Universidad de Valladolid

E. T. S. DE INGENIERÍA INFORMÁTICA

Ingeniería Técnica en Informática de Sistemas

Líneas de productos y modelos de características
Un caso de estudio en el asociacionismo

Alumno: Guillermo Rodríguez Cano

Universidad de Valladolid

E. T. S. DE INGENIERÍA INFORMÁTICA

Ingeniería Técnica en Informática de Sistemas

Líneas de productos y modelos de características
Un caso de estudio en el asociacionismo

Alumno: Guillermo Rodríguez Cano

Tutora: Carmen Hernández Díez

Resumen

"Quien controla el pasado, controla el futuro; quien controla el presente, controla el pasado"

Eric Arthur Blair (George Orwell)

A lo largo de la historia, el desarrollo de *software* se ha encontrado con diferentes retos que de una manera u otra ha sabido afrontar con éxito.

Cuando se está construyendo un conjunto de sistemas dentro del mismo dominio de aplicación se percibe que en ellos hay una funcionalidad común y otra funcionalidad variable que caracteriza a cada sistema individualmente. El reto es conseguir realizar la elicitación, representación y gestión de la parte común y de la parte variable de este conjunto de sistemas, las familias de sistemas, también llamadas líneas de productos.

Una de las soluciones propuestas para la elicitación, representación y manejabilidad de esta variabilidad son los modelos de características, una combinación en un único modelo de varios modelos: estructurales, de comportamiento,...

Sin embargo, los modelos de características no son suficientes para el desarrollo y ha de buscarse otras técnicas complementarias. UML es un lenguaje que permite modelar sistemas *software* con relativa facilidad, siendo uno de los lenguajes más utilizados en la actualidad. Por ello, resulta interesante continuar el proceso de desarrollo de una familia de sistemas iniciado con el modelo de características, mediante el modelado en UML.

El trabajo aquí propuesto plantea el dominio del asociacionismo como contexto principal para el desarrollo de una familia de sistemas. Así, en una primera parte se probará que el modelado de características en un contexto con escasa documentación y poca relación con otros contextos que pudieran ser adyacentes al dominio de estudio es posible.

En una segunda parte, se demostrará que se pueden utilizar y aplicar con éxito alguno de los mecanismos propuestos para el modelado de características en UML, concretamente la combinación de paquetes de UML2, propuesta en el seno del Grupo GIRO.

El resultado será un estudio detallado de una familia de sistemas utilizando modelos de características sobre el que se aplicará la combinación de paquetes de UML2.

Palabras clave desarrollo dirigido por modelos, líneas de productos, modelo de características, combinación de paquetes, FODA, UML2

Abstract

"Who controls the past, controls the future; who controls the present, controls the past"

Eric Arthur Blair (George Orwell)

Throughout history, software development process has encountered many challenges, one way or another they have been successfully faced.

When a collection of systems within the same application domain are being built it's noticed that there is a common functionality as well as a variable functionality, they both uniquely characterize each system. The challenge is to be able to elicit, represent and manage the commonality and variability among this collection of systems, system families, also known as product lines.

One of the proposed solutions for elicitation, representation and management of variability among systems is feature modeling, an amalgamation in one model of various models: structural, behavioural,...

However, feature modeling is not an "all-in-one" development tool; it is necessary to look for complementary techniques. UML, a general-purpose modeling language, is one of the most widely used modeling languages among software developers, which can be easily used for software system modeling. Therefore, it seems relevant to system family development process, started with feature modeling, that it is followed with UML modeling.

The presented work focuses on the association phenomenon domain as the main scope for systems family development. The first part explores if feature modeling can be easily applied to a poor documented scope and with almost no connection to any other adjacent scope.

The second part will prove that one of the proposed mechanisms for feature modeling in UML may be used and applied successfully, UML2's package merging, proposed within Grupo GIRO.

The outcome will be a detailed study of a system family using feature modeling and UML2's package merging.

Keywords model-driven development, product lines, feature model, package merge, FODA, UML2

Este proyecto está dedicado a la ciencia y al conocimiento, jamás podré devolver todo lo que he aprendido de la sabiduría colectiva, aquí va un minúsculo grano de arena

Agradecimientos

"Sólo un exceso es recomendable en el mundo: el exceso de gratitud"

Jean de la Bruyère

A tí, mi tutora, Carmen, por haber aceptado la propuesta del proyecto. Gracias por tus explicaciones, aclaraciones y enseñanzas semana tras semana, y también por tu paciencia y tiempo empleado conmigo
Gracias, Miguel Ángel, por haber escuchado y aclarado las numerosas dudas que han surgido, y por haber colaborado con tus propuestas

A mi mamá y a mi papá, quienes han hecho lo posible para que haya podido llegar a escribir estas líneas; cuantas veces se lo agradezca siempre serán insuficientes

A mi hermano, Mario, porque durante todos estos años ha sido y siempre será un ejemplo para mí

No menos importantes, aunque algo más genérico el agradecimiento pero igual de sincero, a mis compañeros de la carrera, con quienes he tenido la oportunidad y placer de hacer el mismo camino

Gracias a mis amigos, especialmente a Borja, David y Guillermo, algunos en la distancia y otros en la cercanía, pero todos ellos han estado conmigo en esta pequeña parte de este proceso llamado vida

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1
Introducción

"Un comienzo no desaparece nunca, ni siquiera con un final"

Harry Mulisch

El paradigma de la programación orientada a objetos, comúnmente conocido como OOP (*Object Oriented Programming*), ha supuesto toda una revolución en los procesos de desarrollo de *software*.

Un paradigma que tuvo su nacimiento con los lenguajes *Simula* desarrollados en el Centro de Computación Noruego en la ciudad de Oslo, en los años 60, que comenzó a popularizarse a mediados de los años 80, y que experimentó su mayor influencia a comienzos de los años 90. Actualmente, sigue siendo uno de los más utilizados en decenas de lenguajes de programación, y uno de los campos de investigación con mayores contribuciones al desarrollo de *software* en general.

Tres son los grandes beneficios [34], con la misma importancia, del desarrollo orientado a objetos:

- **Comprensión del mundo real**

  La aceptación y extensión del desarrollo orientado a objetos se debe a su cercanía a la percepción y visión del mundo real por parte del ser humano. Dos citas que resumen con gran precisión y exactitud la importancia de la comprensión y descripción de los conceptos del dominio de aplicación son las siguientes:

  > *"La base filosófica que hay detrás de la programación orientada a objetos es el hecho de poder hacer que los programas reflejen lo máximo posible aquella parte de la realidad con la que vayan a trabajar. De este modo, es más fácil comprender y tener una idea general de lo que se describe en los programas. La razón es que desde el comienzo de la existencia de un ser humano, se le enseña sobre la percepción de lo que pasa en el mundo real. Cuánto más cercana posible sea el uso de esta visión en la forma de programar, más fácil será la escritura y comprensión de los programas"*
  >
  > *Krogdahl and Olsen, 1986*

  > *"El análisis orientado a objetos está basado en conceptos que aprendemos por primera vez en la guardería: objetos y atributos, clases y miembros, todo y parte"*
  >
  > *Coad and Yourdon, 1990*

- **Estabilidad del diseño**

  La noción de un modelo físico del mundo real es esencial para la orientación al objeto, y el propio paradigma proporciona un marco para modelar el dominio de aplicación.

- **Reutilización tanto del diseño como de la implementación**

  Uno de los problemas comunes, y probablemente uno de los más importantes, en el desarrollo de *software* es la posibilidad de reutilización de componentes existentes en nuevos componentes, pues la funcionalidad de nuevos componentes suele ser similar a la de otros ya desarrollados pero puede haber diferencias que hagan imposible una reutilización directa, lo que lleva a implementar esos nuevos componentes mediante la copia y modificación del componente; sin embargo, esto supone tener que probarlo de nuevo.

  El desarrollo orientado a objetos proporciona estructuras en los lenguajes que permiten modificaciones incrementales de modo que no sólo se reutilizan componentes sino que también se amplía y extiende su funcionalidad.

Sin embargo, a pesar de los avances en la reutilización, fiabilidad y modelado que lleva consigo el desarrollo orientado a objetos, se ha llegado a los límites pues las clases, como unidades de reutilización, se han quedado pequeñas; los componentes, como unidades independientes con sus propias interfaces, ofrecen un grado superior a las clases en la reutilización, en tanto que ofrecen una mayor funcionalidad, pero a costa de aumentar la complejidad y por lo tanto disminuir el grado de reutilización; y los patrones, implícitamente reutilizables, no son ningún medio de implementación.

Estudios e investigaciones recientes, así como casos prácticos, sugieren que la reutilización de componentes en el desarrollo de *software* debe experimentar un cambio de rumbo hacia la producción de familias de sistemas, en lugar de sistemas individuales, si se quiere evolucionar.

Las familias de sistemas (también conocidas como líneas de productos) se centran en la búsqueda de lo común entre diferentes sistemas, dentro de un mismo dominio, para también controlar los puntos de variabilidad entre esos sistemas de una forma más o menos automática. De esta forma, es posible obtener nuevos sistemas basados en la reutilización[1] de componentes ya existentes.

La ingeniería de familias de sistemas distingue entre dos procesos en el desarrollo: ingeniería del dominio e ingeniería de la aplicación.

La ingeniería del dominio, también llamada desarrollo de líneas de productos, es lo que se denomina como “desarrollar para reutilizar”, ya que su función principal es la de producir elementos que serán reutilizados posteriormente (desde la implementación de componentes hasta la documentación), y al igual que en el desarrollo de sistemas particulares o individuales, se atraviesan las diferentes fases de análisis, diseño e implementación, aunque teniendo en cuenta que se está desarrollando para un conjunto de sistemas y no uno sólo.

El análisis del dominio determinará el ámbito o contexto de la línea de productos, de modo que identificará los elementos comunes dentro del dominio, así como la variabilidad existente. El diseño del dominio se encargará de producir una arquitectura común para toda la línea de productos y también planificará la forma en que se producirán cada uno de los miembros individuales de la familia,

[1] Krzysztof Czarnecki considera que la ingeniería de familias de sistemas se centra en la construcción de nuevos sistemas reutilizando componentes, mientras que la ingeniería de líneas de productos se centra en controlar las características comunes en los productos dentro de un determinado contexto con una perspectiva de mercado

haciendo uso de los componentes reutilizables. La implementación del dominio se encarga de la codificación de los componentes que posteriormente serán reutilizados.

La ingeniería de la aplicación, también denominada con el nombre de desarrollo de un sistema o producto, reúne el conjunto de tareas necesarias para construir aplicaciones concretas reutilizando componentes.

El desarrollo se realiza de la forma tradicional: elicitación de requisitos, análisis,... pero con la particularidad de que los requisitos son especificados a partir de una selección[2] de los requisitos especificados durante la ingeniería del dominio.

La especificación de requisitos es la principal fuente de información para la obtención de diferentes sistemas dentro del dominio. Por este motivo, es muy importante que la ingeniería del dominio proporcione a la ingeniería de la aplicación componentes reutilizables, para que ésta proporcione a su vez nuevos requisitos a tener en cuenta.

El motivo de que se produzca esta retro-alimentación entre ambos procesos se debe a que la construcción de un sistema descubre nuevos requisitos que se consideran relevantes, por lo que se reintroducen en la familia a través de la ingeniería del dominio, para mantener los componentes reutilizables de la familia actualizados, bien porque sean necesarios para el sistema que aportó los requisitos, bien porque sean útiles para futuros sistemas.

La ilustración de la figura 1.1 muestra un diagrama simplificado de la retro-alimentación entre los dos procesos principales del desarrollo de una familia de sistemas.

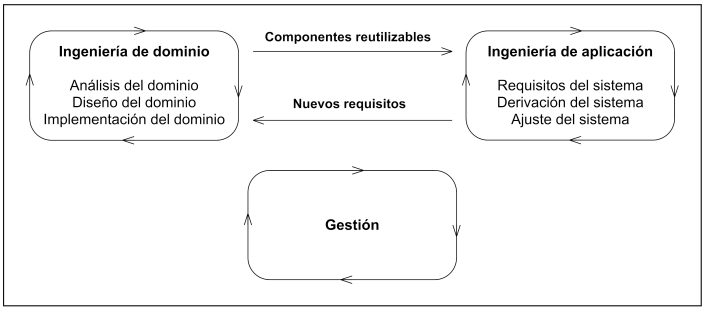


Figura 1.1: Esquema de los principales procesos en el desarrollo de una familia de sistemas

La elicitación de requisitos en una línea de productos se puede hacer de muchas formas; sin embargo, para una familia de sistemas existe un método que permite representar las partes comunes de la familia y también la variabilidad, denominado modelo de características.

Los modelos de características fueron propuestos por Kang y otros en 1990, y desde entonces se han añadido diversos conceptos a la propuesta original para cubrir las necesidades de los desarrollos que los utilizan: características de grupo, cardinalidades, atributos,...

Los modelos pueden ser complementados con restricciones (la selección de una característica puede requerir la selección de otra característica en otro punto del modelo o incluso excluir su selección),

[2] Este proceso se denominará, posteriormente, configuración de una línea de productos

tiempos de configuración (la selección de una característica se puede hacer en un momento del desarrollo pero no en otro, por ejemplo en tiempo de compilación y en tiempo de ejecución), características por defecto, prioridades,...

Czarnecki plantea [7] cuatro tipos diferentes de características:

- **Concretas**: Por ejemplo, el almacenamiento o clasificación se puede llevar a cabo como componentes individuales.
- **Aspectuales**: Por ejemplo, la sincronización y persistencia puede afectar a diversos componentes y se puede modularizar mediante tecnologías de aspectos.
- **Abstractas**: Por ejemplo, los requisitos asociados al rendimiento suelen estar asociados a una configuración de algún componente.
- **De grupo**: Este tipo de característica puede tener una tarea organizativa en la que no hay requirimientos involucrados.

De este modo se llega a denominar al desarrollo de familias de sistemas como MDSPL (*Model-Driven Software Product Line*), ya que el proceso de elicitación de requisitos se realiza mediante el modelo de características, que a su vez es el punto de entrada para la obtención de un sistema concreto de la familia porque proporciona un mapa de toda la funcionalidad de la familia, así como información complementaria necesaria para comprender las consecuencias de la selección de una funcionalidad (representada en características).

En las fases iniciales del desarrollo de una familia de sistemas, los modelos de características proporcionan una base necesaria para comprender el ámbito que abarcará la familia. De este modo, esos modelos son el punto de comienzo de la arquitectura que se usará en la familia y también de lo que se denomina como DSL (*Domain Specific Language*). No hay que olvidar que esta arquitectura deberá permitir representar la parte común de la familia y especialmente la variable, que es la que llevará a que un sistema pueda ser diferente de otro.

