR ó el botón eliminar.

En la parte inferior de la pantalla aparecen los botones que interactúan con la aplicación.

- Botón guardar. Asocia todo el conjunto de relaciones, mostradas en la tabla, al nodo tipo visualizado.
- Botón salir. Cierra el formulario.

A continuación, se muestra la ilustración 20 que representa la pantalla *Gestión Relaciones Nodo Tipo*, con todos los componentes que se han descrito anteriormente.

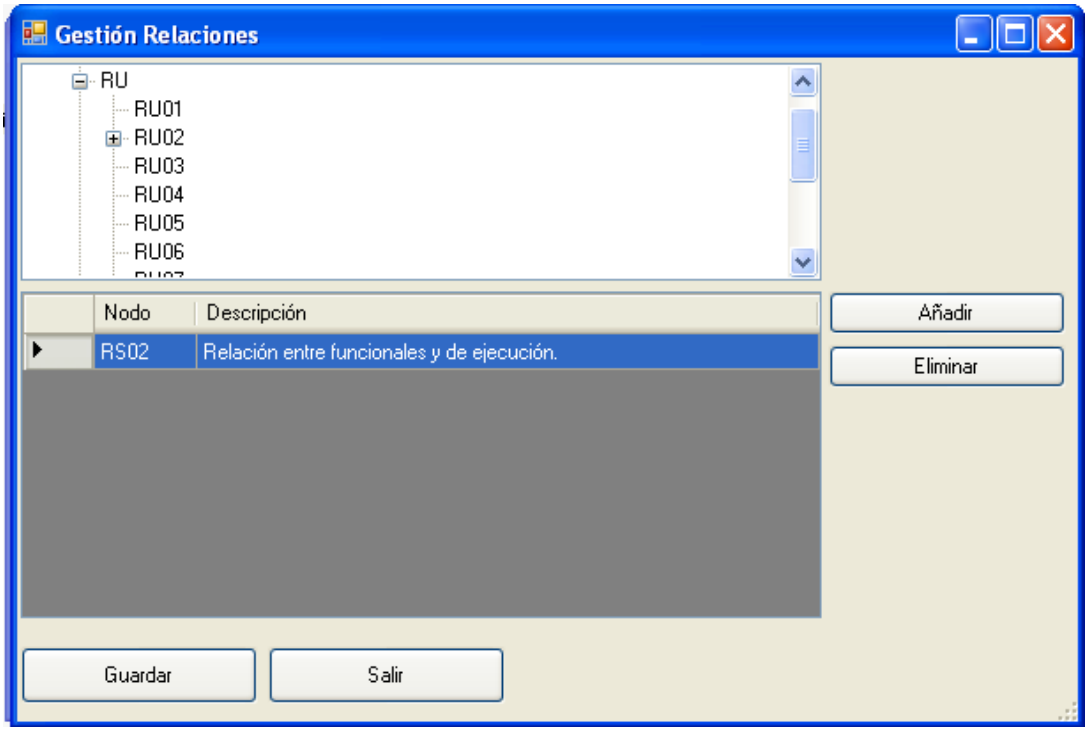

Ilustración 20: Pantalla Gestión Relaciones Nodo Tipo

8.3.5. Pantalla gestión elementos externos

Cuando se pincha sobre el botón Elementos Externos de la pantalla *Nodo Tipo*, se permite relacionar el nodo tipo visualizado con cualquier fichero externo, almacenado en disco, que contenga información relacionada con el nodo.

Es necesario haber introducido previamente un identificador al nodo tipo antes de proceder a la gestión de elementos externos (esto es imprescindible puesto que la carpeta contenedora de los archivos se nombra con este identificador). En caso contrario, se alerta de la situación con un mensaje mostrado en la ilustración 21.

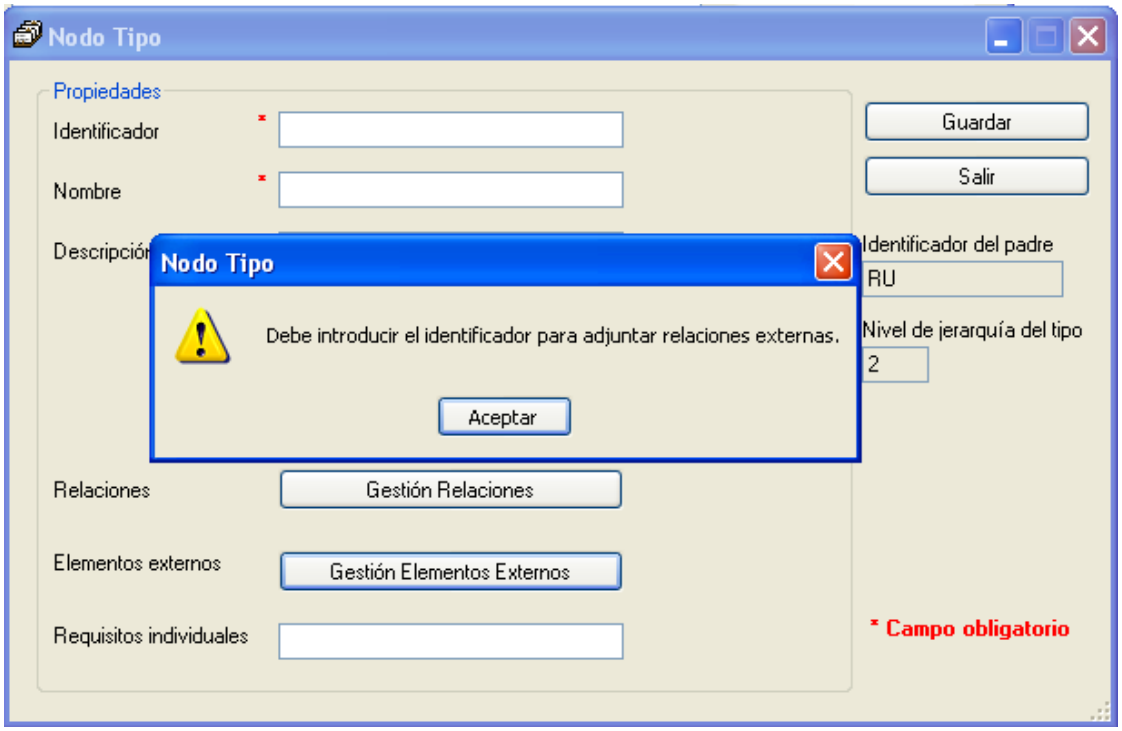

Ilustración 21: Alerta Gestión Elementos Externos Nodo Tipo

A continuación, se explican todos los elementos de la pantalla de *Gestión Elementos Externos* y se muestra en la ilustración 22 todo el conjunto de componentes de la misma.

En la parte superior aparece un listado de elementos externos, relacionando el nodo tipo con un archivo y una breve descripción (opcional).

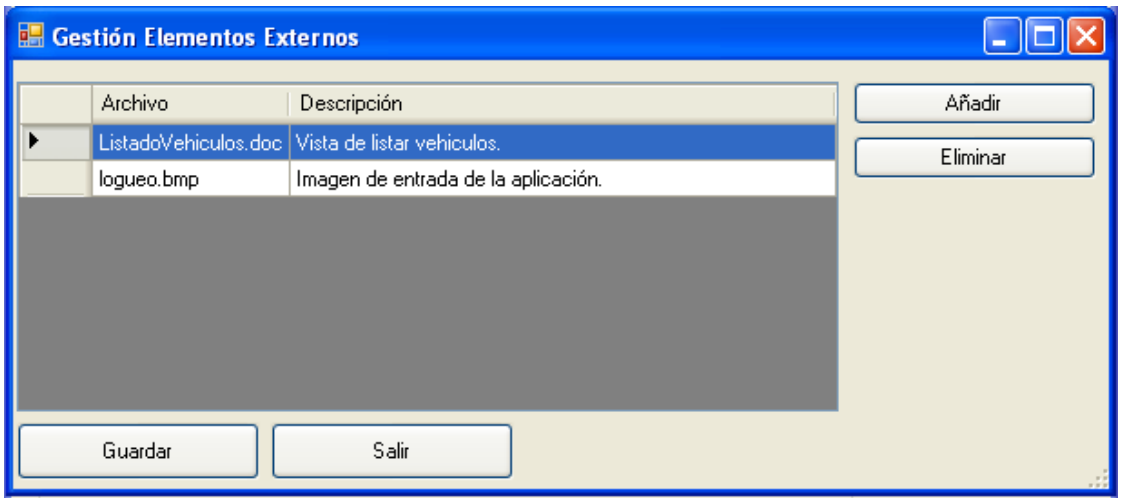

Ilustración 22: Pantalla Gestión Elementos Externos Nodo Tipo

En la parte derecha de la pantalla aparecen los botones necesarios para añadir o eliminar un fichero.