Un DSL es un lenguaje que permite expresar de una forma particular un dominio. La reducción del ámbito de un lenguage a un dominio concreto permite proprocionar un mejor soporte para la resolución de los problemas dentro del dominio comparado con lo que un lenguage de propósito general podría ofrecer.

Las ventajas de un DSL frente a un lenguaje de propósito general son:

- **Abstracciones específicas**: un DSL proporciona abstracciones predefinidas que representan conceptos del dominio de aplicación.
- **Sintaxis concreta**: un DSL ofrece una notación natural para un dominio y evita complicaciones sintácticas que muchas veces surgen en los lenguajes de propósito general.
- **Comprobación de errores**: un DSL permite que se puedan construir analizadores de errores que encuentren más errores, y además muestren esos errores en un lenguage más familiar frente a los lenguajes de propósito general.
- **Optimización**: un DSL permite generar código optimizado en base al conocimiento existente del dominio, algo que pocas veces es posible hacer con un lenguaje de propósito general.

- **Soporte de herramientas**: el conocimiento específico del dominio que un DSL permite representar se puede usar para proporcionar herramientas a los desarrolladores más específicas al dominio.

Los DSL se pueden encontrar de muchas formas, desde lenguages de texto (independientes o integrados en un lenguaje de propósito general) hasta interfaces gráficas interactivas y asistentes, e incluso una combinación de ambos. Un ejemplo de un DSL con una interfaz gráfica es *fmp* (*Feature Model Plug-In*).

La estructura de un DSL así como la tecnología utilizada para implementarlo dependerá en gran medida de lo que se necesite hacer, por ejemplo, *fmp* estaría entre los asistentes y los lenguajes como tales.

Lo importante es que el DSL sea útil para los desarrolladores dentro del dominio de aplicación, y el modelado de características junto con el *plug-in fmp* son un buen soporte para el desarrollo de familias de sistemas.

Por último, aunque no menos importante, no hay que olvidar que la elicitación de requisitos de una familia de sistemas es una pequeña parte de todo el proceso de desarrollo de la familia, si bien es cierto que es una de las fases más importantes, ya que es el momento en el que se define el contexto del dominio.

Hoy en día, el estándar de modelado en la industria del desarrollo de *software* para fases posteriores del desarrollo de cualquier producto es UML (*Unified Modeling Language*). Un lenguaje gráfico de modelado de sistemas, que incluye aspectos tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de *software* reutilizables. Si bien es importante saber que UML es un lenguaje de especificación y no de descripción de métodos o procesos de desarrollo *software*.

UML ha evolucionado a lo largo de la historia, proporcionando en las diferentes revisiones solución a los problemas surgidos en diversos dominios. Sin embargo, los modelos de características todavía no son soportados por UML, por lo que no existe una especificación concreta para representar todo ese trabajo de la fase de análisis en UML y así facilitar la fase de diseño e implementación. No obstante, UML permite ser extendido, y proporciona para ello mecanismos como perfiles y estereotipos.

Sin embargo, por mucha extensión que se haga, seguirá siendo particular al dominio de aplicación y no estándar. Por ello es necesario encontrar elementos existentes en el propio lenguaje, estableciendo los mecanismos que sean necesarios, para representar los modelos de características en UML, evitando la inclusión de gramática nueva y también, si es posible, semántica.

El trabajo que a continuación se presenta es un estudio pormenorizado de la validez de los modelos de características aplicado al ámbito del asociacionismo, dado que es un dominio de aplicación relativamente manejable y no estudiado desde el punto de vista de las familias de sistemas (al menos la búsqueda de documentación existente ha resultado ser fallida).

No obstante, este trabajo no sólo se quedará en los modelos de características sino que también pretende estudiar las propuestas existentes para el modelado de características en UML, y aplicar alguna de ellas.

1.1. Motivación y objetivos

Este Proyecto Fin de Carrera surgió a partir de la propuesta del autor del mismo a la tutora, quien "recondujo" las aspiraciones de la propuesta inicial hacia una terreno más manejable por un alumno, teniendo presente que el tiempo de desarrollo era limitado y también lo eran los recursos humanos.

No obstante, la motivación de este proyecto tuvo su origen en la pertenencia del autor a una asociación universitaria y la necesidad de disponer de una herramienta que facilitase la organización del quehacer diario de los miembros de la asociación, así como la comunicación interna, con el objeto de mejorar internamente para mejorar externamente.

Un requisito que se quería mantener fue que la herramienta resultante no tuviese aplicación exclusiva en la asociación que motivó la idea, sino que fuese una herramienta más genérica, de modo que pudiese ser utilizada por otras asociaciones (logicamente modificando la herramienta de acuerdo a las necesidades particulares de cada asociación).

Visto lo que se quería hacer y el contexto en el que se quería aplicar la propuesta, se llegó a la conclusión de que lo que se quería desarrollar era una línea de productos *software*, y en base a esta premisa surgen los siguientes objetivos:

- **Estudio del proceso de desarrollo basado en líneas de productos**

  Dado que en los estudios de Ingeniería Técnica en Informática de Sistemas, no se imparten conocimientos sobre líneas de producto, una parte importante del trabajo inicial será el estudio, comprensión y documentación del origen y bases del desarrollo basado en líneas de productos.

- **Estudio del modelo de características y su aplicación en el desarrollo de líneas de productos**

  Derivado del objetivo anterior, se propone comprender y aprender sobre uno de los modelos de decisión de aplicación en el desarrollo de líneas de productos, los modelos de características.

  Para este objetivo se propone partir de la documentación del estudio FODA (*Feature-Oriented Domain Analysis*) [24].

- **Estudio de una línea de productos para asociaciones**

  Una vez adquiridos y comprendidos los conocimientos, el siguiente paso que se propone cumplir es comprobar la viabilidad de los modelos de características en una línea de productos para asociaciones, para la que no existe documentación (en el momento de la propuesta) escrita, teniendo en cuenta que los modelos de características son una parte más del proceso de desarrollo.

  Se pretende obtener un modelo de características completo y detallado. Se usará una herramienta para representar los modelos de características: *Feature Model Plug-In* (un *plug-in* del entorno de desarrollo *Eclipse*).

- **Construcción de la línea de productos con UML**

  Con el modelo de características de la línea de productos se propone comenzar la construcción de la línea para comprobar la validez del modelo.

  Para la construcción habrá de aplicarse la propuesta [31] de los investigadores Laguna y otros

del Grupo GIRO (*Grupo de Investigación en Reutilización y Orientación a Objeto*) del Departamento de Informática de la Universidad de Valladolid.

Esta propuesta plantea una serie de pautas de transformación del modelo de características a diferentes tipos de diagramas en el lenguaje de modelado por excelencia en ingeniería del *software*: UML.

No obstante, aunque se aplique esta propuesta, se tendrán que estudiar otras propuestas existentes.

1.2. Resumen de capítulos

A continuación se resume brevemente cada uno de los capítulos que forman esta memoria para tener una idea general de cómo se ha organizado este trabajo de acuerdo a los objetivos planteados en el apartado anterior.

- **Capítulo 2: Base teórica y metodología**

  En este capítulo se introducen las bases teóricas del desarrollo de líneas de productos dirigido por modelos, desde las motivaciones hasta los beneficios a los que da lugar esta metodología.

  También se introduce el análisis de una línea de productos basado en un modelo de decisión, concretamente, los modelos de características, junto con la transposición de dichos modelos en el meta-modelo del lenguaje de modelado de *software* más extendido hoy en día, UML, de acuerdo a las pautas establecidas en la propuesta [31] de los investigadores Laguna y otros.

  Por último, el capítulo finaliza explicando la metodología aplicada para la elaboración del análisis de una línea de productos para asociaciones.

- **Capítulo 3: Caso de estudio de una línea de productos para asociaciones**

  En este capítulo se realiza un análisis pormenorizado y con gran detalle del modelo de características de una línea de productos para asociaciones, explicando cada una de las características, así como las relaciones existentes con otras características y también las restricciones que habrán de tenerse en cuenta posteriormente.

  El capítulo finaliza con un breve comentario sobre la validez del modelo analizado y su modificación o ampliación a lo largo del ciclo de vida de la línea de productos. Por último, se explican las motivaciones que han llevado a reducir el amplio análisis hecho del modelo de características a uno más modesto pero también asequible.

- **Capítulo 4: Construcción de la línea de productos para asociaciones**

  Este capítulo se dedica a plantear los comienzos de la construcción de la línea de productos basada en la simplificación del modelo de características detallado en el capítulo anterior.

  En primer lugar se describe brevemente el modelo de características simplificado, haciendo referencia al apartado anterior, dado que es una reducción del modelo de dicho apartado.

  Los restantes apartados del capítulo se dedican a plantear los diagramas de casos de uso y de clases del dominio siguiente las pautas de la propuesta [31] de Laguna y otros. Los diagramas de casos de uso son descritos y relacionados con el modelo de características, pero los diagramas de clases simplemente son mostrados, ya que la descripción que les correspondería sería la misma que la de los diagramas de casos de uso.

- **Capítulo 5: Conclusiones y líneas futuras**

  En este último capítulo se recopilan las conclusiones a las que se ha llegado una vez se ha terminado el trabajo. Conclusiones, que más allá de ser positivas o negativas, pretenden plantear posibles líneas futuras de trabajo así como ampliaciones del trabajo realizado.

- **Apéndice A: Meta-modelo de *Feature Modeling***

  Este apéndice muestra una propuesta [41] del investigador Vranić para representar el meta-modelo para modelos de características utilizando la notación más extendida hoy en día (Notación *Czarnecki-Eisenecker*) de los modelos de características.

- **Apéndice B: Diagrama de clases del meta-modelo utilizado en *Feature Model Plug-In***

  Este otro apéndice muestra el diagrama de clases del meta-modelo de los modelos de características que se pueden generar con la herramienta *Feature Model Plug-In* realizada [1] por el equipo del investigador Krzysztof Czarnecki y que se utiliza a lo largo de todo el trabajo.

- **Apéndice C: Instalación y uso en *Eclipse* de *Feature Model Plug-In***

  Este penúltimo apéndice proporciona información gráfica sobre la instalación del *plug-in* de *Eclipse*: *fmp*, en su versión 0.7.0 (la más moderna durante la elaboración de este trabajo).

  También se muestra un breve ejemplo de uso del *plug-in* con el modelo de características simplificado que se ha utilizado para obtener los diagramas del capítulo 4.

- **Apéndice D: Contenido del disco**

  En el último apéndice se muestra una relación de los diferentes ficheros así como su organización en el disco adjunto al presente trabajo.

Capítulo 2
Base teórica y metodología

"No hay que empezar siempre por la noción primera de las cosas que se estudian, sino por aquello que puede facilitar el aprendizaje"

Aristóteles

2.1. Desarrollo de *software* basado en líneas de productos

En la actualidad, los procesos, las técnicas y las herramientas de desarrollo de *software* están orientadas, mayoritariamente, a la obtención de productos "individualizados". Sin embargo, esto plantea ciertos problemas a la hora de obtener productos, que tienen partes en común, pero cuya variabilidad entre sí no es lo suficientemente alta como para considerarlos productos distintos, o incluso independientes, y que son desarrollados en más tiempo y con más recursos, tanto humanos como materiales, de los que sería necesario si se aprovechase esta característica entre otras.

Los fabricantes de *software* han utilizado diferentes técnicas para "emular" el desarrollo de una línea de productos, utilizando una especie de cadena de ensamblaje común que uniese los diferentes componentes reutilizables, de acuerdo a las diferentes configuraciones para cada producto de la línea de productos.

La industria automovilística es un ejemplo, bastante gráfico, de esta forma de proceder mediante una única cadena de montaje, en la que se ensamblan las diferentes partes que conforman un vehículo, es posible producir vehículos diferentes (el color, el número de puertas,...), sin necesidad de tener que disponer una cadena de montaje para cada configuración de un vehículo.

El desarrollo de *software* basado en líneas de producto, SPLD (*Software Product-Line Development*)[1], se diferencia de intentos anteriores, en la predictibilidad de la reutilización frente al oportunismo.

En lugar de desarrollar diferentes componentes *software* para ser integrados en una librería, con la esperanza de que en algún momento sean utilizados, las líneas de productos buscan el desarrollo de componentes que realmente vayan a ser utilizados, es decir, que haya al menos un producto de dicha línea que haga uso de ese componente (logicamente, podrá ser utilizado en más productos que puedan

[1] Es posible que también lo pueda encontrar como SPLE (*Software Product-Line Engineering*) o abreviado, como SPL (*Software-Product Line*), o incluso como SFE (*System-Family Engineering*)

resultar de la línea de productos).

El objetivo está en, por un lado, maximizar y también encontrar lo común evitando las duplicidades y compartiendo los recursos, y, por otro, controlar e identificar las variaciones estableciendo puntos de control. Un ejemplo ilustrativo de esto se muestra en la figura 2.1.

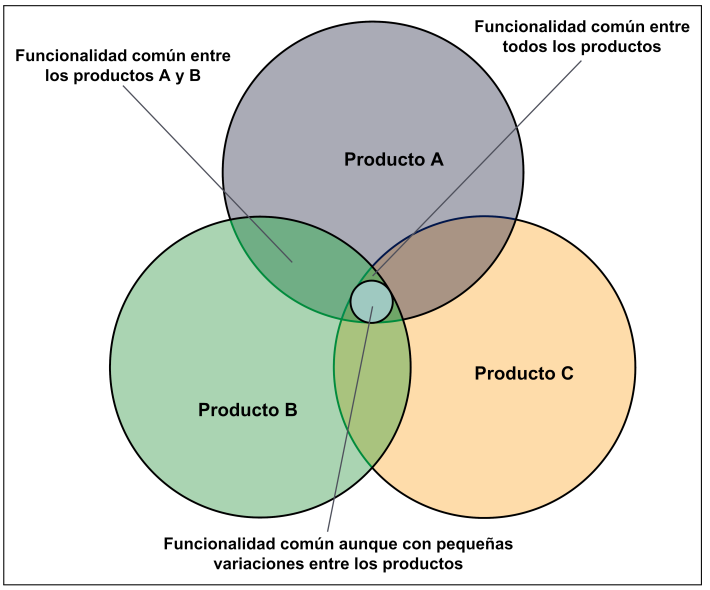


Figura 2.1: Diagrama de variabiliad y elementos comunes entre productos

2.1.1. Definición[2] y elementos principales

Una línea de productos es un conjunto de sistemas de *software* que comparten una serie de características que satisfacen las necesidades específicas de un determinado sector del mercado y que son desarrollados a partir de una serie de elementos base comunes.

En una línea de productos, se pueden distinguir, principalmente, cuatro partes, según se ilustra en la figura 2.2:

- **Entrada**: el conjunto de elementos que son de utilidad para la creación de los diferentes productos que se producirán mediante la línea de productos: componentes *software*, requisitos, diagramas, pruebas,...

  Cada uno de estos elementos tiene un rol que desempeñar dentro de la producción en la línea de productos, algunos serán opcionales y otros obligatorios para todos los productos, también es posible que alguno de ellos sea, a su vez, configurable, para poder variar su comportamiento de acuerdo a las necesidades.

[2]Traducción de la definición dada por Paul C. Clements del SEI (*Software Engineering Institute*)

- **Configuración**: el modelo de decisión, el cual describe los elementos opcionales y obligatorios dentro de la línea de producto.

  Cada producto de la línea de productos queda definido, unívocamente, por una serie de decisiones, que se han tomado en base al modelo de decisión.

- **Producción**: el conjunto de medios y mecanismos necesarios para, de acuerdo al modelo de decisión y las decisiones tomadas en base a dicho modelo, ensamblar los diferentes elementos de entrada.

- **Salida**: la colección de elementos que la línea de productos puede "fabricar".

  El ámbito de la línea de productos queda determinado por la finalidad que se haya establecido en la misma de acuerdo a los elementos de entrada y al modelo de decisión.

  Por lo tanto, una línea de productos no tiene por qué tener como salida un producto *software* que sea una aplicación funcional, es decir, podría, perfectamente, ser una línea de productos orientada a la producción de componentes *software* que, a su vez, sean parte de la entrada de otra línea de productos.

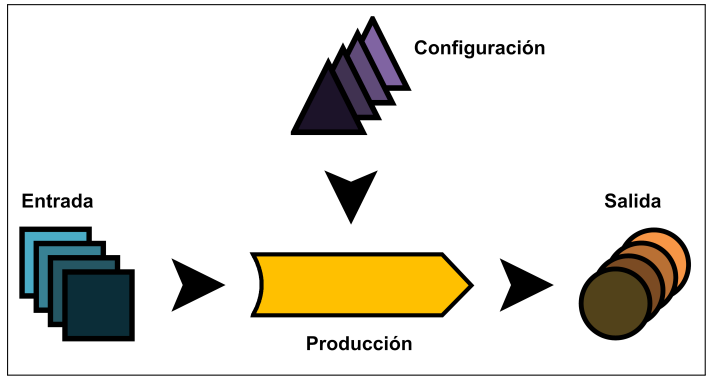


Figura 2.2: Esquema de las partes de una línea de productos

Se ha mencionado que uno de los objetivos de una línea de productos está en controlar las variaciones estableciendo puntos de control. Estos puntos de control van a permitir decidir sobre el funcionamiento del producto *software*. A lo largo del ciclo de vida de la línea de productos se van tomando decisiones, hasta llegar a un modelo de decisión, en el que el producto *software* está completamente especificado.

El momento de la toma de decisiones no tiene por qué ser único; puede estar en cualquier punto de la línea de productos. Dependiendo del momento elegido, la decisión deberá ser tomada por un actor o por otro. Por ejemplo, durante la fase de codificación en la que el programador tomará la decisión, o durante la fase de despliegue o incluso ejecución en la que el desarrollador o el propio cliente, respectivamente, tomará esa decisión, también podría ser durante la fase de compilación o diseño de la arquitectura,...

Al no tomar todas las decisiones en un único instante, es decir, al no elaborar un modelo de decisión completo, sino parcial para una o varias fases[3], pero no para todas las fases del desarrollo, la salida

[3]También se pueden denominar especializaciones, en tanto que son modelos derivados de otros modelos a su vez

obtenida se convierte en la entrada para la siguiente fase en tanto que se trata de una salida "parcial" y requerirá de su correspondiente modelo de decisión parcial.

2.1.2. Fase de producción

La parte de producción de una línea de productos es la encargada de obtener productos *software*, bien parciales o bien completos, a partir del ensamblado de los elementos de la parte de entrada, de acuerdo al modelo de decisión establecido en la parte de configuración. Ésta parte de producción es la que permite diferenciar una línea de productos respecto de otra.

La fase de producción tiene como características principales:

- **Automatización**: la producción de *software*, desde una línea de productos puede estar completamente automatizada, parcialmente o ser manual, e incluso, tener partes automatizadas y otras manuales.
- **Periodicidad**: si bien el desarrollo de *software* basado en líneas de producto supone una mejora en los procesos de desarrollo de *software*, no deja de tener los mismos problemas de todo programa, y además, los productos *software* sufren revisiones para corregir fallos o errores, o para añadir funcionalidades nuevas. La periodicidad de la producción de *software* también está ligada a lo automatizada que esté la línea de productos.
- **Roles**: algunas líneas de producción pueden establecer roles, como por ejemplo, el rol del ingeniero del dominio, encargado del diseño de los elementos de entrada, o el del ingeniero de la aplicación, encargado de la producción. Otras líneas establecen un único rol o incluso ninguno.

  De nuevo, la existencia de roles en una línea de productos está ligada al grado de automatización de la misma. La existencia de un único rol suele darse en las líneas completamente automatizadas. En aquellas en las que hay alguna parte o todo manual, suelen establecerse los roles habituales de todo proyecto *software*.

Como ya se ha comentado, la parte de producción requiere una serie de elementos de entrada. Esos elementos de entrada, pueden tomar muchas formas, no estando restringidos a un tipo concreto o un origen determinado. Estos elementos pueden ser binarios, código fuente que podrá ser modificado en cualquier momento de la producción, requisitos, casos de uso, pruebas, diagramas de clases,...

Si bien la configuración no es parte de la entrada, ya que no es un elemento que vaya a estar físicamente en el producto *software* que se obtenga en la línea de productos, sí que forma parte de la entrada en tanto que dirige todo el proceso; sin esta configuración la línea de productos no podría producir nada, pues no sabría cómo realizar el ensamblaje en cada fase.

El modelo de decisión, que rige la parte de configuración, puede estar descrito en lenguaje natural mediante descripciones hasta constar de sentencias en un lenguaje de programación.

Algunas líneas de productos pueden proporcionar algún tipo de ayuda a la hora de la toma de esas decisiones, en forma de guías, o incluso comprobación de restricciones. Éstas ayudas permitirán reducir el tiempo, la complejidad y los errores.

No hay que olvidar que una línea de productos puede generar productos *software* similares, por lo

que es interesante tener en cuenta los modelos de decisión realizados para otros productos que se hayan obtenido anteriormente, para que sean ejecutados en lugar de tener que volver a generarlos.

La parte de salida de la producción, que podrá ser un producto *software* completamente funcional o bien un producto *software* parcial, y en este último caso, pasará a formar parte de los elementos de entrada de otra línea de productos o bien de otra fase de la misma línea de productos, podrán tener múltiples representaciones, desde binarios hasta código fuente.

2.1.3. Actualización y mantenimiento

Como todo desarrollo *software*, las líneas de productos están sujetas al mantenimiento, y supeditadas también, a la evolución de la tecnologías y a las nuevas necesidades de los usuarios que hacen uso de los productos *software* obtenidos con ellas.

Éstas variaciones son distintas a los puntos de control que se establecen en el modelo de decisión, y que permiten generar todo el espacio de productos *software* de la línea de productos; las últimas, denominadas variaciones en el tiempo, están impuestas de antemano, y las primeras, denominadas variaciones en el espacio, aparecen a lo largo del tiempo, de forma más o menos previsible o no.

Algunas de las variaciones en el tiempo se pueden solventar facilmente con algún pequeño retoque en la configuración de alguno de los elementos de entrada o del propio producto *software*.

Hay otras variaciones, ilustradas en la figura 2.3, que implican mayores cambios de modo que un cambio en la entrada deba propagarse por toda la línea de productos hasta la salida, pasando por el modelo de decisión; pero también puede ocurrir lo contrario si se ha permitido que los productos generados tengan cierta capacidad de decisión, y por lo tanto, la salida alimenta la entrada, habiendo antes pasado por el modelo de decisión, para volver a propagarse de nuevo por toda la línea de productos.

La automatización de la línea de productos simplifica, en gran medida, estos cambios, especialmente los cambios que se producen en la salida.

2.1.4. Ámbito

Si los límites, o el ámbito de un producto *software* están delimitados por las funcionalidades que ese producto provee, del mismo modo el ámbito de una línea de productos está delimitado por la unión de las funcionalidades de cada uno de los productos *software* que esa línea de productos es capaz de generar.

El ámbito de los productos *software* de una línea de productos se puede caracterizar en términos de los siguientes usos del producto *software*: quién, qué, cuándo, dónde, cómo y porqué. La multiplicidad entre éstas es lo que lleva a obtener diferentes productos en una línea de productos.

Cuando se planifica y desarrolla el ámbito de una línea de productos, hay dos enfoques a tener en cuenta. El primero de ellos, un enfoque “activo”, por el que la línea de productos soportará la generación de todo producto que se pueda preveer necesitar en un futuro cercano; el segundo de ellos, un enfoque “pasivo” (o reactivo), por el que la línea de productos soportará la generación de

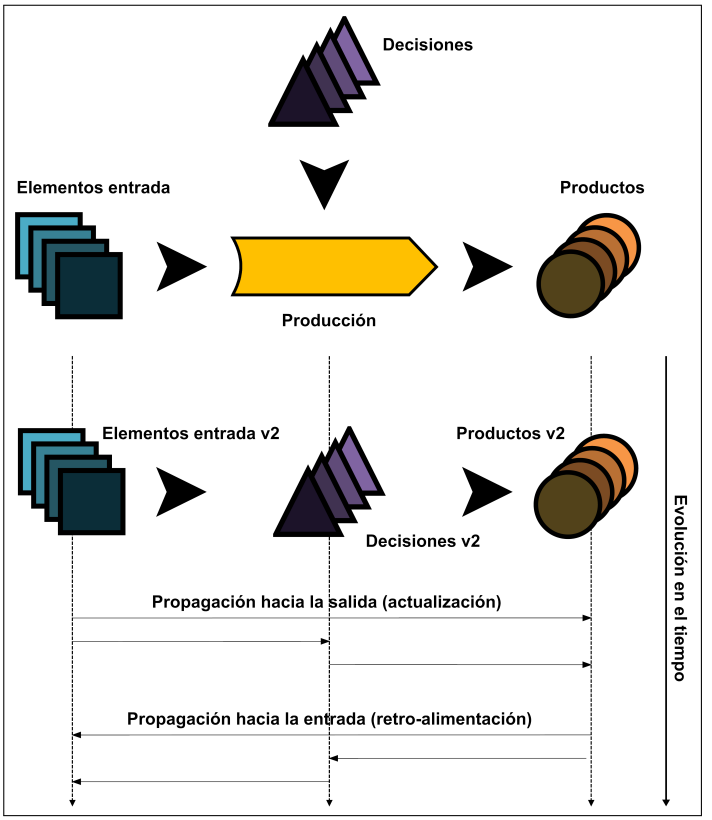


Figura 2.3: Influencia en la línea de productos de las variaciones en el tiempo

los productos que se necesitan a corto plazo, y para aquellos que se pueda necesitar en un futuro, se añadirán a la línea de productos de acuerdo a las necesidades.

Queda claro, que el primer modelo de planificación supone un mayor coste inicial, que se amortizará a lo largo del tiempo según se vayan necesitando esos productos que estaban previstos, y para los que no hay que hacer grandes modificaciones en la línea de productos. Sin embargo, el segundo modelo de planificación no supone un gran coste inicial, pero para cubrir el mismo ámbito que el primer modelo, necesitará la misma inversión, aunque se repartirá a lo largo del tiempo.

Hay ciertos elementos, y especialmente en las primeras fases de la línea de productos, que pueden tener un carácter más “activo”, como por ejemplo la arquitectura que tendrán los productos de la línea, y otros elementos que podrán tener un carácter más “pasivo”, como por ejemplo el código fuente.

En cualquier caso, el ámbito de la línea de productos queda definido por los productos que la línea de productos puede generar para ser probados y desplegados.

2.1.5. Beneficios

Las ventajas, o beneficios que el desarrollo de *software* basado en líneas de productos aporta a la ingeniería del *software*, se puede cuantificar en forma de beneficios "tácticos", es decir, el despliegue de un producto *software* se realiza con mayor rapidez, menor coste y más calidad. Sin embargo, estas ventajas se ven con frecuencia relegadas por la política de desarrollo de *software*.

De acuerdo a diversos estudios realizados, las mejoras que se pueden obtener al usar una línea de productos, oscilan en un factor de entre 3 y 50. Entre estas mejoras destacan:

- Reducción en el tiempo medio de creación y despliegue de un nuevo producto.
- Reducción en el número medio de errores por producto.
- Reducción en el esfuerzo medio que se ha de hacer para el despliegue y mantenimiento del producto, y por lo tanto se logra una reducción en el coste medio de cada producto.
- Incremento en el número total de productos que pueden ser desplegados y mantenidos.

Las líneas de productos se basan, como ya se ha comentado anteriormente, en lo común a lo largo de la línea, así como en los puntos de variabilidad de la misma. En toda línea de productos se minimizan las duplicaciones, ya que ello supone esfuerzo y tiempo que se podría haber empleado para mejorar la línea o incluso ampliarla.

2.1.6. Calidad

La calidad de las líneas de productos se puede medir de dos formas. La primera de ellas, y más evidente, está basada en el grado de coincidencia del producto *software* con las necesidades del usuario; la segunda, se centra en el número de errores que hay en cada producto de la línea.

Un menor número de errores agiliza el ciclo de vida de desarrollo, porque las fases de pruebas serán más cortas, y los equipos que las realicen serán más pequeños. Los desarrolladores podrán emplear su tiempo en desarrollar nuevas funcionalidades para los productos de la línea.

Esta calidad está directamente relacionada con el grado de búsqueda de lo común a la hora de planificar y desarrollar la línea. La optimización en la reutilización de los diferentes elementos a lo largo de la línea y de su ciclo de vida permite aumentar la calidad de los productos generados y facilita la generación de productos futuros.

Un error en cualquier componente de entrada de la línea de productos llevará a una revisión de todos los productos obtenidos que integren dicho componente, pero su corrección permitirá generar futuros productos libres de ese error.

Es importante tener en cuenta que los primeros productos obtenidos (véase la ilustración de la figura 2.4) a través de la línea de productos, que en ocasiones serán productos "test" o "piloto", van a presentar un mayor número de errores que productos que se obtengan posteriormente. Algo similar ocurrirá con nuevas versiones de un mismo producto. El objetivo está en minimizar el número de errores, para destinar el esfuerzo que se ha de emplear en la solución de ellos, en los aspectos indicados anteriormente.