- Añadir un fichero. Se debe pinchar sobre el botón añadir. A continuación, aparecerá el explorador de windows para poder elegir el archivo, tal como se observa en la ilustración 23. En ese momento, se insertará un nuevo registro en la tabla de elementos externos.

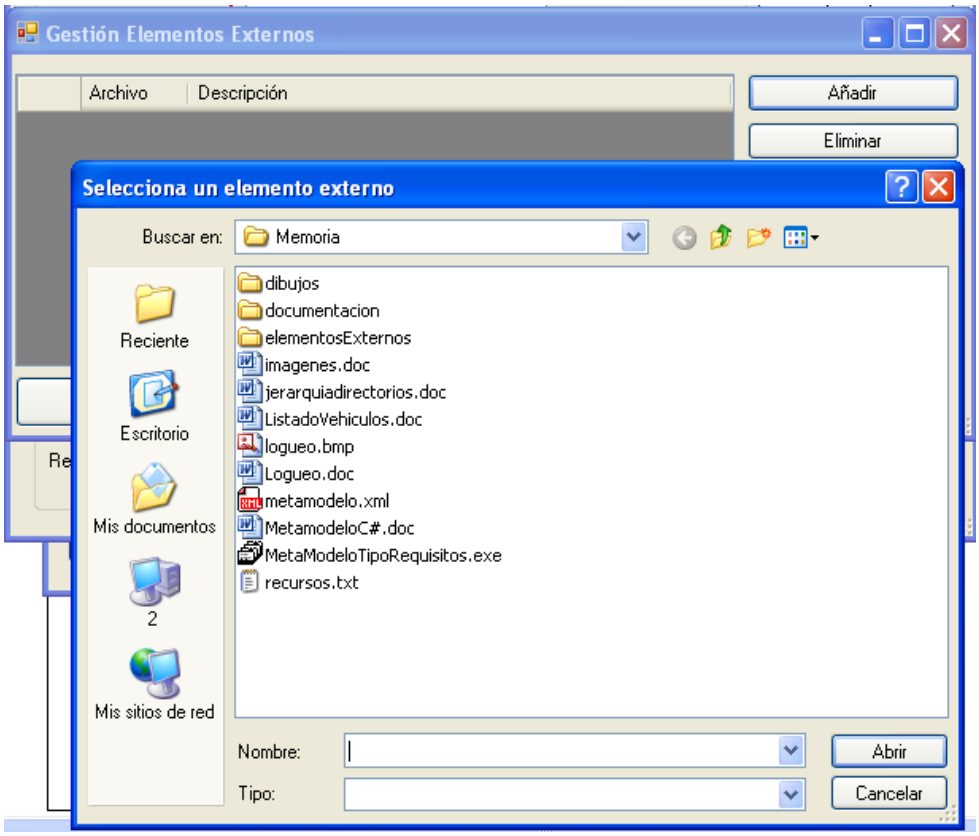

Ilustración 23 : Añadir Elemento Externo Nodo Tipo

Si el fichero a adjuntar ya estuviese relacionado previamente con ese nodo tipo, aparecería un mensaje, advirtiendo de ello, mostrado en la ilustración 24.

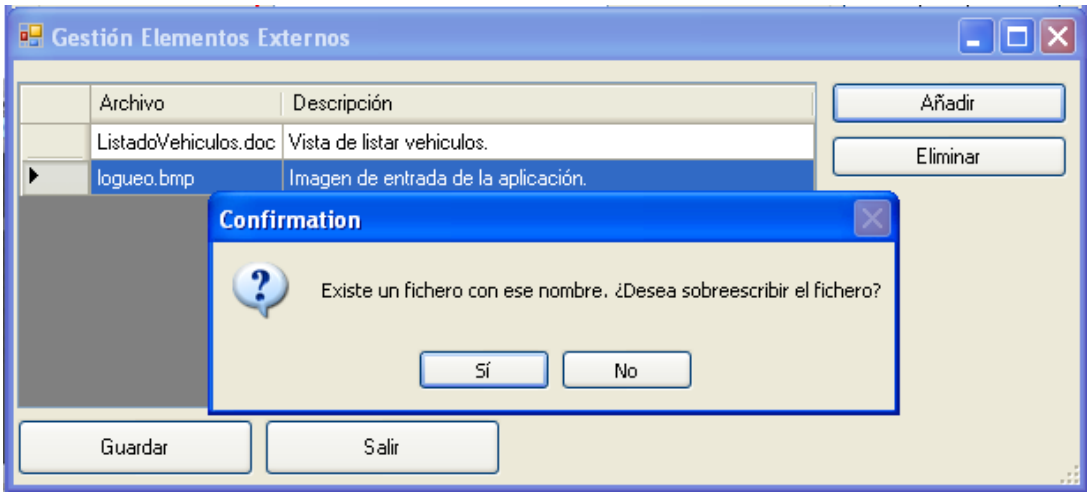

Ilustración 24 : Sobrescribir Elemento Externo Nodo Tipo

A continuación, se explican las acciones asociadas a la tabla de gestión de elementos externos.

- Visionar un fichero previamente adjuntado. Se debe hacer doble click sobre la celda contenedora del nombre del fichero. De esta manera, se abrirá el fichero con el programa asociado a la extensión del archivo.
- Eliminación de elemento externo. Se debe seleccionar el registro o registros de la tabla, que se deseen quitar, y pulsar a la tecla SUPR ó sobre el botón eliminar.

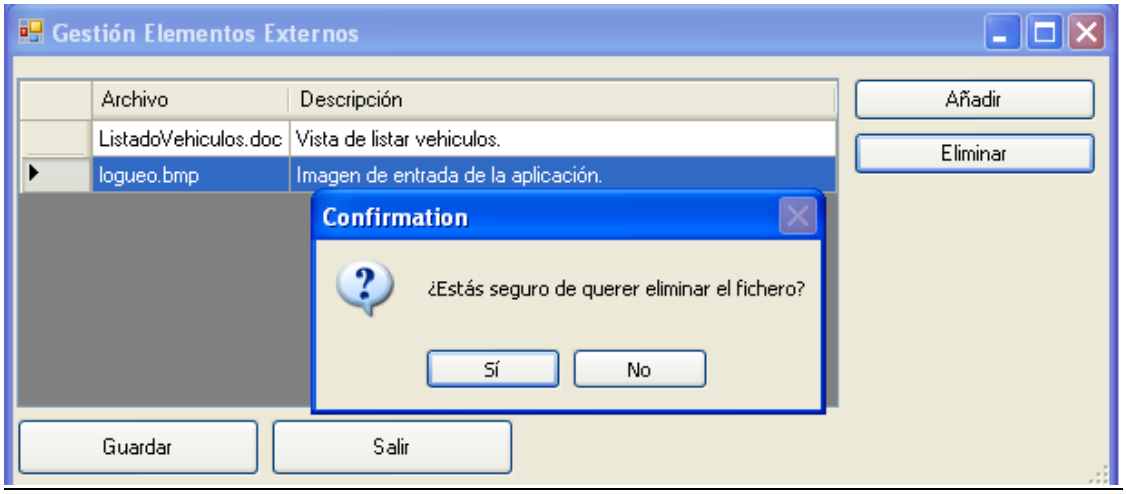

Ilustración 25: Eliminación Elemento Externo Tipo

Antes de proceder a la eliminación, se debe confirmar, respondiendo al mensaje, mostrado en la ilustración 25, afirmativamente.