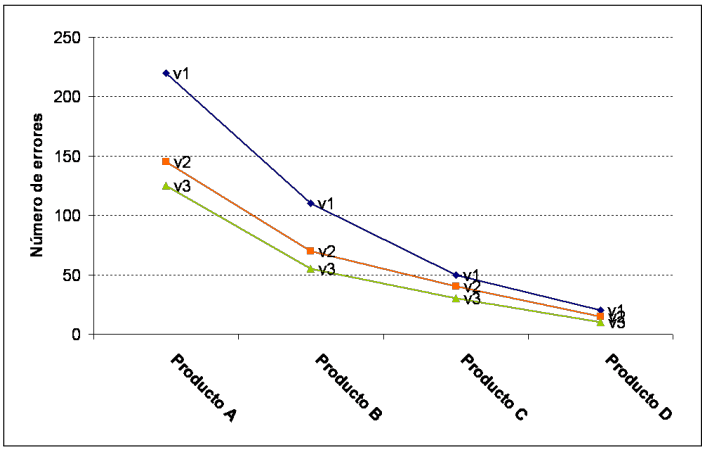


Figura 2.4: Evolución de los errores en la línea de productos

El desarrollo basado en líneas de productos ofrece significativas e importantes mejoras en los procesos de desarrollo de *software*. No sólo es una forma o método de desarrollo de *software*, sino que también es una manera de aportar calidad al *software* que se desarrolla.

2.2. Análisis basado en modelos de características

Anteriormente se ha explicado la importancia del modelo de decisión en el desarrollo de *software* basado en líneas de productos. Esta manera de desarrollar *software* se conoce, de forma abreviada, como MDPSL (*Model-Driven Software Product Line*), que a su vez no es más que la combinación de dos prácticas en constante auge dentro de los diferentes campos que abarca la ingeniería del *software*: SPLE (*Software Product-Line Engineering*) y MDSD (*Model-Driven Software Development*).

El desarrollo de *software* dirigido por modelos, MDSD, es una práctica de desarrollo de *software*, que utiliza modelos en lugar de código fuente como elemento principal para la implementación de un producto.

Los modelos tienden a ser más aptos para la comprensión de los objetivos de los requisitos que se hayan establecido porque están más próximos a éstos que el propio código fuente. Ciertos lenguajes de modelado, como UML (*Unified Modeling Language*), proporcionan en su semántica elementos que permiten al usuario[4] mayor precisión a la hora de la elicitación de requisitos en lugar de tener que preocuparse por aspectos de implementación en un lenguaje concreto que muy probablemente provocarán problemas en dicha fase.

El uso de modelos como primer elemento, o elemento principal, en el proceso de desarrollo, lleva a que éstos sean considerados de la misma forma, en cuanto a importancia, que el código fuente en un desarrollo de *software* "tradicional".

2.2.1. Orígen del modelado basado en características: FODA

Con frecuencia, los diagramas de flujo de datos han sido utilizados por los desarrolladores para poder "entenderse" con el usuario, e incluso con otros desarrolladores. Sin embargo, estos diagramas contienen información (definición de funciones entre otras) que no interesa, y en ocasiones confunde a los usuarios finales. Los usuarios quieren conocer las características, funcionalidades o necesidades que el producto *software* satisfará.

Se busca un modelo que sea lo suficientemente simple como para que el usuario pueda comprenderlo, y de este modo, opinar sobre él; y al mismo tiempo que ese modelo sea mínimamente técnico como para servir de base para comenzar el desarrollo.

Los orígenes del modelado basado en características, se remotan a 1990, cuando el SEI (*Software Engineering Institute*) publicó la metodología de análisis del dominio, FODA (*Feature-Oriented Domain Analysis*), ya que fue el modelo desarrollado para expresar de forma gráfica el dominio de aplicación. Sin embargo no es la única metodología donde se ha utilizado, FORM (*Feature-Oriented Reuse Method*), ODM (*Organization Domain Modeling*), GP (*Generative Programming*) o MPD_{FM} (*Multi-Paradigm Design with Feature Modeling*) hacen uso del modelado basado en características.

FODA define un proceso para el análisis del dominio que consta de tres fases:

[4] El usuario del lenguaje no debe ser entendido como el usuario final de la aplicación o producto *software*, sino el rol del desarrollador encargado del modelado en un determinado lenguaje, en este caso, UML

- **Análisis del contexto**: define la extensión o los límites del dominio a analizar.
  El análisis del contexto permite definir el ámbito del dominio en el que se producirán productos *software*. El análisis del contexto, la experiencia en el dominio tratado, la disponibilidad de información y las restricciones del propio proyecto, se utilizan para determinar el contexto.

  El resultado del análisis es el modelo del contexto, que incluye un diagrama de estructura y otro de flujo de datos. El primero es un diagrama de bloques en el que el dominio objetivo es colocado respecto a otros dominios de alto (aquellos en los que el dominio objetivo es una parte o es aplicable) y bajo nivel (o también llamados subdominios, aquellos que están en el ámbito del dominio objetivo), y también otros dominios (siempre que tengan alguna relación con el dominio objetivo). El segundo es un diagrama que muestra el flujo de información entre el dominio objetivo y el resto de dominios así como el resto de entidades con las que se comunica. La variabilidad del flujo de datos a través de los límites del dominio se debe indicar mediante diagramas o descripciones textuales.

  Esta fase permite tener una idea de:

  - el ámbito del dominio
  - los flujos de entrada/salida
  - los requisitos de almacenamiento de datos en el dominio

- **Modelado del dominio**: proporciona una descripción del problema en el dominio.

  El modelado del dominio permite encontrar lo común y las diferencias que caracterizan a las aplicaciones dentro del dominio. Esta fase, está compuesta a su vez de tres partes, subfases o actividades:

  - ERM (*Entity-Relationship Modeling*): obtiene y define el conocimiento y los requisitos de datos mínimos para poder implementar aplicaciones en el dominio. También incluye información que se asume proviene de fuentes externas.

    El conocimiento suele ser información que habitualmente está empotrada en el *software* y es difícil de seguir el rastro. Quienes mantienen o reutilizan *software* necesitan esta información para poder comprender los problemas que el dominio resuelve.
  - FA (*Feature Analysis*): obtiene una idea general de las funcionalidades que el usuario final quiere que tengan las diferentes aplicaciones.

    Estas funcionalidades se recogerán como características (*features*) en un diagrama, FM (*Feature Model*). Este diagrama representa lo común en el dominio así como la variabilidad dentro de dicho dominio; y también es un elemento de comunicación entre el usuario final y el desarrollador, para depurar el dominio.
  - FuA (*Functional Analysis*): identifica los flujos de datos e información de control comunes y diferentes entre las diferentes aplicaciones del dominio.

    En esta subfase se abstrae, para a continuación organizar las funciones comunes encontradas en el dominio y secuenciar en un modelo. Las características comunes y los ERM conforman la base del modelo funcional abstracto.

    El modelo funcional es la base sobre la que se comienza a comprender como proporcionar las características haciendo uso de las entidades seleccionadas.

  Al finalizar esta fase también se debe haber elaborado un diccionario del dominio, DD (*Domain Dictionary*), con términos y/o abreviaturas usados para describir las características y entidades,

así como una descripción textual de cada uno de ellos.

Este diccionario evita, en gran medida, las malinterpretaciones a los usuarios del dominio porque es el sitio donde pueden acudir para encontrar información sobre términos desconocidos.

- **Modelado de la arquitectura**: crea la arquitectura (o arquitecturas) necesaria para implementar las soluciones de los problemas del dominio.

  La búsqueda de procesos concurrentes y módulos o elementos comunes del dominio dirigen esta fase junto con la asignación de características, funciones y elementos de datos definidos en el modelo de dominio a los procesos y módulos.

  FODA describe hace uso de la metodología DARTS (*Design Approach for Real-Time Systems*) para el modelado arquitectónico.

  Concretamente, FODA se centra en las dos capas superiores que se pueden observar en la ilustración de la figura 2.5, la arquitectura del dominio, en la que se presentan[5] los diferentes procesos del dominio y las conexiones entre ellos, y las utilidades del dominio, en donde se proporcionan las diferentes funciones y objetos de datos organizados en paquetes dentro de módulos así como las conexiones o relaciones entre ellos.

  Las restantes capas se encargan de proporcionar los módulos comunes entre varios dominios (comunicación de procesos, sincronización,...) y los elementos básicos del sistema operativo o del lenguaje de programación usado respectivamente.

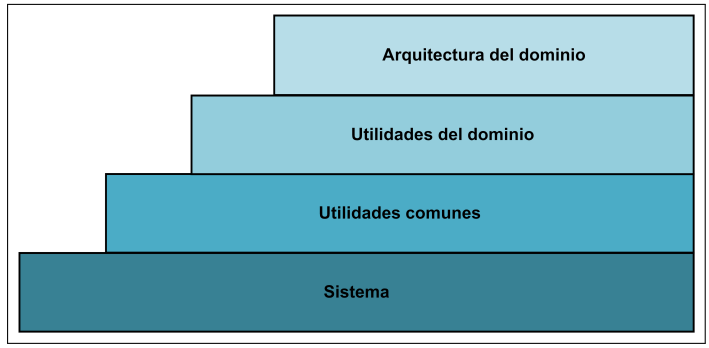


Figura 2.5: Esquema arquitectónico de capas

FODA define un método para realizar el análisis del dominio y también describe los productos resultado de ese análisis. En la ilustración de la figura 2.6 se muestran los tres componentes principales del modelo de dominio que se han explicado anteriormente.

El resultado de la selección de características es la definición de las funcionalidades que el sistema en desarrollo tendrá. El modelo funcional apoya tanto al modelo de características como al desarrollo arquitectónico. La selección de características parametrizará el modelo funcional estableciendo la dinámica de las funcionalidades del sistema.

Cuando se implemente el sistema, tanto el analista del dominio como el diseñador deben trabajar

[5] En FODA, este modelo se denomina modelo de interación de procesos, y como se ha indicado, hace uso de la metodología DARTS

juntos para decidir la arquitectura del sistema.

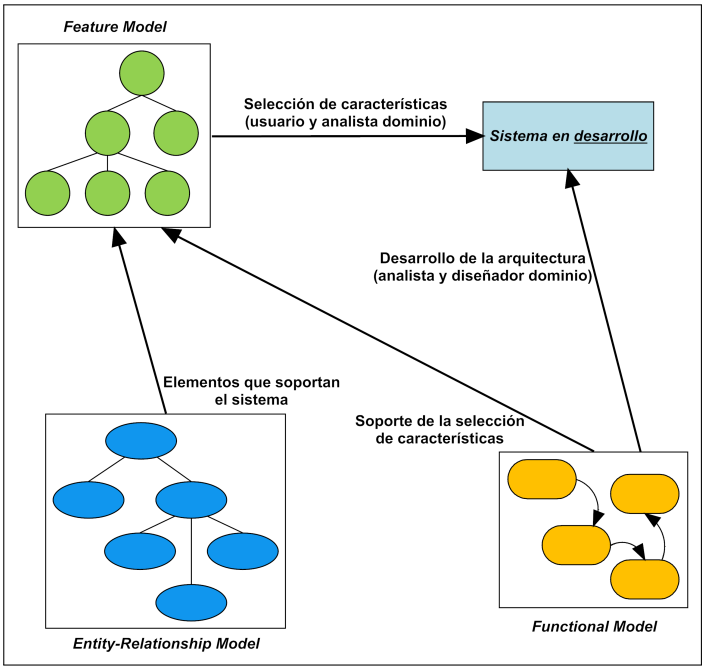


Figura 2.6: Modelo de dominio del sistema en desarrollo

El modelo funcional permite saber qué actividad genera determinada información, y también permite saber qué actividad necesita información. De este modo es posible establecer un flujo de datos. Haciendo uso del modelo de características, el diseñador podrá desarrollar una arquitectura que permita la implementación de los elementos comunes y también la inclusión y gestión de la variabilidad.

2.2.2. Modelado basado en características: FM

El modelado basado en características, *Feature Modeling*, es el proceso por el cual se busca lo común y lo variable de los conceptos[6] que definen una línea de productos *software*, así como las relaciones que puedan existir entre ellos, para posteriormente, organizarlo todo en un esquema jerarquizado denominado modelo de características, FM (*Feature Model*).

Este modelo describe todas las posibles variantes de productos *software* que se pueden obtener mediante la línea de productos a la que pertenezca. La generación[7] de un determinado producto *software* supone la eliminación de toda variabilidad posible que estuviese representada en el modelo de características, y a ello se llega mediante la selección o eliminación de las diferentes características (*feature*) y subcaracterísticas, para obtener la configuración concreta del producto *software* que se

[6]Por concepto debe entenderse cualquier funcionalidad o elemento, no atómicos (aquello que sea una entidad), que el cliente o el usuario final del producto *software* consideren relevantes

[7]También se podría denominar instanciación, o proceso de instanciación

desea obtener.

Los modelos de características representan la parte común, así como las dependencias existentes entre las características variables.

Un modelo de características es una colección, a su vez, de diagramas de características, a los que generalmente se acompaña de información anexa, por ejemplo, descripción de cada característica, restricciones entre características sin relaciones de dependencia, reglas genéricas de dependencias,...

Los diagramas de características están compuestos por un conjunto de nodos, una serie de líneas dirigidas y un conjunto de descriptores gráficos para las líneas (véase la ilustración de la figura 2.7). Los nodos y las líneas forman un árbol y los descriptores gráficos se dibujan en forma de arcos, de modo que conecten un grupo de líneas procedentes de un mismo nodo, o bien en forma de círculos rellenos o vacíos al final de cada línea.

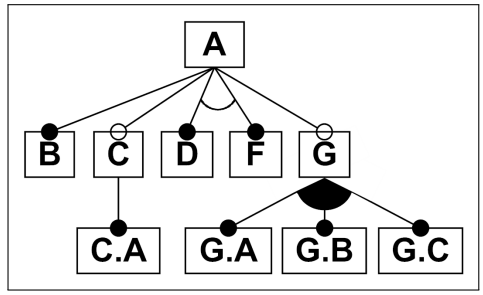


Figura 2.7: Ejemplo de representación de un diagrama de características

Una característica, un nodo en el diagrama, es un aspecto del sistema que es relevante para uno o varios involucrados en el desarrollo del producto: usuarios finales, clientes, ingenieros de dominio, diseñadores de *software*, desarrolladores,...

Cada característica puede ser común a todos los productos o bien puede ser variable, pero está incluida en al menos un producto de la línea de productos *software*. Las características se pueden unir entre sí mediante relaciones de jerarquía, dibujando una línea en el diagrama. También es posible establecer restricciones adicionales que no se puedan representar mediante relaciones entre características o características propiamente.