En la parte inferior de la pantalla aparecen los botones que interactúan con la aplicación.

- Botón guardar. Asocia todo el conjunto de ficheros, mostrados en la tabla, al nodo tipo visualizado.

- Botón salir. Cierra el formulario.

8.3.5.1. Directorio almacén de elementos externos

El directorio, donde se almacenan todos los ficheros considerados elementos externos, se compone de la estructura explicada a continuación.

El directorio raíz se denomina elementosExternos y se encuentra contenido en el mismo directorio que el .exe de la aplicación.

A partir de él, se crean los directorios, nombrados con los identificadores de cada nodo tipo, necesarios para almacenar el fichero adjuntado.

El directorio, contenedor de cada archivo, es un subárbol al que pertenece el nodo tipo asociado al elemento externo, dentro del conjunto de su clasificación.

Por ejemplo, un archivo adjuntado para el nodo tipo Interacción persona-ordenador (RU03), estaría contenido en el árbol de directorios que aparece en la ilustración 26.

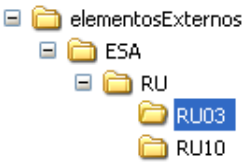

Ilustración 26 : Directorio Elementos Externos

La eliminación de un nodo tipo o de un elemento externo conlleva la limpieza del almacén, que se explica a continuación.

- Eliminación de nodo tipo en la pantalla *Plantilla para la gestión de patrones de tipos de requisitos*, mostrada en ilustración 27.

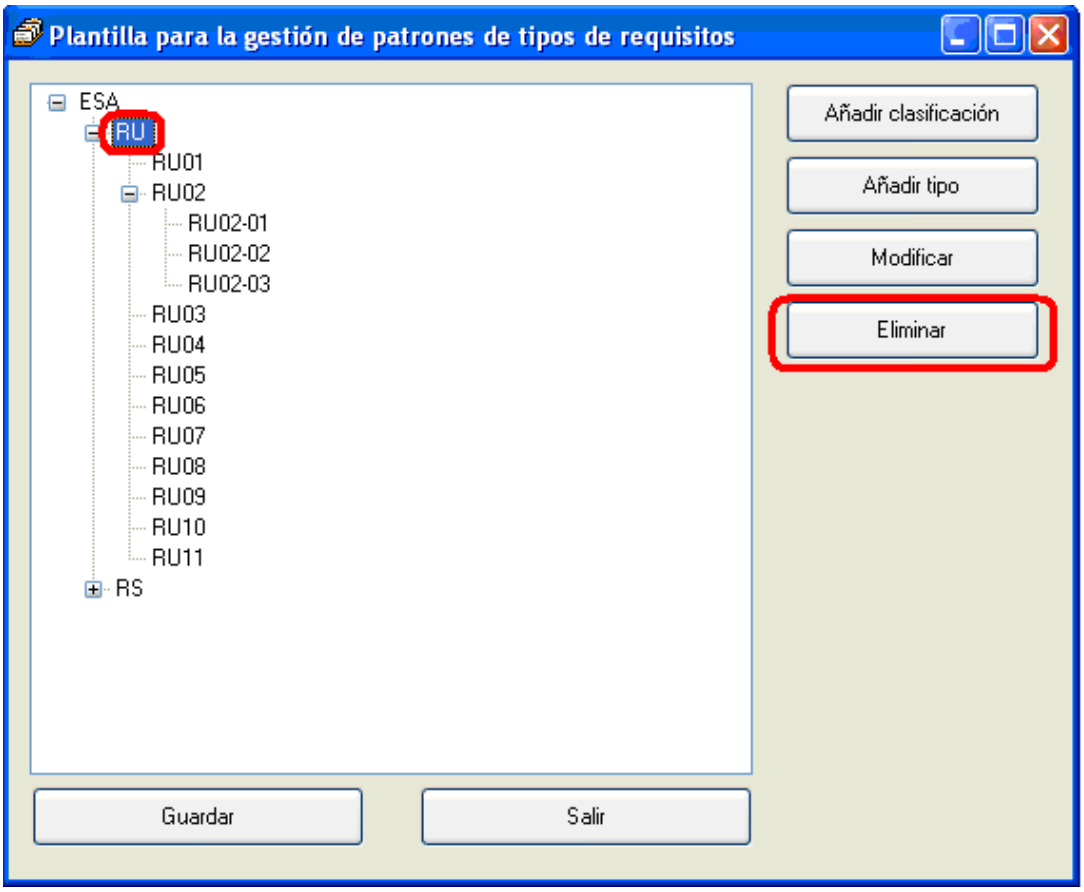

Ilustración 27: Eliminación Nodo Plantilla Gestión Patrones Tipos Requisitos

En este caso, se procede a eliminar del almacén, todos los directorios y ficheros, que penden del directorio nombrado como el nodo tipo seleccionado, así como él mismo. Además se procede a inspeccionar todos los directorios, a partir, del directorio elementosExternos, para eliminar todos aquellos que no contengan ninguna información (esto es que no contengan ficheros o cuyos directorios hijos no almacenen ningún archivo).

- Eliminación de elemento externo en la pantalla *Gestión Elementos Externos,* mostrada en ilustración 28.

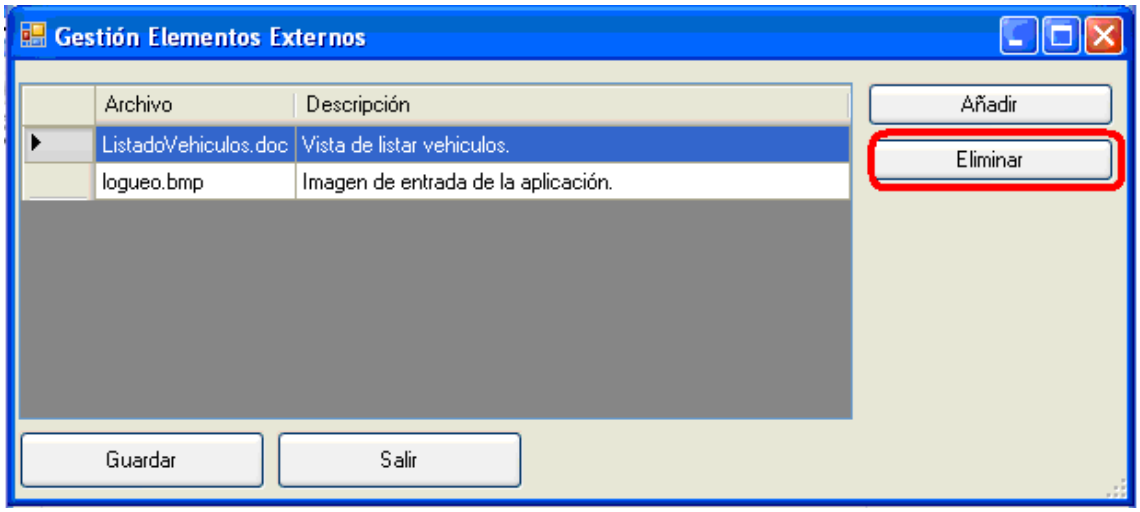

Ilustración 28: Eliminación Elemento Externo

En este caso, se elimina el fichero del directorio contenedor. Además se procede a inspeccionar todos los directorios, a partir, del directorio elementosExternos, para eliminar todos aquellos que no contengan ninguna información (esto es que no contengan ficheros o cuyos directorios hijos no almacenen ningún archivo).

8.3.6. Pantalla buscador clasificaciones

Pantalla que aparece cuando se pincha sobre el botón buscar clasificación en la pantalla *Plantilla para la gestión de patrones de tipos de requisitos*.

La búsqueda de clasificaciones se puede filtrar mediante criterios de búsqueda y los resultados obtenidos pueden ordenarse.

La ilustración 29 muestra la pantalla *Buscador Clasificaciones*.

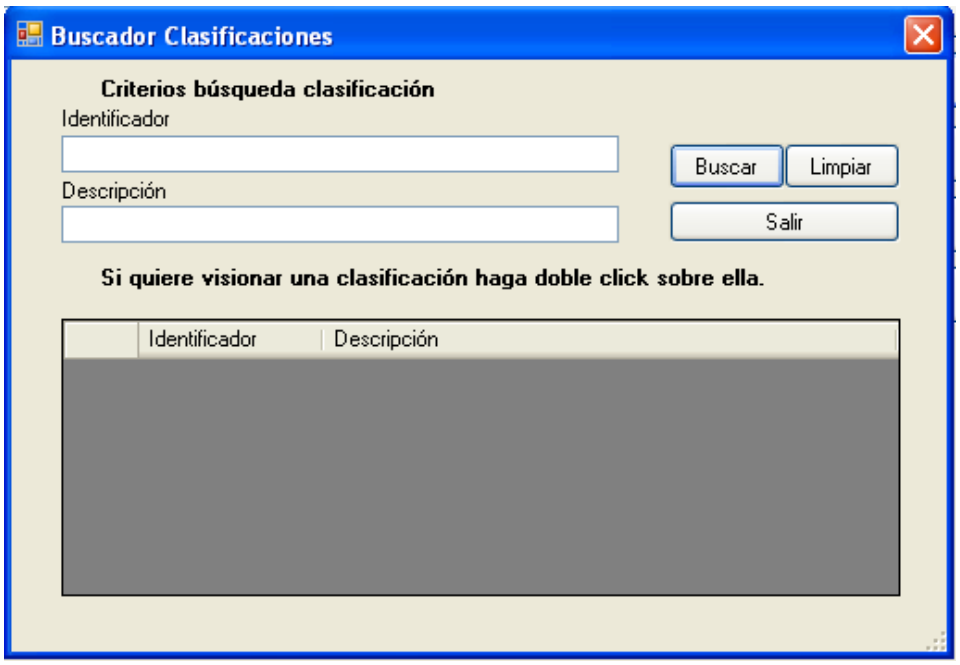

Ilustración 29: Pantalla Buscador Clasificaciones

En la parte superior izquierda de la pantalla aparecen los criterios de búsqueda.

- Identificador. Filtro aplicable al identificador de las clasificaciones contenidas en el metamodelo.

- Descripción. Filtro aplicable a la descripción de las clasificaciones contenidas en el metamodelo.

Ambos filtros no son sensibles a mayúsculas/minúsculas a la hora de realizar la búsqueda.

A continuación, en la tabla 11, se explican los resultados obtenidos, en función de los filtros de búsqueda introducidos.

Filtro Identificador	Filtro Descripción	Resultado
Vacío	Vacío	Todas las clasificaciones contenidas en el metamodelo.
No vacío	Vacío	Todas las clasificaciones cuyos identificadores contengan el filtro introducido.
Vacío	No vacío	Todas las clasificaciones cuyas

		descripciones contengan el filtro introducido.
No vacío	No vacío	Todas las clasificaciones cuyos identificadores y descripciones contengan los filtros introducidos.

Tabla 11: Resultados Buscador Clasificaciones

En la parte superior derecha de la pantalla aparecen los botones que interactúan con la aplicación.

- Botón buscar. Realiza la búsqueda de aquellas clasificaciones, contenidas en el metamodelo, que cumplan con los criterios de búsqueda introducidos.

- Botón limpiar. Limpia las cajas de texto asociadas a los criterios de búsqueda.

- Botón salir. Cierra el formulario.

En la parte inferior aparece el listado de las clasificaciones obtenidas en la búsqueda. Cada registro resultante contiene el identificador y la descripción de cada clasificación. La ilustración 30 muestra un ejemplo de listado de clasificaciones.

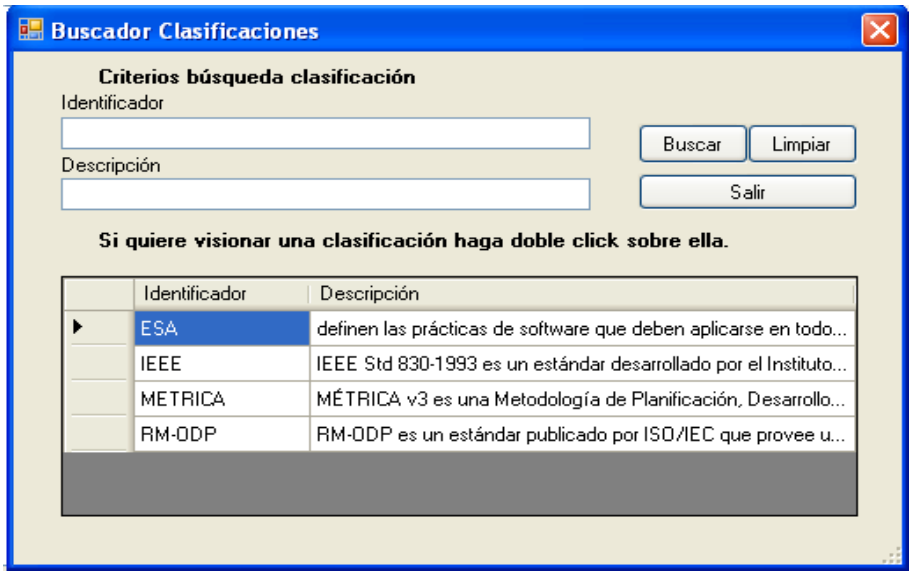

Ilustración 30: Listado Clasificaciones Filtradas

Al posicionar el puntero sobre cualquier descripción de la tabla de resultados aparece su descripción completa en modo emergente. En la ilustración 31 se puede observar un tooltip descriptor de RM-ODP.

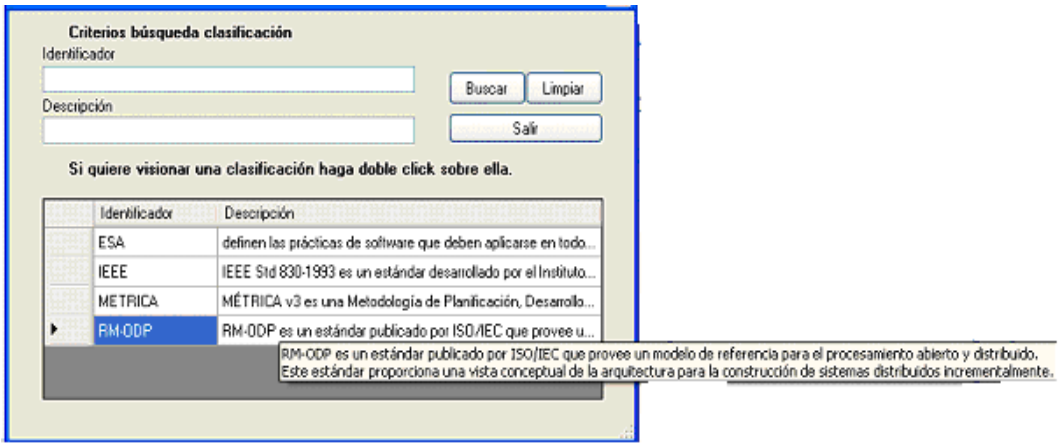

Ilustración 31: Tooltip Descriptor Clasificación

Si se quiere visionar una clasificación, se tiene que hacer doble click en la fila que muestre su información.

Si se quiere ordenar las clasificaciones, por orden alfabético, usando el identificador, se tiene que hacer doble click sobre la columna Identificador.

La ilustración 32 refleja un ejemplo de ordenación por el atributo identificador de clasificación.