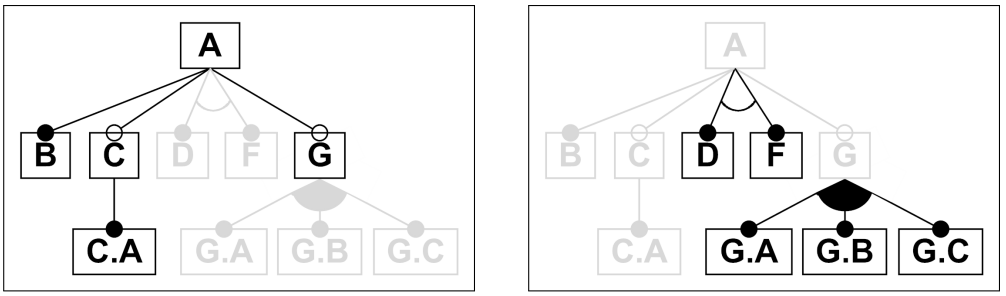
La organización jerárquica en forma de árbol permite definir características de primer, segundo, tercer,... nivel, es decir, un nodo del diagrama se describe y detalla mediante las características que tengan como nodo raíz a dicho nodo.

El motivo del uso o definición de características de primer nivel (o también subcaracterísticas), es debido a la modularización, ya que una característica de cualquier nivel puede ser descompuesta en subcaracterísticas que la detallen en profundidad.

También, la selección de una característica implica la selección de todos sus nodos padre (o ancestros), por la relación jerárquica en forma de árbol (no es posible llegar a un nodo del árbol si no se pasa por todos los nodos de los que depende para existir comenzando por el nodo raíz, con lo que, necesariamente, la selección de un nodo, implica la selección de todos los nodos por los que haya

que pasar para llegar a él partiendo del nodo raíz).

Es posible clasificar a las características de acuerdo a su obligatoriedad u optatividad. Otra clasificación, ilustrada en la figura 2.8, permite determinar si una característica pertenece a un grupo (característica de grupo), aquella que es colocada en el modelo como parte de un grupo de características, o si la característica es unitaria (característica unitaria), aquella que es colocada en el modelo como subcaracterística de otra. Para representar la pertenencia de una característica a un grupo ha de usarse el arco, cubriendo, con dicho arco, todas las características que se desee pertenezcan al mismo grupo.



(a) Características unitarias

(b) Características de grupo

Figura 2.8: Ejemplo de características unitarias y de grupo

El carácter de obligatoriedad u optatividad, al igual que el carácter de grupo, está dentro del concepto de cardinalidad. En el modelo de características, hay dos tipos:

- **Cardinalidad de característica**: frecuencia con la que una característica puede ser clonada como hija de una característica padre en la especificación de un producto concreto.

  Las cardinalidades se expresan entre corchetes; por ejemplo, `[1..k]`, indica que el número mínimo de clones puede ser uno y el máximo puede ser k en un producto concreto.

  Dos cardinalidades muy comunes, y que tienen una representación gráfica en el modelo de características son: `[1..1]` y `[0..1]`, la primera se corresponde con la obligatoriedad (un único clon) de la característica a la que acompaña, y la segunda se corresponde con la optatividad (uno o ningún clon) de la característica a la acompaña.

- **Cardinalidad de grupo**: número de características que pueden ser seleccionadas de una característica de grupo para un producto concreto.

  Este otro tipo de cardinalidad se expresa entre corchetes en forma de ángulo; por ejemplo, `<1..k>`, indica que al menos una, y como máximo k características podrán ser seleccionadas en una característica de grupo para un producto.

La notación mayoritariamente aceptada hoy en día en los modelos de características es la notación *Czarnecki-Eisenecker*, así como la versión extendida. Una de las principales ventajas es la compatibilidad con la notación usada en el estudio FODA y otra es las nuevas posibilidades que ofrece.

En lugar de explicar cada uno de los elementos de esta notación, se ilustrará mediante la ilustración mostrada en la figura 2.9; en la que se parte de la suposición de que la figura representa el concepto

`Compra` incluido en todos los productos de la línea de productos dónde esté ésta característica.

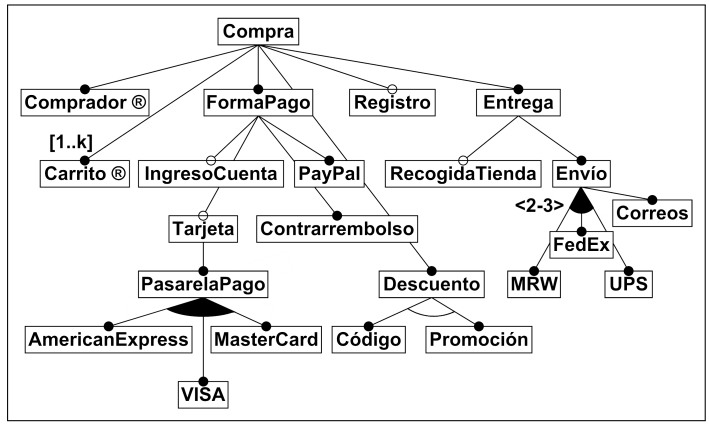


Figura 2.9: Ejemplo de notación en el modelo de características

Toda característica (o subcaracterística o característica de segundo, tercero,... nivel) se modela en el diagrama en un recuadro con un nombre simbólico y descriptivo: por ejemplo, `Comprador`, característica que representa a las subcaracterísticas[8] del comprador de toda venta.

Las características se unen entre sí mediante líneas, organizándolas en forma de árbol; por ejemplo, `PayPal` es una subcaracterística, o característica hija, de la característica padre, `FormaPago`, que a su vez es una subcaracterística de la característica `Compra`; por lo tanto se está indicando que toda compra tiene una forma de pago, y entre sus formas de pago, estaría el servicio PayPal.

Los círculos rellenos o vacíos indican el carácter de obligatoriedad u optatividad respectivamente. Para el caso anterior, se indica que, para todos los productos de la línea dentro de las formas de pago, la característica `PayPal` estará en todos los productos de la línea, pero el pago mediante tarjeta podrá estar o no. Fíjese que si se decide que para un producto el pago por tarjeta esté disponible, automáticamente, la característica `PasarelaPago` pasa a formar parte del producto, ya que está marcada como obligatoria (no olvidando que es obligatoria si y sólo si la característica `Tarjeta` se marca en el modelo).

Los arcos, también rellenos o vacíos, permiten agrupar características (lógicamente, el arco agrupa como mínimo a dos características y como máximo a todas las que sean subcaracterística de la misma característica y que estén en el mismo nivel). Un arco relleno responde a la función lógica OR y, por lo tanto, se ha de seleccionar como mínimo una característica y, como máximo, todas las que agrupe. Por ejemplo, la característica `PasarelaPago` establece tres pasarelas de pago por tarjeta, de las que hay que integrar al menos una y como máximo las tres. Pero obsérvese lo que ocurre en la característica `Envío`; por un lado está la subcaracterística `Correos`, que ha de ser incluída obligatoriamente y por otro lado están las otras tres: `MRW`, `FedEx` y `UPS`; en principio, habría que escoger o una o dos o las tres, pero aparece al lado una cardinalidad, `<2-3>`, la cual indica que como mínimo se deben escoger dos y como máximo tres.

[8]El hecho de haber escogido esta característica no es casual, el símbolo ®, indica que el subárbol descriptor de ésta característica, a la que acompaña el símbolo, está descrito en otro sitio. Sin embargo, por cuestiones de espacio y de brevedad en este ejemplo, no se muestra

El arco vacío tiene el mismo significado que el arco relleno, a excepción de la función lógica, que en este caso es la función XOR; por ejemplo, la característica `Descuento` es obligatoria, pero hay que decidir cómo será el sistema de descuentos: mediante códigos o mediante promociones (algo así como proporcionar unos símbolos alfanuméricos o bien, de acuerdo al producto comprado, es posible que podamos optar a un descuento por alguna campaña promocional).

La cardinalidad `[1..k]` que acompaña a la característica `Carrito`, se refiere a la multiplicidad que hay que establecer para cada producto, y en este ejemplo, se dice que toda compra ha de tener al menos un carrito, y como máximo los que se desee (la decisión del número máximo de carritos para cada compra, muy probablemente, sería del programador o del diseñador, porque los recursos son limitados y esta funcionalidad se podría integrar en un único carrito; otra cosa bien distinta, es la usabilidad que una opción u otra pueda suponer, pero aquí no se tratará ese tema).

Otra forma de visualizar el diagrama de características mostrado en la ilustración de la figura 2.9, es el que se puede observar en la ilustración de la figura 2.10. Este diagrama hace uso de la notación extendida de *Czarnecki-Eisenecker*.

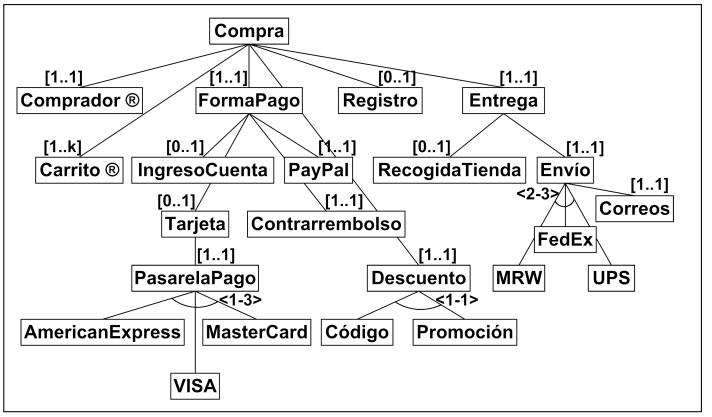


Figura 2.10: Modelo de características expresado en notación extendida

En el apéndice A se encuentra una propuesta de un meta-modelo (en inglés) realizada por Valentino Vranić[9] para la representación del modelo de características con todos sus elementos, que además está expresada en términos del modelo de características (utilizando la notación de *Czarnecki-Eisenecker*).

Notación de *Czarnecki-Eisenecker*

La notación de *Czarnecki-Eisenecker* permite establecer una serie de restricciones en el diagrama en una rama o subrama, aprovechando la estructura jerárquica en forma de árbol, pero no hay notación para expresar restricciones entre ramas o subramas, es decir, no es posible restringir, por ejemplo, la existencia de una característica de una rama, en función de la existencia de otra carac-

[9] Investigador perteneciente al Instituto de Informática e Ingeniería del Software en la Facultad de Informática y Tecnologías de la Información de la Universidad Tecnológica Eslovaca en Bratislava

terística en otra rama.

Estas relaciones han de ser especificadas en documento aparte, anexo al modelo de características. Las restricciones que es posible establecer son de dos tipos: restricciones "fuertes", en tanto que condicionan el modelo porque influyen activamente en él a la hora de la toma de decisiones; y que a su vez, se dividen en *require* (una característica requiere la selección de otra) y *exclude* (una característica excluye la selección de otra característica), y restricciones "débiles", en tanto que son recomendaciones que no es necesario seguir; y se dividen en *recommend* (la selección de una característica recomienda la selección de otra) y *discourage* (la selección de una característica desaconseja la selección de otra característica).

Este tipo de relaciones no tiene por qué ser bidireccional; lo habitual es que si una característica requiere a otra, la selección de esta última recomiende la selección de la primera, pero también puede haber independencia en un sentido, e incluso independencia parcial. Sin embargo, son muy importantes a la hora de establecer restricciones entre características de diferentes subramas.

Proceso de análisis

El proceso de análisis, para la obtención del modelo de características, se compone de las siguientes fases:

- **Recogida de información**: las fuentes de información que servirán de base para la identificación de las características, como los libros (se considera una buena fuente para investigar sobre teorías, técnicas, metodologías, modelos,... pero no aportan una visión amplia sino la del autor), los estándares (son una referencia válida y aceptada pero pueden estar obsoletos, y, por tanto, no ser aplicables en el momento del análisis), las aplicaciones existentes (son la fuente de información más importante porque permiten encontrar las características con facilidad, porque hay documentación ya escrita y porque también es posible conocer otros aspectos como la arquitectura utilizada o incluso la implementación, pero, todo esto a costa de analizar muchos sistemas), y los expertos de dominio (pueden proporcionar información que no se encuentra en ningún otro sitio, pero es posible que sea necesario disponer de varios debido a que cada uno abarca un área de conocimiento concreto).

- **Identificación de las características**: las características se pueden clasificar, de acuerdo a FODA, en cuatro categorías, según se puede ver en la ilustración de la figura 2.11. A su vez, la funcionalidad se puede dividir en tres subcategorías: características "funcionales" (principalmente los servicios ofrecidos por los productos *software*; se podría relacionar con los requisitos funcionales), características operacionales (las relacionadas con la forma de proceder del usuario para realizar las tareas) y las características de presentación (todas aquellas que tengan algo que ver con el cómo y el qué se muestra al usuario).

  Esta clasificación no es única y se puede ampliar; la práctica permitirá aumentar las categorías y mejorar la clasificación. Para cada característica identificada se debe poner un nombre descriptivo.

- **Elaboración del modelo**: a partir de la colección de características elaborada anteriormente, se ha de elaborar un modelo jerárquico, clasificando y estructurando las características mediante relaciones; al mismo tiempo, o en una fase posterior, hay que establecer el carácter obligatorio, opcional o alternativo de las características (o lo que es lo mismo, establecer las cardinalidades).

También ha de indicarse el momento en el que cada característica será incorporada al producto (tiempo de compilación, de carga o de ejecución). Por último, han de anotarse las restricciones que puedan existir entre características no relacionadas directamente entre sí.

- **Validación**: este proceso deberá ser realizado por alguien que no haya intervenido en ninguna fase del análisis para minimizar cualquier influencia. Si es posible, se debe realizar la validación haciendo uso de alguna aplicación existente como modelo (lógicamente, mientras más aplicaciones-modelo se disponga mejor), que no haya sido utilizada durante la recogida de información para determinar el grado de variabilidad y los elementos comunes.

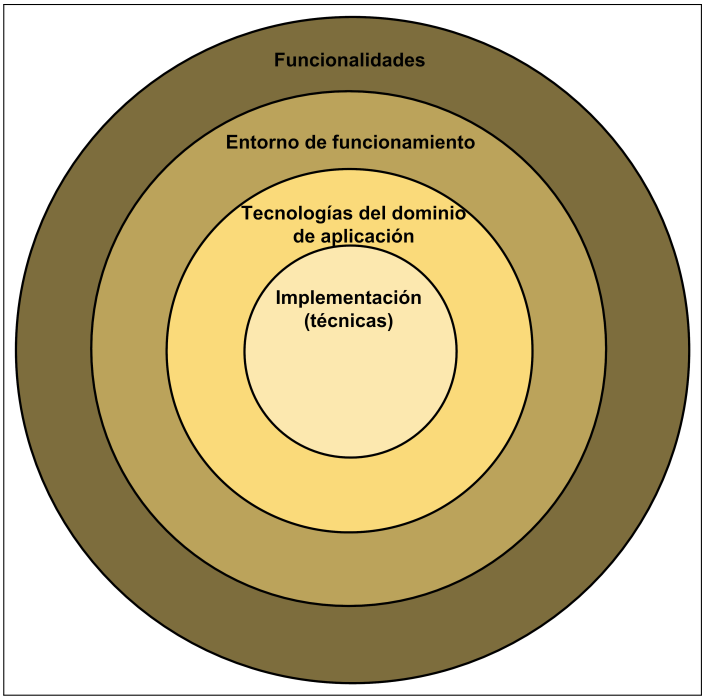


Figura 2.11: Categorización de características

El modelo de características no tiene utilidad alguna si no es configurado, es decir, si no se concretan los puntos de variabilidad que puedan existir en él, para obtener un modelo que represente las funcionalidades (o características) que un determinado producto *software* de la línea de productos tendrá.