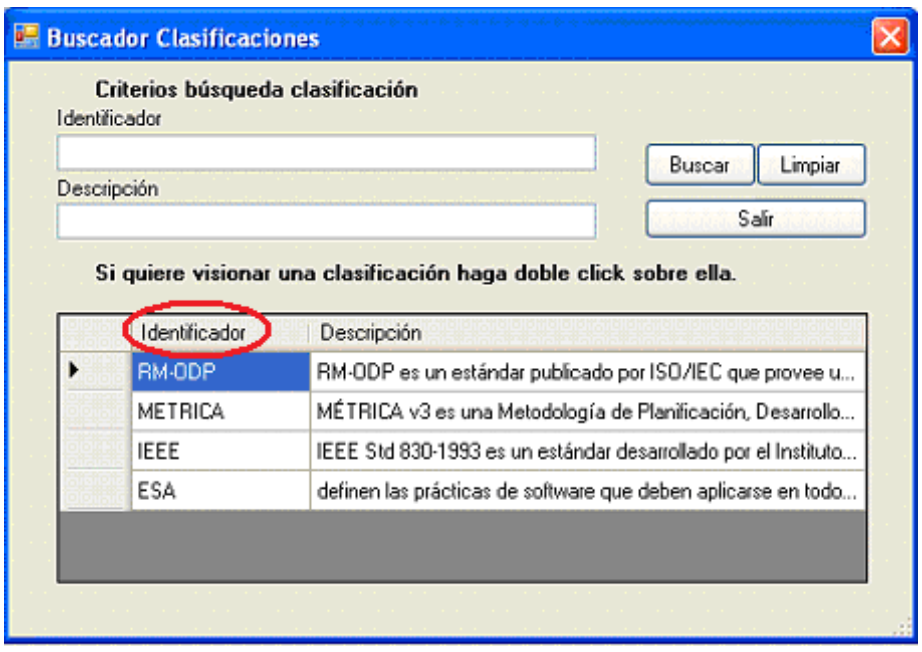

Ilustración 32: Ordenación Alfabética Descendente Por Identificador Clasificaciones

8.4. Aspectos de la programación

La plataforma .NET y el lenguaje de programación C# han sido necesarios para implementar la aplicación descrita.

En concreto, la plataforma .NET [NET 10] de Microsoft es un componente de software que puede ser añadido al sistema operativo Windows.

Se ha escogido .NET por ventajas tan relevantes como la independencia de la plataforma y del lenguaje de programación (una misma aplicación puede desarrollarse en múltiples lenguajes, lo que permite la reutilización del código).

C# [C# 01] es el lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft para el marco de trabajo.NET, escogido para la programación de la aplicación.

Además, se ha usado Microsoft Visual Studio 2008 Professional [Microsoft Visual Studio 2008 Professional] como entorno de desarrollo integrado, donde cada formulario creado se corresponde con cada una de las pantallas explicadas anteriormente. Las clases implementadas se explican a continuación.

8.4.1. Clases

Las clases desarrolladas, con sus respectivos métodos y propiedades, son las que se muestran en el diagrama de clases que aparece en la ilustración 33. A continuación, se explican brevemente cada una de las clases implementadas.

- **CLMetamodelo**, es un arraylist tipado con objetos clasificación, es decir, la estructura contenedora de todas las clasificaciones.

- **FicheroXML** es la clase encargada de realizar la lectura y escritura de todas las clasificaciones contenidas en el metamodelo. El almacenamiento se hace sobre un fichero xml (metamodelo.xml), localizado en el directorio donde se encuentra el ejecutable de la

aplicación. El procesamiento del mismo se detalla en un apartado posterior (Tratamiento fichero xml).

- **Clasificacion** es la clase que implementa objetos clasificación. Contiene un arraylist de objetos tipo así como unas serie de atributos string (identificador, nombre, descripción, autor, fechaCreacion, ordenacion).

  El identificador de la clasificación tiene que ser unívoco.

  El atributo ordenación es necesario para determinar el tipo de ordenación (ascendente o descendente) en el buscador de clasificaciones y ordenar correctamente un listado de clasificaciones determinado.

  - Buscador de clasificación.

    El buscador de clasificaciones se ha desarrollado teniendo en cuenta una serie de aspectos que se detallan a continuación.

    Se ha utilizado un dataGridView, control de .NET, que muestra los datos en una cuadrícula personalizable. Ésta se alimenta de un datasource, el cual, establece un origen de datos (conjunto de clasificaciones obtenidas en la búsqueda realizada con los filtros aplicados por el usuario).

    Para realizar la ordenación alfabética de las clasificaciones por identificador, ha sido necesaria la implementación de la interfaz IComparer en la clase Clasificacion.

- **Tipo** es la clase que implementa objetos Tipo. Se compone de una serie de atributos string (descripción, identificador, nivel, nombre, padre, requisitos individuales) así como de listas (elementos externos, relaciones y subtipos).

El atributo nivel sirve para identificar el nivel de profundidad de cada nodo en el árbol de la clasificación a la que pertenece éste.

El atributo padre indica el identificador del nodo padre al que pertenece el nodo.

A continuación, se explica brevemente los listados contenidos en cada objeto Tipo.

  - Elementos externos. Lista de objetos pertenecientes a la clase ElementoExterno. Almacena todos los archivos enlazados con un nodo Tipo dado.

  - Relaciones. Lista de objetos pertenecientes a la clase Relacion. Almacena todas aquellas relaciones, que un nodo Tipo dado, puede tener con otro nodo Tipo que pertenece a su misma clasificación.

  - Subtipos. Lista de objetos pertenecientes a la clase Tipo. Contiene todos aquellos objetos Tipo que son hijos de un nodo Tipo dado.

- **ElementoExterno** es la clase encargada de guardar el nombre del archivo y una descripción de aquellos archivos adjuntados a un nodo Tipo.

- **Relacion** es la clase encargada de guardar el identificador del nodo relacionado y una descripción, cuando dos nodos Tipo están relacionados y están incluidos en la misma clasificación.

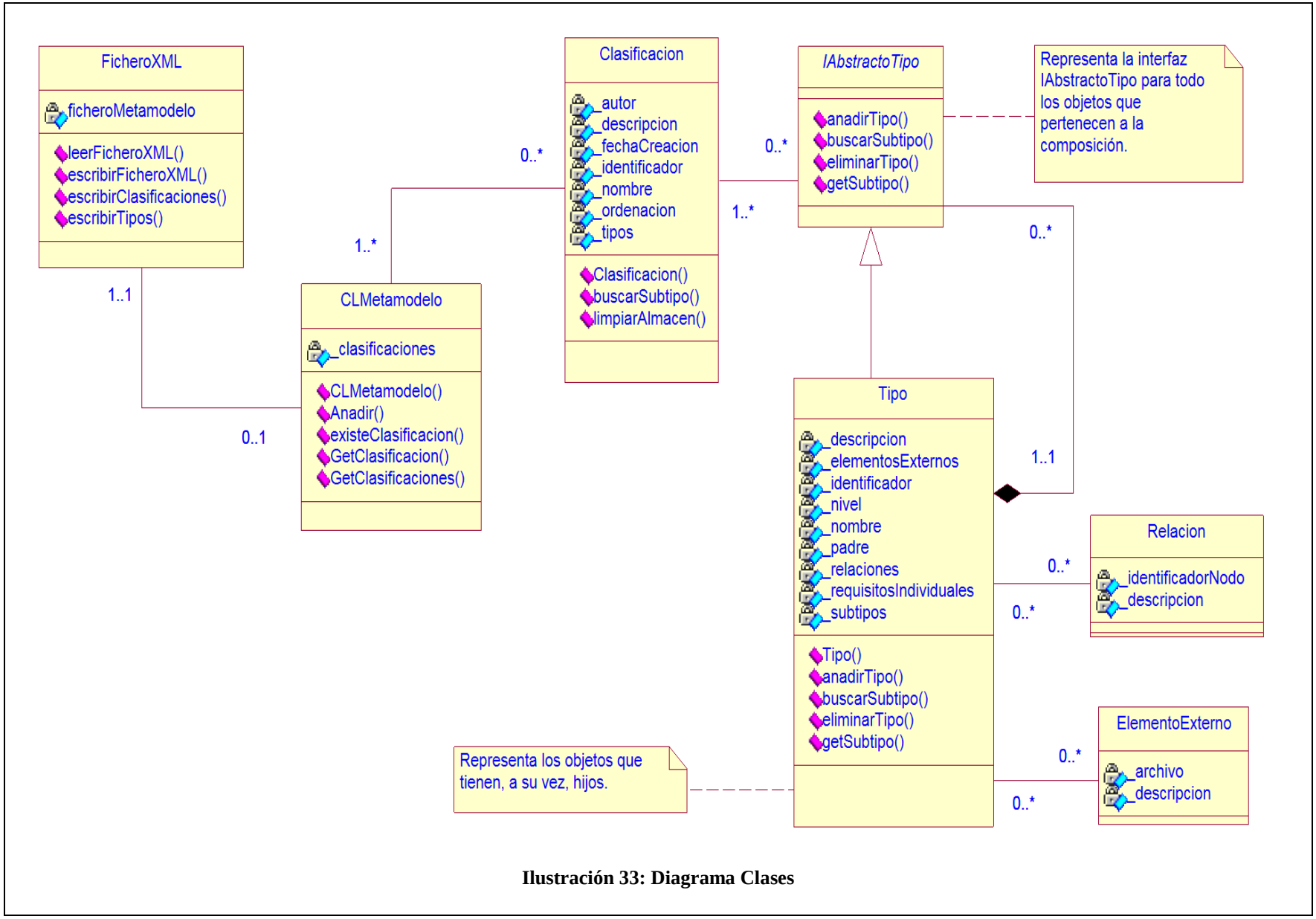

Ilustración 33: Diagrama Clases

Existe una relación de herencia donde los objetos Tipo comparten la estructura y el comportamiento del padre.

Existe una agregación de composición donde las partes sólo tienen existencia dentro del todo que las contiene. Además, la multiplicidad de la agregación, en la parte del todo, no puede exceder de uno.

Ambas relaciones (generalización y composición) son imprescindibles para implementar el patrón composite, necesario en el metamodelo.

8.4.2. Patrón composite

El patrón composite descrito por Gamma [Gamma 95] permite componer objetos en estructuras arborescentes para representar jerarquías parte-todo. Lográndose así trabajar indistintamente con objetos hoja como con objetos composición.

A continuación, en la ilustración 34 se muestra el diagrama de clases asociado al patrón composite aplicado a este proyecto.

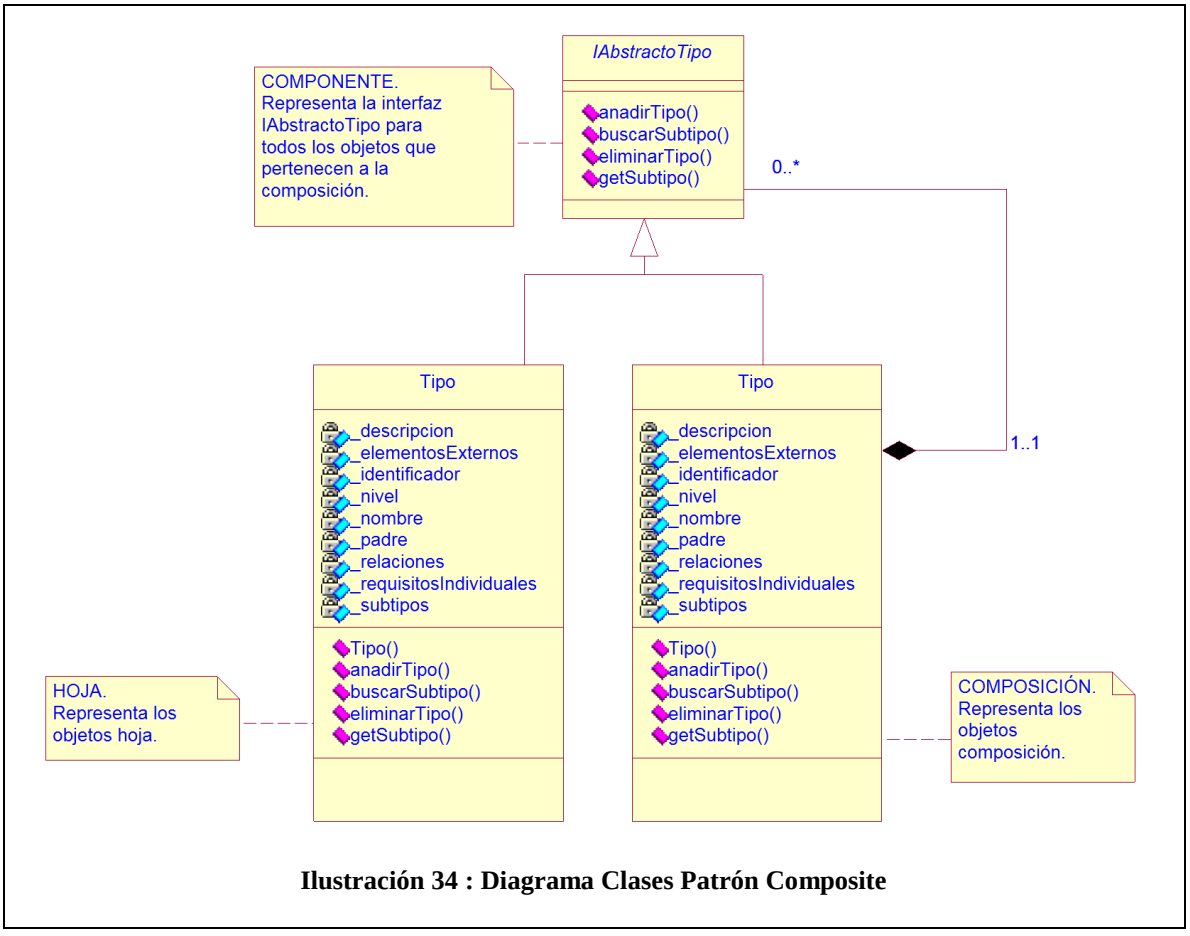

Ilustración 34 : Diagrama Clases Patrón Composite

Los elementos que participan en este patrón son:

- **Componente**. Declara la interfaz para todos los objetos que pertenecen a la composición.

- **Composición**. Representa los objetos que tienen, a su vez, hijos componentes. Son instancias de la clase Tipo.

- **Hoja**. Representa los objetos hoja de la composición, es decir, aquellos que no tienen hijos. Son instancias de la clase Tipo.

A continuación, en la ilustración 35 se muestra un ejemplo de estructura arborescente, donde aparecen objetos complejos, los cuales, a su vez están formados por objetos complejos y/o

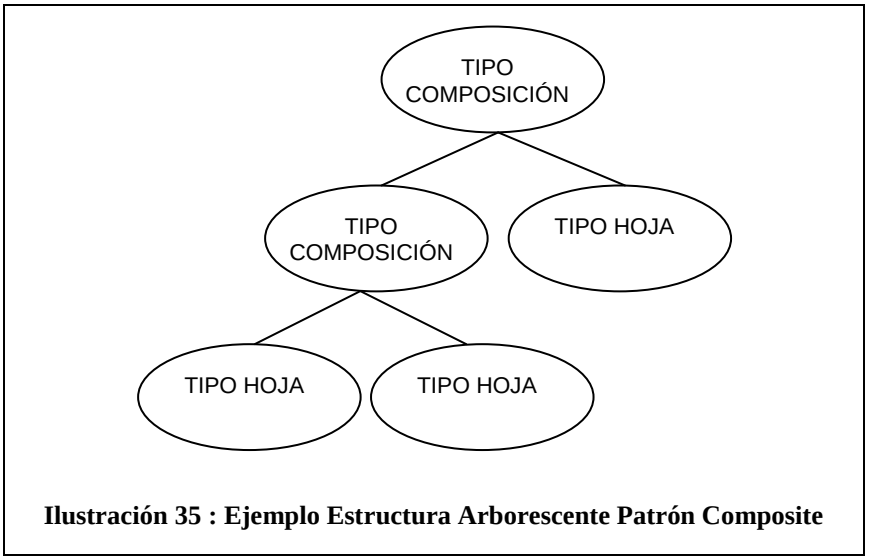

Ilustración 35 : Ejemplo Estructura Arborescente Patrón Composite

8.4.3. Formularios

Los formularios creados, con sus respectivos métodos y propiedades, se explican brevemente a continuación y se muestran en el diagrama de formularios de la ilustración 36.

- **Plantilla**. Formulario contenedor del treeview que gestiona el metamodelo completo: gestión de clasificaciones, gestión de tipos, almacenamiento metamodelo y salida de la aplicación.