Proceso de configuración

La configuración es el proceso por el que se eliminan los puntos de variabilidad del modelo de características mediante la selección o eliminación de características. El resultado de este proceso es o bien una especialización, un modelo de características intermedio que todavía tiene puntos de decisión sobre los que actuar, o bien una configuración, un modelo de características que representa a un producto *software* concreto de la línea de productos, ya que toda variabilidad existente ha sido

eliminada.

Las especializaciones son útiles cuando no se desea eliminar toda la variabilidad en una única iteración; por lo tanto la creación[10] de un producto ocurre varias veces, tantas como iteraciones se realicen. Esta forma de proceder se denomina configuración multi-etapa. En cada iteración sucesiva que se haga, la variabilidad es menor a la de la etapa anterior, por lo tanto, toda decisión en una etapa se ve reflejada en las siguientes etapas.

En la ilustración de la figura 2.12, se puede observar cómo en cada fase del proceso de desarrollo[11] el modelo de características es refinado poco a poco, y para cada fase, resulta una especialización, realizada por la persona o personas involucradas en esa fase, que es la base para la siguiente fase, hasta llegar a la fase de ejecución dónde ya no hay ningún punto de variabilidad.

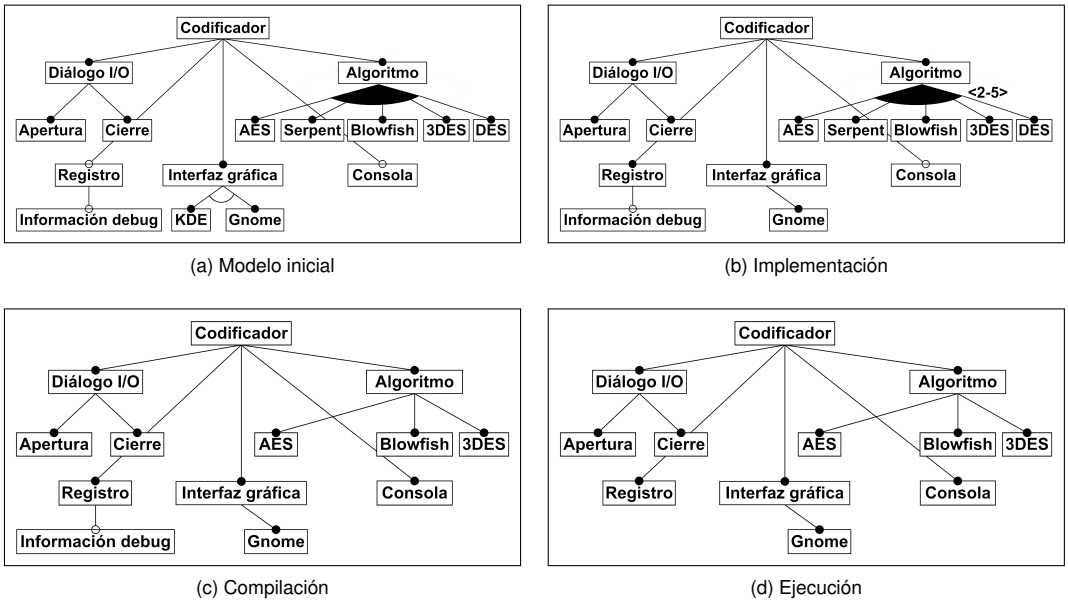


Figura 2.12: Configuración por etapas del modelo de características

Otra forma de proceder se denomina configuración multi-nivel; en la cual existen diferentes niveles, cada uno de ellos representado mediante su correspondiente modelo de características. El objetivo de esta forma de proceder está en disponer de distintos modelos de características de modo que representen diferentes niveles de abstracción, resultando en una especialización automática de los modelos de características de los niveles siguientes.

La configuración multi-nivel permite minimizar uno de los mayores problemas en el diseño del modelo de características, la "parálisis" en el análisis, es decir, el bloquear el modelo de características

[10] Si bien se usa el término de creación para la generación u obtención de un producto *software* a partir de una línea de productos, también se podría usar el término de instanciación

[11] El ejemplo mostrado aquí está simplificado y por tanto ni muestra todas las fases ni lo hace en detalle. Nótese la introducción de la cardinalidad en la etapa de la implementación, compatible con el modelo inicial

por alguna discrepancia o problema en el modelo de características, que no permite continuar con su normal desarrollo.

En la ilustración de la figura 2.13, se muestra un pequeño ejemplo en el que se ilustra lo explicado. Se han dispuesto dos niveles, el nível de la línea de productos, nivel 0, y el nivel del sistema, nivel 1; cada uno de ellos dispone de su propio modelo de características.

En una primera etapa se produce una especialización en ambos niveles; en el nivel 0 se realiza una configuración manual por parte del ingeniero encargado del dominio de análisis; esa especialización afectará a la misma etapa, pero en el nivel 1, en tanto que condicionará las posibles decisiones que se puedan tomar en el modelo de características del nivel 1 porque, automaticamente, la especialización de la primera etapa del nivel 0 ha provocado[12] una especialización en el modelo de características del nivel 1.

En una segunda etapa, el modelo de características del nivel 0 ya no sufre más especializaciones porque ya no tiene ningún punto de variabilidad, y por tanto, se convierte en la configuración del nivel 0. El modelo de características del nivel 1 sufre una nueva especialización, aunque esta vez será manual, y será realizada por los agentes involucrados en esta nueva etapa. Por último, al finalizar esta etapa, se dispondrá de la configuración para el nivel 1.

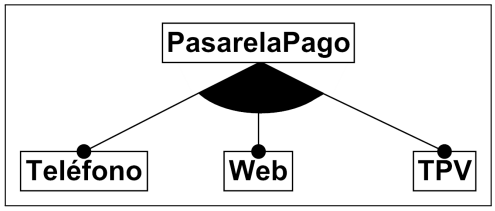
La toma de decisiones, especialmente patente durante la configuración, es un proceso que se conoce como *binding time*. Estas decisiones están marcadas por el modelo de negocio y de trabajo que se siga en la línea de prodcutos *software*, así como el desarrollo y despliegue de los productos.

Si bien hay mútiples *binding time* a lo largo del desarrollo, es relevante indicar dos de ellos: *build-time* (tiempo de desarrollo/construcción), que se produce durante el proceso de desarrollo principalmente, y donde las decisiones, normalmente basadas en los requisitos o especificaciones, que se toman sobre el modelo de características afectan directamente a la funcionalidad que tendrá o dejará de tener el producto *software*; y *run-time*, que se produce después de la configuración, durante el despliegue y uso diario del producto (generalmente las decisiones en este punto suelen afectar a la propia configuración del programa, por ejemplo, cambiar el tema visual del programa,...).

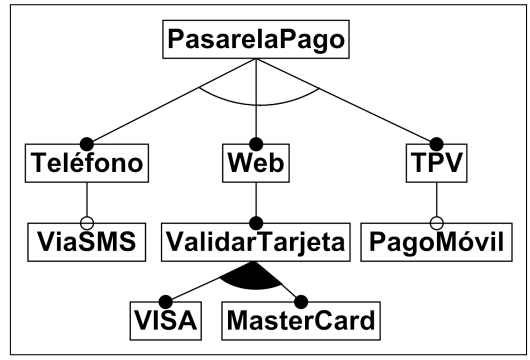
No suele haber un momento establecido para la toma de decisiones en cada fase porque resulta complicado conocer el comienzo y el fin de cada fase, especialmente en las configuraciones multi-etapa y multi-nivel. Dependiendo de las funcionalidades, las fases se pueden solapar más o incluso no ser diferenciables (por ejemplo, el uso de *plug-ins* puede alterar el flujo de configuración planificado). Incluso es posible que haya subfases dentro de una fase.

El análisis basado en modelos de características proporciona una representación abstracta (ya que es independiente de la implementación de la que se haga uso), concisa y explícita de la variabilidad presente en una línea de productos, así como de la parte común a todos los productos de la línea de productos.

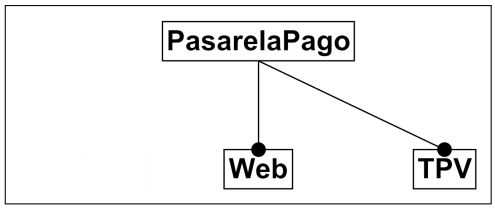
[12] Se podría interpretar como un proceso de solapamiento de ambos modelos en donde el modelo de un nivel inferior condiciona aquello que tenga en común con los modelos de niveles superiores



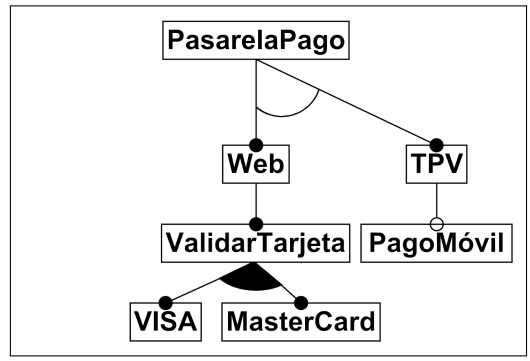
(a) Nivel 0 - Etapa 0



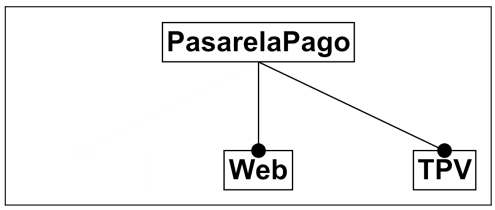
(b) Nivel 1 - Etapa 0



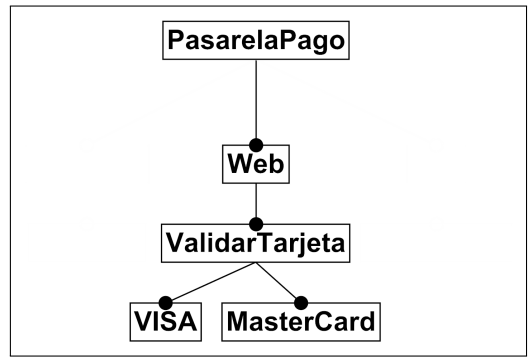
(c) Nivel 0 - Etapa 1



(d) Nivel 1 - Etapa 1



(e) Nivel 0 - Etapa 2



(f) Nivel 1 - Etapa 2

Figura 2.13: Configuración por niveles, y etapas, del modelo de características

2.3. El modelo de características y su relación con UML

El modelo de características, elemento clave en la elaboración de un producto *software* concreto en una línea de productos por su capacidad para gestionar la variabilidad eficientemente, no tiene ninguna utilidad si no es integrado de alguna forma en el propio proceso de desarrollo de la línea.

Hoy en día, UML (*Unified Modeling Language*) es el lenguaje de modelado de *software* por excelencia en la ingeniería del *software*, ya que es una especificación visual (con su propia notación, semántica, reglas y un meta-modelo) para la creación de un modelo abstracto de un sistema, lo que se conoce como "modelo UML".

Dada la importancia de UML en la mayoría de procesos *software* es lógico pensar que será el lenguaje de modelado a utilizar junto con los modelos de característas en el desarrollo de la línea de productos *software*. Sin embargo, hoy en día, UML no soporta los modelos de características, por lo que, en principio no sería posible incluirlos de acuerdo a las especificaciones del estándar.

Por lo tanto, es importante encontrar una solución a este problema, bien integrando en el meta-modelo de UML una representación para los modelos de características o bien aprovechando los elementos del propio lenguaje incluyendo nuevas notaciones mediante estereotipos.

Diversas soluciones han sido propuestas desde la presentación del estudio FODA, y los modelos de características. A continuación se describen brevemente algunas de las diferentes propuestas realizadas que se han encontrado:

- von der Maßen y Lichter [39] proponen hacer uso de una nueva notación gráfica estableciendo las relaciones: `option` y `alternative`, lo cual implica una modificación del meta-modelo de UML.

- Halmans y Pohl [19] consideran válida la modificación de los casos de uso (mediante anotaciones) para representar gráficamente los puntos de variación, y una vez más, se vuelve a proponer la modificación del meta-modelo de UML.

- John y Muthig [22] proponen el uso de plantillas de casos de uso haciendo uso de estereotipos, por lo que también implica un cambio del meta-modelo de UML; aunque no distinguen entre variantes obligatorias, alternativas u opcionales.

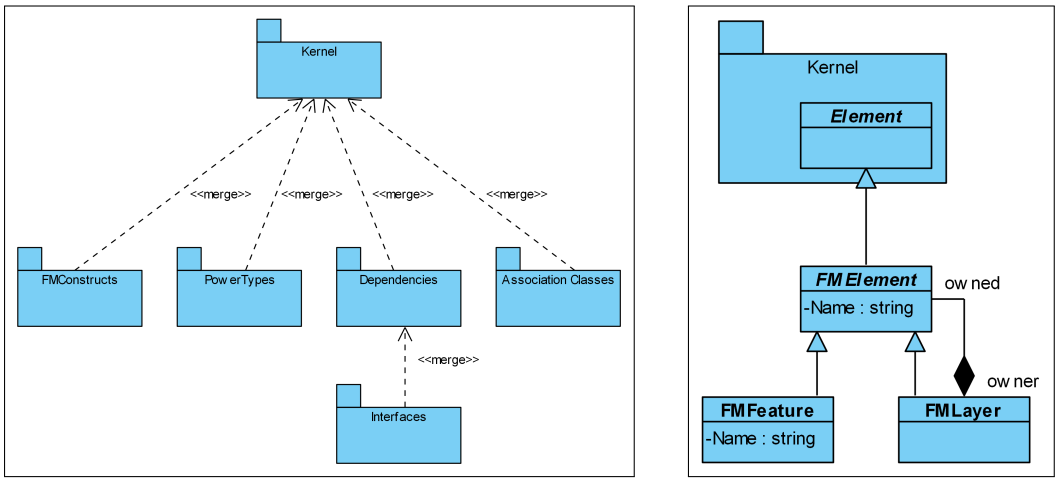
- Vranić y Śnirc [45] sugieren la extensión del meta-modelo de UML desde la raíz del mismo, es decir, desde las meta-clases más abstractas.