- **FormNodoClasificacion.** Formulario que visualiza y administra un nodo clasificación determinado. El atributo usado es el nodo clasificación.

- **FormNodoTipo**. Formulario que visualiza y administra un nodo tipo determinado. Los atributos usados son el nodo tipo y la clasificación a la que pertenece éste.

- **FormElementosExternos.** Formulario que visualiza y administra los elementos externos asociados a un nodo tipo determinado. Los atributos usados son el nodo tipo y la clasificación a la que pertenece éste.

- **FormRelaciones.** Formulario que visualiza y administra las relaciones asociadas a un nodo tipo determinado. Los atributos usados son el nodo tipo y la clasificación a la que pertenece éste.

- **FormBuscadorClasificaciones**. Formulario que busca clasificaciones a partir de unos criterios de búsqueda. Los atributos usados son el metamodelo contenedor del conjunto de clasificaciones. Los resultados obtenidos en la búsqueda se pueden ordenar alfabéticamente por el identificador de las clasificaciones.

  Al visionar una clasificación obtenida en la búsqueda, se redirige al formulario Plantilla, refrescándose el treeview con la clasificación escogida.

  Para la implementación de dicho refresco se usa un delegado que permite una comunicación entre ambos formularios. Es decir, en el momento en que se hace doble click sobre una clasificación (en la lista de resultados del formulario FormBuscadorClasificaciones) se

actualiza automáticamente el treeview perteneciente al formulario Plantilla.

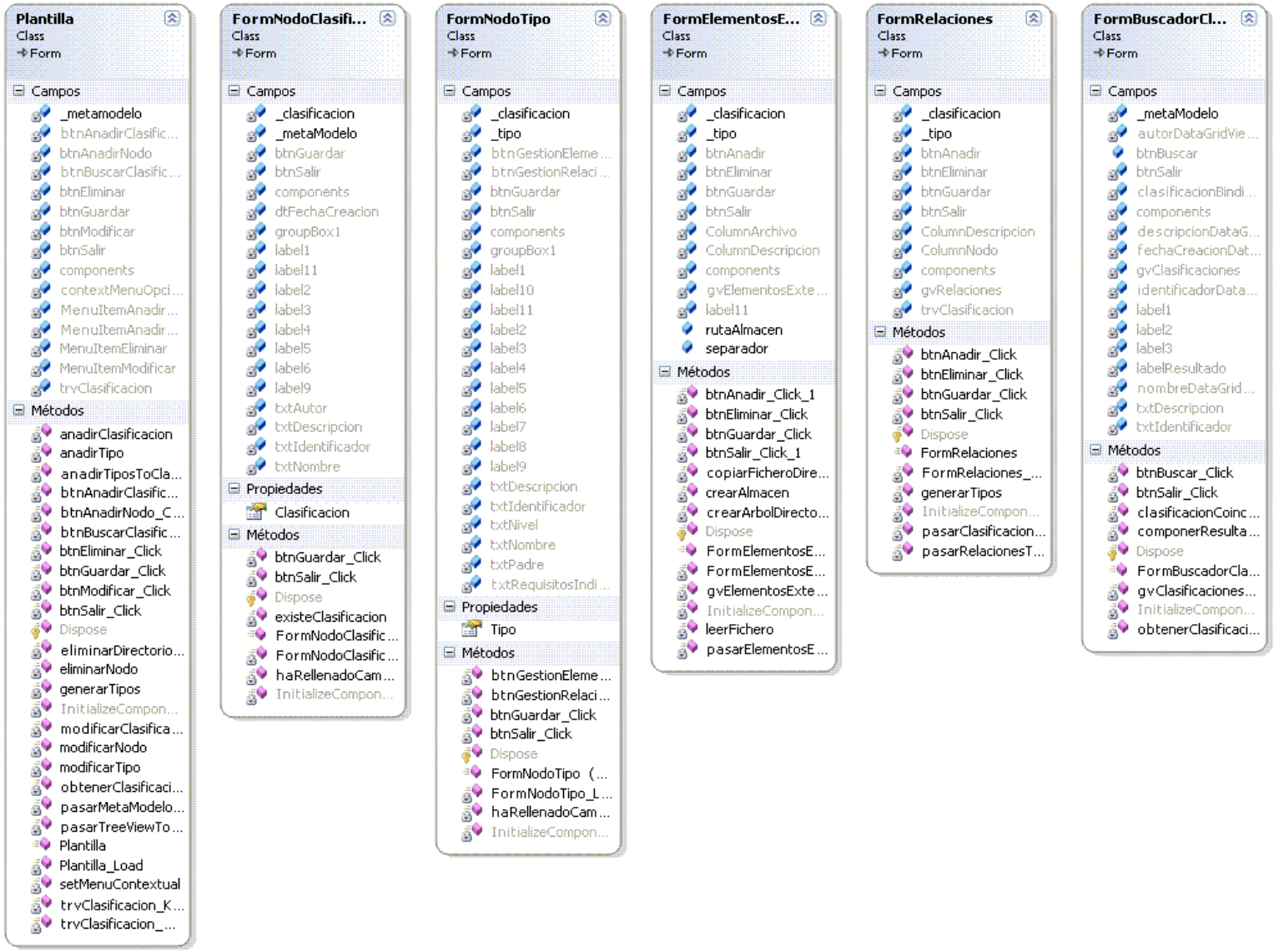

Ilustración 36: Diagrama Formularios

8.4.4. Recursividad

Ha sido necesario el uso de recursividad para la lectura, tratamiento y escritura del metamodelo.

A continuación, se enumeran las situaciones en la que se han tenido que implementar funciones recursivas.

- Lectura/escritura del fichero xml que alberga el metamodelo.

  Cada clasificación contiene un conjunto de tipos que, a su vez, pueden tener nodos hijos tipo. Para la lectura y/o escritura de estos tipos es necesaria una implementación recursiva.

- Obtención de un nodo tipo dentro del árbol que contiene al metamodelo.

  Para obtener un nodo tipo perteneciente a una clasificación del metamodelo, es necesario leer el árbol en profundidad mediante un algoritmo recursivo para llegar al nodo buscado.

- Limpieza del almacén de elementos externos debido a la eliminación de un nodo tipo.

  Para limpiar el directorio que alberga los archivos asociados a un nodo tipo, es necesario inspeccionar el árbol de directorios en profundidad, mediante una función recursiva, para eliminar todos aquellos directorios vacíos.

8.4.5. Tratamiento fichero xml

El fichero xml que conforma el metamodelo, del cual, lee y escribe la aplicación, contiene una serie de etiquetas que identifican a cada uno de los atributos así como estructuran la información para posibilitar el patrón composite.

Para procesar el documento xml es necesario un lenguaje (XPath) que permite construir y recorrer documentos de este tipo.

- XPath permite buscar y seleccionar en documentos xml, teniendo en cuenta la estructura jerárquica de éstos.

- La implementación realizada mediante el espacio de nombres XPath, contiene las clases que definen un modelo de cursor para navegar por los elementos de información xml.

El tratamiento en la lectura del fichero, para la carga inicial del árbol, se hace de manera recursiva puesto que la esencia del metamodelo se basa en el almacenamiento de todos los tipos asociados tanto a una clasificación como a un nodo tipo “padre”.

El tratamiento en la escritura del fichero, a la hora de guardar el metamodelo, también se hace de modo recursivo puesto que un tipo puede estar formado, a su vez, de otros nodos tipo ‘hijos’, es decir, composición de objetos en jerarquías todo-parte.

8.4.5.1. Ejemplo fichero xml

A continuación, se muestra un xml compuesto por un metamodelo de ejemplo.

Se puede observar como un tipo (marcado en rojo) con posibles relaciones (marcadas en morado) y posibles elementos externos (marcados en naranja) perteneciente a una clasificación (marcada en verde) contiene un subtipo (marcado en azul) que es, a su vez, un tipo.

```
<metamodelo>
  <clasificacion>
    <identificador>ESA</identificador>
    <nombre>ESA</nombre>
    <autor>Agencia Espacial Europea</autor>
    <descripcion>definen las prácticas de software
que deben aplicarse en todos los proyectos de la agencia</descripcion>
    <fechaCreacion>23/01/2010</fechaCreacion>
    <tipos>
      <tipo>
        <identificador>RU</identificador>
        <nombre>Requisitos de usuario</nombre>
        <descripcion>Componen el proceso de organizar la información sobre las
        necesidades del usuario. Ofrecen al usuario una vista del problema pero
        no al desarrollador.</descripcion>
        <relaciones />
        <elementosExternos />
        <requisitosIndividuales />
        <padre>ESA</padre>
        <nivel>1</nivel>
        <tipos>
                <tipo>
                <identificador>RU03</identificador>
                <nombre>Interacción persona-ordenador</nombre>
                <descripcion>Especifica cualquier aspecto relacionado con la
interfaz de usuario. Por ejemplo, estilo, formato, tipo de mensajes, etc.
</descripcion>
                        </relaciones>
                        <elementosExternos>
                         <elementoExterno>
                                <archivo>ListadoVehiculos.doc</archivo>
                                <descripcion>Vista de listar vehiculos.
                                </descripcion>
                        </elementoExterno>
                        </elementosExternos>
                        <requisitosIndividuales />
                        <padre>RU</padre>
                        <nivel>2</nivel>
        <tipos />
    </tipo>
    <tipo>
        <identificador>RS</identificador>
        <nombre>Requisitos software</nombre>
        <descripcion>Permite que el desarrollador construya un modelo de
        implementación que es necesario donde se muestra que es lo que el
        sistema debe hacer. </descripcion>
        <relaciones />
        <elementosExternos />
        <requisitosIndividuales />
        <padre>ESA</padre>
```

```
        <nivel>1</nivel>
        <tipos>
          <tipo>
            <identificador>RS01</identificador>
            <nombre>Funcionales</nombre>
              <descripcion>Especifica las funciones que el sistema debe ser capaz
            de ejecutar. Éste debe definir qué procesos deben hacerse y no  cómo
            implementarlos. </descripcion>
                    <relaciones>
                     <relacion>
                       <nodo>RS02</nodo>
                           <descripcion>Relación entre funcionales y de
                                  ejecución.
                            </descripcion>
                     </relacion>
                    </relaciones>
            <elementosExternos />
            <requisitosIndividuales />
            <padre>RS</padre>
            <nivel>2</nivel>
          </tipo>
        <tipos />
      </tipo>
    </tipos>
  </tipo>
 </tipos>
</clasificacion>
</metamodelo>
```

9. CONCLUSIONES

Aplicando con éxito las habilidades y conocimientos adquiridos durante este proyecto, el objetivo principal del mismo, la definición e implementación de un metamodelo de clasificación de tipos de requisitos, ha sido llevado a cabo con éxito para obtener una mejora en la gestión de los requisitos del software.

El metamodelo, con posibilidad de ser integrado en una herramienta case, soporta todas las clasificaciones determinadas por los estándares Common Criteria, ESA, IEEE, Métrica y RM-ODP. También tiene la posibilidad de instanciar a otras estructuras organizativas que pudieran ser creadas por las personas o instituciones que usen la aplicación.

Y más en concreto, el reuso de los esquemas de clasificación propuestos en los estándares así como otros generados y contenidos en el metamodelo, conlleva los correspondientes beneficios que se obtienen usando técnicas de reuso en el ámbito de la ingeniería de requisitos.

- Mejora en la aplicación de los estándares estudiados.

- Uso de una estructura de requisitos adecuada dependiendo del contexto.

- Mejora en la organización de los requisitos, utilizando menos tiempo y recursos. Esto favorece la gestión y el mantenimiento de los requisitos.

- Reuso de los activos software, para su posterior integración en proyectos específicos.

- Mejora en la calidad del producto software.

En definitiva, el metamodelo implementado proporciona a los ingenieros una notable mejora en la clasificación de los tipos de requisitos con propósito de gestionarlos de una manera más eficiente.

10. REFERENCIAS

- [Alexander, C. 77]: Alexander, C. A pattern language. Oxford University Press. ISBN 0195019199. (1977).

- [Ayuda Microsoft Visual Studio 08]: Ayuda on line Microsoft Visual Studio. Disponible: http://msdn.microsoft.com/en-us/library.

- [Blair, G. 98]: Blair, G. Stefani, J-B. Open Distributed Processing and Multimedia. Addison Wesley. ISBN 9780201177947 (1998).

- [Christel, M. 92]: Christel, M. G. Kang, K. C. Issues in Requirements Elicitation. Disponible: http://www.sei.cmu.edu .

- [Christerson M. 92]: Christerson, M. Jonsson, P. Overgaard, G. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley. ISBN 0-201-54435-0 (1992)

- [Common Criteria 08]: The Common Criteria Recognition Agreement. Common Criteria for Information Technology Security Evaluation. Disponible: http://www.commoncriteriaportal.org

- [C# 01]: Lenguaje C#. Disponible: http://en.csharp-online.net/CSharp_Language_Specification

- [ESA 08]: European Space Agency. ESA Software Engineering & Standardisation. Disponible: http://www.esa.int/TEC/Software_engineering_and_standardisation/

- [Ezran, M. 00]: Ezran, M. Tully, C. Practical Software Reuse: the essential guide. Tata Mcgraw Hill. ISBN 0074635921 (2000).

- [Gamma, E. 95]: Gamma, E. Helm, R. Johnson, R. Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN 0201633612 (1995).

- [IEEE 98 830]: IEEE Standars Association. IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998. Disponible: http://www.ieee.org/publications_standards/index.html

- [Ishikawa, C. 97]: Ishikawa, A. Silverstein, S. Jacobson, M. Fiksdahl-King, M. Pattern Language: Towns, Buildings, Construction. Oxford University Press. ISBN 0195019199 (1997).

- [Karlsson, E. 95]: Karlsson, E. Software Reuse: A Holostic Approach. John Wiley & Sons Ltd. ISBN 0471958190 (1995).

- [Krueger, W. 92] C. W. Krueger, Software Reuse ACM Computing Surveys. ISSN 0360-0300 (1992).

- [Lowe y Hall 99]: Lowe, D. Hall, w. Hypermedia and the Web. An Engineering approach. John Wiley & Son. ISBN 0471983128 (1999)

- [Lowe, D. 02]: Lowe D. Eklund J. Client Needs and the Design Process in Web Projects, Journal of web engineering. Rinton Press. (2002).

- [Metamodelo 06]: Hurtado, O. Fraga, A. Hernanz, I. Llorens, J. Metamodel of requirements' type's classifications for improving the software development process. ISBN Hardcopy: 0889866422 / CD: 088986599X. (2006).

- [Métrica versión 3 08]: Ministerio de Administraciones Públicas de España. Métrica Versión 3. Disponible: http://www.csi.map.es/csi/metrica3/

- [Microsoft Visual Studio 2008 Professional 08]: Microsoft Visual Studio Professional 2008. Disponible: http://www.microsoft.com/visualstudio/es-es/visual-studio-2008-launch?WT.mc_id=SEARCH

- [.NET Framework 10]: .NET. Disponible: http://msdn.microsoft.com/es-es/netframework/default.aspx

- [Open Distributed Processing 98]: Open Distributed Processing - Reference Model. OMG, 1995-98. Disponible: http://isotc.iso.ch/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm

- [Pressman, R. 05]: Pressman, R., Un Enfoque Práctico. McGraw-Hill. ISBN 9701054733 (2005).

- [Rational Rose 07]: Rational Rose. Disponible: http://www-01.ibm.com/software/awdtools/developer/rose/

- [Sommerville, I. 97]: Sommerville, I., Requirements Engineering: A Good Practice Guide. John Wiley & Sons. ISBN 0471974447 (1997).

- [Sommerville, I. 05]: Sommerville, I., Ingeniería de Software. Addison Wesley. ISBN 84-7829074-5 (2005).

- [SwREUSER 08]: The Reuse Company, swREUSER. Disponible: http://www.reusecompany.com