  Con esta propuesta, la extensión del modelo de características es englobada en el paquete `FMConstructs` que es combinado con el paquete `Kernel` de UML mediante la operación `<<merge>>` para garantizar la propagación al resto de elementos del lenguaje.

  Los elementos para la construcción de un modelo de características son incluidos en una meta-clase abstracta, `FMElement`. Para evitar que incluir cualquier semántica en el lenguaje, ésta deriva de la meta-clase del modelo UML, `Element`. Las características se modelarán como subclases de la clase `FMElement`.

  Todas las restricciones y reglas de dependencia son representadas mediante expresiones lógicas, aunque si el meta-modelo propuesto fuese integrado en UML, se podría usar OCL (*Object Constraint Language*) en su lugar.

Esta propuesta, a la par que ambiciosa, es interesante porque parece cubrir los aspectos en los que otras propuestas fallaban, pero sigue teniendo el problema de un cambio en el meta-modelo para poder soportarla.



(a) Jerarquía de paquetes

(b) Jerarquía de clases

Figura 2.14: Ejemplo de la solución propuesta por Vranić y Śnirc

- Jacobson y otros [20] propusieron el uso de los elementos de UML: relación de especialización, multiplicidad de asociaciones,...

  Sin embargo, esta propuesta no distingue entre las diferentes variaciones, por ejemplo durante el tiempo de configuración y durante el tiempo de ejecución. Es decir, no distingue entre la variabilidad de la propia línea de productos *software* y la variabilidad de cada producto *software* concreto.

- Gomaa [16] propone el modelado de los modelos de características, dependiendo del modelo a realizar, como paquetes o clases estereotipadas («`kernel`», «`optional`» y «`variant`», que corresponderían a las clases obligatorias, opcionales y alternativas respectivamente) en las que las características de grupo son modeladas como agregados.

  El uso de agregados resulta problemático en tanto que hay que tomar decisiones en la fase de análisis durante la cual se confeccionan los modelos de características, es decir, obliga a tomar ciertas decisiones que se habían postergado a fases posteriores.

- Clauß [5] sugiere la extensión de UML mediante la representación de un concepto y una característica como estereotipos de una clase. Donde los estereotipos para las clases serían: «`mandatory`», «`optional`» y «`alternative`» (obligatorio, opcional y alternativo respectivamente) y los estereotipos para denotar la variabilidad serían: «`variationPoint`» y «`variant`» (puntos de variación y sus subclases variantes respectivamente).

  El principal inconveniente de esta propuesta es el mismo que la propuesta de Gomaa. A mayores, se presenta algún problema con el arco XOR, que se representa aplicando {`xor`} entre los extremos de los bordes respectivos, pues si bien se puede usar para particionar las subcaracterísticas de una característica en grupos de características, no es posible denotar varios grupos

OR independientes dentro de una misma característica, lo cual es una limitación bastante importante.

- Griss y otros [17] consideran el uso de estereotipos pero no hay una distinción hacia los conceptos. Aunque es posible tener un modelo de características como un grafo.

  Las características junto con su variabilidad son modeladas mediante atributos de una clase, permitiendo uno de ellos que una característica tenga el rol de punto de variación o característica alternativa, y esto trae consigo el inconveniente de tener un único grupo de subcaracterísticas por característica.

  Las relaciones de las subcaracterísticas con su caracterísica "padre" son modeladas mediante dependencias con el estereotipo `«consists_of»`.

- Dolog y Nejdl [14] proponen estereotipos en las clases para representar las características, pero tratan las características en grupo mediante puntos de variación, y de este modo consiguen diagramas de características en forma de grafos dirigidos y acíclicos aunque la información de la variabilidad no está correctamente incluida en las propias características.

- Czarnecki y Antkiewicz [8] sugieren anotar en los diagramas de UML las condiciones de presencia mediante un código de colores, si bien plantea el problema de la escalabilidad en cuanto se aumentan las variantes.

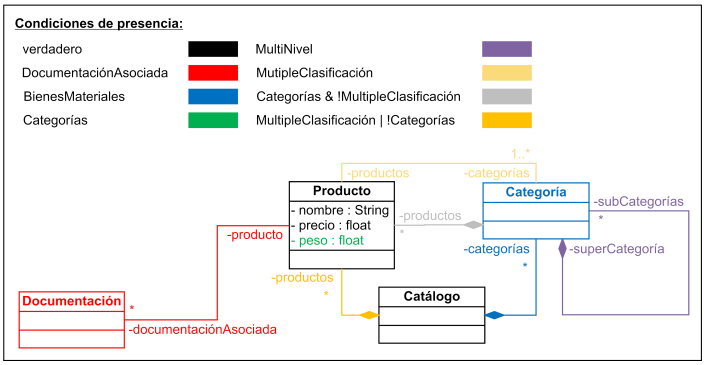
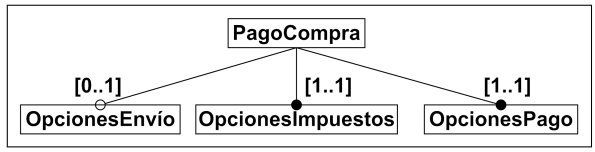


Figura 2.15: Ejemplo de la solución propuesta por Czarnecki y Antkiewicz

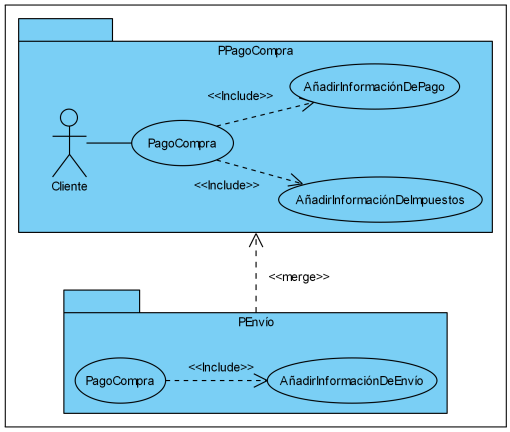
- Laguna y González-Baixauli [30] proponen[13] una solución basada en la combinación de paquetes en el modelado de sistemas [47], llegando a la conclusión de que si bien respeta las propiedades de unicidad y asociatividad, no ocurre lo mismo con la propiedad de conmutatividad. Por lo que, al no poder realizar una combinación de dos o más paquetes en un orden arbitrario (algo que se cumpliría en la propiedad de conmutatividad), dado que influiría en el resultado, en principio, no se podría.

  Sim embargo, la propuesta de Laguna y González-Baixauli construye los modelos de paquetes de forma jerarquizada, por lo que evitan el problema que se acaba de mencionar, ya que se garantiza que un paquete receptor siempre se relaciona con un único paquete base.

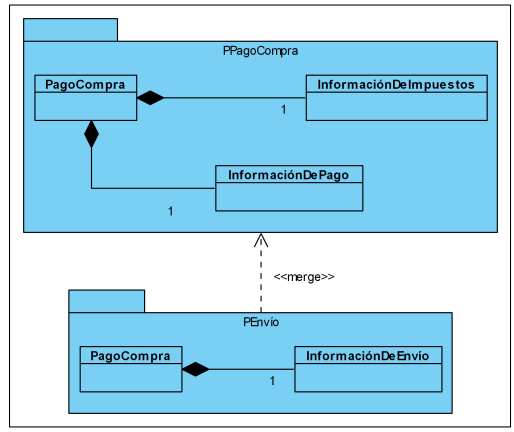
[13] Esta propuesta se explicará con más detalle en el apartado



(a) Modelo de características



(b) Diagrama de casos de uso



(c) Diagrama de clases

Figura 2.16: Ejemplo de la solución propuesta por Laguna y González-Baixauli

Como se puede observar, todas las propuestas son perfectamente válidas, pero o bien obligan a modificar el meta-modelo de UML[14], lo cual conlleva inevitablemente a tener que aprender esas modificaciones, así como esperar a su integración en las herramientas CASE (*Computer Aided Software Engineering*) y al posterior aprendizaje de uso en la herramienta CASE, o bien se trata de artefactos añadidos al modelo UML mediante estereotipos y especializaciones, que además de complicar la comprensión del diagrama, en algunos casos no permiten realizar la representación completa en UML del modelo de características.

No obstante, la última propuesta [30] plantea un modelado basado en la combinación de paquetes, es decir, hace uso del propio lenguaje UML sin modificarlo, y únicamente aplica una serie de patrones para la transformación del modelo de características en diagramas de casos de uso y de clases (aunque también sería posible su aplicación a otro tipo de diagramas que se pueden construir con el meta-modelo de UML).

Esta misma propuesta establece un conjunto de requisitos mínimos, que a juicio de los autores, debería cumplir toda técnica de representación y gestión de la variabilidad a nivel de diseño, en UML, de una línea de productos *software*, y que ha sido obtenida mediante el análisis de los puntos fuertes y débiles de varias de las propuestas citadas anteriormente. Dichos requisitos se enumeran a continuación:

- Localizar en un único punto del modelo de diseño todas la variabilidad que produce cada

[14] *Czarnecki y Eisenecker* denominan [10] a estas estrategias "*diagram hacking*" (engaño o rodeo al diagrama) ya que añaden al meta-modelo UML lo que consideran necesario para poder realizar el modelado en lugar de utilizar los propios elementos que el meta-modelo de UML ofrece

característica opcional, de modo que se mantenga una correspondencia uno a uno y se facilite la gestión de la trazabilidad.

- Separar la variabilidad originada en el nivel de la línea de productos *software* de la variabilidad originada en el nivel de las aplicaciones concretas, eliminando ambigüedades (y minimizando las posibles restricciones que hubiese que hacer y también tener en cuenta).

- Mantener inalterado el meta-modelo de UML para eliminar la barrera de entrada a este paradigma para cualquier desarrollador, además de permitir el uso de herramientas CASE convencionales (evitando tener que aprender nuevos artefactos de modelado en UML que no sólo suponen tiempo sino complejidad en el modelo).

- Conectar con los modelos de implementación para acercarse al ideal de "*seamlesss development*" (desarrollo sin costuras).

2.3.1. Combinación de paquetes en UML2

El meta-modelo de UML2 incluye el concepto de "*package merging*" ó "*package merge*" (combinación de paquetes).

Este mecanismo se basa en el proceso de ir añadiendo detalles de forma incremental mediante una relación similar a la generalización, `<<merge>>`, que sólo es válida[15] cuando se tienen un conjunto de elementos en distintos paquetes tales que al menos hay un elemento en dos o más paquetes con el mismo nombre simbólico, logicamente representando el mismo concepto, y partiendo de una base común.

UML2 utiliza dos conjuntos de conceptos principalmente:

- **Paquete combinable**, **paquete receptor** y **paquete resultado**: Donde el paquete combinable es el primer operando y el paquete receptor es el segundo operando, y el resultado de la operación de combinación es el paquete resultado, aunque este último en el modelo coincide con el paquete receptor (una vez se ha realizado la operación).

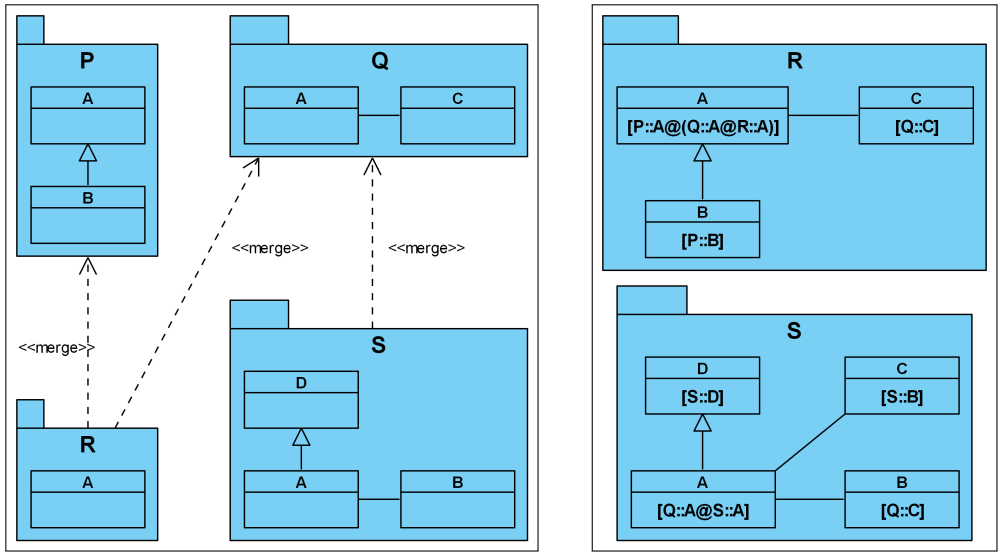
- **Elemento combinable**, **elemento receptor** y **elemento resultado**: Donde el elemento combinable pertenece al paquete combinable, el elemento receptor pertenece al paquete receptor y en caso de haber coincidencia con otro elemento combinable se fusiona con él, y el resultado de la combinación de los dos elementos da lugar al elemento resultado.

  Los elementos que no coinciden en el conjunto de elementos combinables y receptores, se incorporan al paquete resultado sin modificación alguna.

De este modo, según se ilustra[16] en la figura 2.17, un concepto es incrementado poco a poco (por cada nueva combinación de un paquete). Por ejemplo, el paquete `R` contiene a las clases `A`, `B` y `C`; la clase `B` (que proviene del paquete `P`) es una especialización de la clase `A`, que a su vez es el resultado de la combinación de la clase `A` del paquete `P` con la combinación de la clase `A` del paquete `Q` y de la clase `A` del propio paquete `R`.

[15] Válida debe entenderse en este contexto como apta o adecuada en el uso

[16] En las expresiones de las clases, el símbolo @ representa al operador de combinación. Estas expresiones no forman parte de ninguna notación estándar



(a) Mecanismo de combinación

(b) Resultado de la combinación

Figura 2.17: Ejemplo de la combinación de paquetes (extraído de la especificación de UML2)

Si bien el uso más frecuente que se puede dar a este mecanismo es en el diagrama de clases (y de paquetes), el meta-modelo de UML define una serie de reglas generales que son aplicables al resto de elementos del meta-modelo.

2.3.2. Transformación del modelo de características en modelos de UML2

Para el modelado en UML, se ha de establecer un paquete base que reunirá toda la parte común de la línea de productos *software*. Este paquete base podrá ser organizado utilizando descomposición recursiva en paquetes y estableciendo las diferentes relaciones, `<<import>>` y `<<access>>`, que pueda haber entre ellos.

Para cada característica opcional se creará un paquete que se añadirá al paquete base, de modo que este paquete reúna todas las modificicaciones del diseño asociadas a la característica. Este paquete añadido se unirá al paquete base mediante la operación `<<merge>>`.

La ilustración de la figura 2.18 muestra un pequeño ejemplo en donde un catálogo tiene una estructura y un registro obligatorios en la línea de productos *software* (por lo tanto serán características, y posteriormente funcionalidades, que compartirán todos los productos de la línea) y opcionalmente una sección de ofertas.

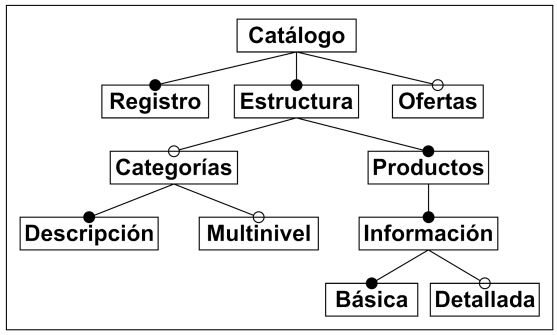
La estructura del catálogo, modelada en el paquete `PEstructuraCatálogo`, tendría una búsqueda, el caso de uso `BuscarEnCatálogo`, y una selección de productos, el caso de uso `SeleccionarProducto` (que incluiría al caso de uso `VerInformaciónProducto`).

De acuerdo al modelo de características, la búsqueda en el catálogo se puede hacer por categorías, y por ser opcional el caso de uso correspondiente, `SeleccionarCategoría`, se coloca en el paquete

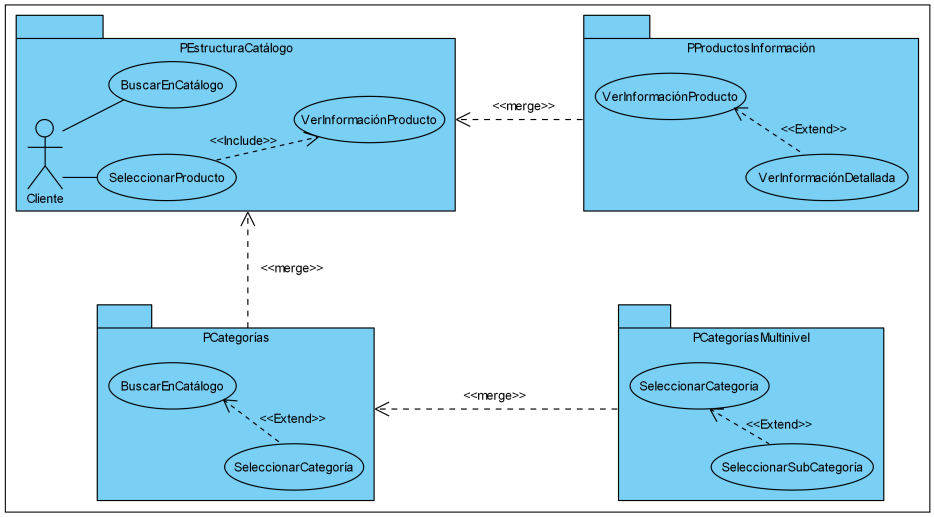
PCategorías y se relaciona con el paquete base mediante la operación <<merge>>; lo mismo ocurre con la subcaracterística opcional, Multinivel, de la característica Categorías, que ha sido modelada en el caso de uso SeleccionarSubCategoría dentro del paquete PCategoríasMultinivel, y a su vez este paquete es unido con su paquete base, PCategorías, mediante la operación <<merge>>.

Igualmente ocurre con la característica Detallada, que es modelada como el caso de uso VerInformaciónDetallada dentro del paquete PProductosInformación, unido al paquete base PEstructuraCatálogo (el paquete donde se encuentra representada la característica Información y por extensión Producto) mediante la operación <<merge>>.

Las características Descripción e Información que no han sido modeladas mediante casos de uso, son un ejemplo de "características de estructura", es decir no representan ni caracterizan ningún comportamiento, y por lo tanto, se harán visibles en el diagrama de clases.



(a) Modelo de características



(b) Aplicación al diagrama de casos de uso

Figura 2.18: Ejemplo de la transformación del modelo de características al meta-modelo UML2

Es importante darse cuenta que el caso de uso que modela a la característica padre de una ca-

racterística opcional también es incluido en el paquete donde es modelada la característica opcional además de en el paquete base al que corresponda; no obstante, no se incluyen todos los escenarios del caso de uso, sino sólo las partes que afecten al caso de uso que represente a la característica opcional, en tanto que no tiene sentido describir escenarios o parte de ellos en los que el escenario de la característica opcional no tiene nada que ver.

2.4. Metodología de trabajo

Una vez expuesta la base teórica en las secciones anteriores de este capítulo, en esta última sección se expondrá la metodología que se seguirá en los siguientes capítulos, así como las herramientas utilizadas.

El motivo de haber elegido una asociación como contexto de estudio para el desarrollo de una línea de prodcutos *software* se debió a la pertenencia del autor de este Proyecto Fin de Carrera a una asociación, y aprovechando ambas circunstancias, se elaboró una propuesta de Proyecto Fin de Carrera, con el objetivo de desarrollar una aplicación que permitiese mejorar los problemas de coordinación y comunicación interna existentes en la asociación para la que se pretendía desarrollar la aplicación (posteriormente se detallará como se trata de un problema más o menos común en otras asociaciones con las que se ha contactado para ampliar el contexto del estudio).

No obstante, también se pensó en otros entornos (por ejemplo, una fundación, un departamento,...) en los que se cumpliese el requisito de ser un conjunto de personas con unos fines comunes y una necesidad de organización a través de un sistema de información, aunque finalmente no han sido contemplados, dada la amplitud de la envergadura de una línea de productos para asociaciones; pero no hay que olvidar que gracias a las técnicas y metodologías explicadas, el estudio en otros ámbitos relacionados no sería un trabajo en el que se partiese de cero.

En una fase preliminar del dominio, no documentada en esta memoria, se observó que si bien había una serie de elementos comunes en la gestión de toda asociación, también había un grado de variabilidad y ese era el factor que diferenciaba a una asociación de otra.

Por ello, se optó por el estudio de una línea de productos *software* para asociaciones, ya que no sólo se buscaba una aplicación específica para una asociación, sino una aplicación que tuviese un conjunto de usuarios más amplio, y una línea de productos permitía manejar y representar la variabilidad y lo común de las diferentes variantes que se pudiesen contemplar, al mismo tiempo que permitía establecer una forma de desarrollo de aplicaciones *software* diferente al modelo tradicional y orientada a la reutilización de componentes *software*, y a la producción, más o menos automátizada, de productos *software*.

2.4.1. Análisis del dominio de una línea de productos para asociaciones

Al no encontrar mucha literatura, así como aplicaciones suficientes como para poder realizar un estudio del dominio de la línea de productos lo suficientemente amplio, se optó por la búsqueda de información en otros dominios no relacionados directamente con el que se iba a estudiar, al disponer de partes comunes o con grandes similitudes con lo que se pretendía modelar.

En concreto, tanto el dominio deaplicaciones *software* de gestión empresarial, ERP (*Enterprise Resource Planning*), como el de aplicaciones *software* de gestión de contenidos *web*, CMS (*Content Management Solution*), fueron de gran utilidad para el modelado del dominio de análisis de la línea de productos *software* propuesta.

A continuación, se detallan las diferentes fuentes de información utilizadas[17]:

[17] El orden mostrado no se corresponde con el seguido. De hecho, la búsqueda y refinamiento de la información se hizo de forma simultánea en las diferentes fuentes una vez se obtuvo una primera información de "campo"

- **Investigación de "campo"**: Principal fuente de información y, posiblemente, la más fidedigna dada su relación directa con el dominio objeto de estudio.

  La asociación a la que pertenece el autor de este Proyecto Fin de Carrera fue el contexto de partida para la elaboración de una primera lista de funcionalidades, requisitos y necesidades. A partir de esta primera lista, se contrastó con otras asociaciones, no relacionadas con la primera, mediante entrevistas con personal involucrado en la asociación y con conocimiento del funcionamiento interno de la misma.

  Posteriormente, se elaboró un conjunto de características comunes a todas las asociaciones, otro conjunto de características, opcionales o alternativas, que no eran compartidas por todas (en este último caso, solía tratarse de características relacionadas directamente con el fin de la asociación) y un último conjunto de características sin un grupo definido a la espera de un análisis más exhaustivo.

- **Investigación de aplicaciones existentes**: Desgraciadamente no se encontró ninguna aplicación que permitiese la gestión de una asociación tal y como se planteó en un principio y se ha definido anteriormente, aunque sí que se pudo realizar un estudio de una aplicación para una asociación concreta, "Aplicación web para la Asociación Casa de Beneficencia de Valladolid" [36], que permitió reforzar los puntos comunes para la línea de productos *software* y encontrar algún que otro punto variable e incluso alguna excepción a lo que en la investigación de "campo" se clasificó como punto común.

  Sin embargo, también se estudiaron aplicaciones muy relacionadas, en tanto que su orientación era hacia organizaciones no gubernamentales como "GONG: Herramienta de Gestion para ONG's" [4].

- **Investigación de aplicaciones en dominios relacionados**: El objetivo de esta fuente de información no fue tanto la obtención de funcionalidades, requisitos o características sino el estudio del tratamiento de información por parte de aplicaciones *software* orientadas a dicha tarea.

  Los grupos de aplicaciones estudiados fueron:

  - Gestores de contenidos (CMS): Este subgrupo tuvo gran relevancia en este apartado ya que aunque el principal uso de estas aplicaciones está en la gestión de un portal corporativo con información, durante los últimos años han sido modificados y adaptados para la realización de tareas muy diversas.
    El CMS *Joomla!* fue uno de los referentes en este subgrupo.
  - Gestores empresariales (ERP): Los gestores empresariales se estudiaron para conocer el funcionamiento automatizado de los procesos de gestión de una empresa, en tanto que una asociación comparte, de forma muy simplificada, alguno de esos procesos.
    El ERP *OpenBravo* fue de gran utilidad para este subgrupo.
  - Gestores de contabilidad: Si bien este subgrupo se podría integrar en el subgrupo anterior, se optó por obviar el proceso de contabilidad empresarial por su gran complejidad para buscar aplicaciones, como *GNUCash*, que implementasen una gestión de la contabilidad más próxima a las necesidades de una asociación.

- **Consulta de literatura**: Entre la literatura consultada, con el objetivo de obtener información teórica así como metodologías de desarrollo de *software*, cabe destacar la consulta de documentación sobre *frameworks*, líneas de productos y factorías de *software*.

- **Investigación *web***: Esta fuente de información tuvo una orientación más práctica que teórica, ya que lo se pretendía era comprobar el funcionamiento *in-situ* de aplicaciones de gestión para experimentar sobre los procesos que había que llevar a cabo para la realización de diferentes tareas.

  Como ya se ha indicado anteriormente, no se encontraron aplicaciones relacionadas directamente con el dominio de análisis, por lo que en este caso se optó por experimentar sobre los gestores de contenidos *Joomla!* y *phpBB* (éste último se trata de un foro), el gestor empresarial *OpenBravo* y sobre una aplicación de comercio electrónico, como la dispuesta por la empresa *Amazon.com*.

2.4.2. Desarrollo dirigido por modelos

Durante toda la fase de análisis del dominio de la línea de productos *software* estuvo presente el desarrollo dirigido por modelos, MDSPL, ya que, como se ha explicado en el apartado 2.2, los modelos, y concretamente, los modelos de características, son el elemento más óptimo para representar, organizar y gestionar la variabilidad inherente a toda línea de productos *software*.

Una vez acotado el dominio, y obtenida toda la información necesaria, se procedió a describir el caso de estudio de una línea de productos *software* para asociaciones utilizando el modelo de características como modelo de decisión de la línea de productos *software*.

Para desarrollar todos los diagramas del modelo de características se ha utilizado la herramienta de modelado *fmp*, un *plug-in* para el entorno de desarrollo *Eclipse*, diseñado y programado por el equipo liderado por el investigador Krzysztof Czarnecki [1].

Se ha optado por la solución desarrollada en la Universidad de Waterloo, ya que otras herramientas, como *AmiEddi* (el primer editor que utilizó la notación extendida *Czarnecki-Eisenecker*), *Captain-Feature*, *ReqiLine* o *Pure::Variants*, no implementaban cardinalidades o no obligaban a realizar una configuración de "arriba-abajo" (*top-down*) o no soportaban la clonación de características o bien estaban más orientadas al modelado de relaciones que características.

La herramienta *fmp* implementa un meta-modelo que define la estructura del modelo de características, junto con los diferentes tipos de características. También permite establecer cardinalidades, así como algunas propiedades básicas de cada característica: nombre, identificador y descripción. Además, la herramienta permite acceder y modificar el meta-modelo concreto de un determinado modelo de características para definir nuevos atributos de las características, si así se desea.

Con esta herramienta es posible realizar la configuración del modelo de características, pues provee de una interfaz para ello basada en la réplica del modelo de características sobre la que es posible decidir uno de los tres estados posibles para cada característica opcional, vacío (el estado de la característica se mantiene indefinido, y por lo tanto no se ha tomado una decisión acerca de la inclusión o exclusión en el producto concreto), marcado (la característica ha sido seleccionada y será incluida en el producto concreto) y eliminado (la característica no será seleccionada y por lo tanto no será incluida en el producto concreto). La configuración e incluso el propio diagrama se pueden exportar en un fichero XML (*eXtensible Markup Language*) para su posterior tratamiento por la herramienta, que se encarga del ensamblado para la generación del producto concreto o para realizar una edición y configuración más concisa en caso de que se haya optado por una configuración parcial.

Por último, la herramienta también permite establecer restricciones entre características, que se pro-

pagan de forma automática, por lo que es posible seleccionar o eliminar características automáticamente en función de las decisiones tomadas en otros puntos del árbol, de acuerdo a las restricciones que se hayan implementado mediante el lenguaje *XPath* en su versión 2.0.

En el cuadro 2.1 se puede ver los símbolos, acompañados de su correspondiente descripción, más utilizados en el diseño de cualquier modelo de características con la herramienta *fmp*.

SÍMBOLO	SIGNIFICADO
F	Característica unitaria F con cardinalidad [1..1] (característica obligatoria)
G	Característica unitaria G con cardinalidad [0..1] (característica opcional)
[m..n] H	Característica unitaria H con cardinalidad [m..n] / m > 0 ∧ n > 1 (característica obligatoria clonable)
[0..m] I	Característica unitaria I con cardinalidad [0..m] / m > 1 (característica opcional clonable)
J ('valor':T)	Característica J con un atributo de tipo T y el valor asignado valor
	Grupo de características con cardinalidad <1-k>, donde k es el tamaño del grupo (grupo de características OR)
	Grupo de características con cardinalidad <1-1> (grupo de características XOR)
<i-j>	Grupo de características con cardinalidad <i-j>
K	Característica de grupo K con cardinalidad de grupo [1..1]
L	Característica de grupo L con cardinalidad de grupo [0..1]

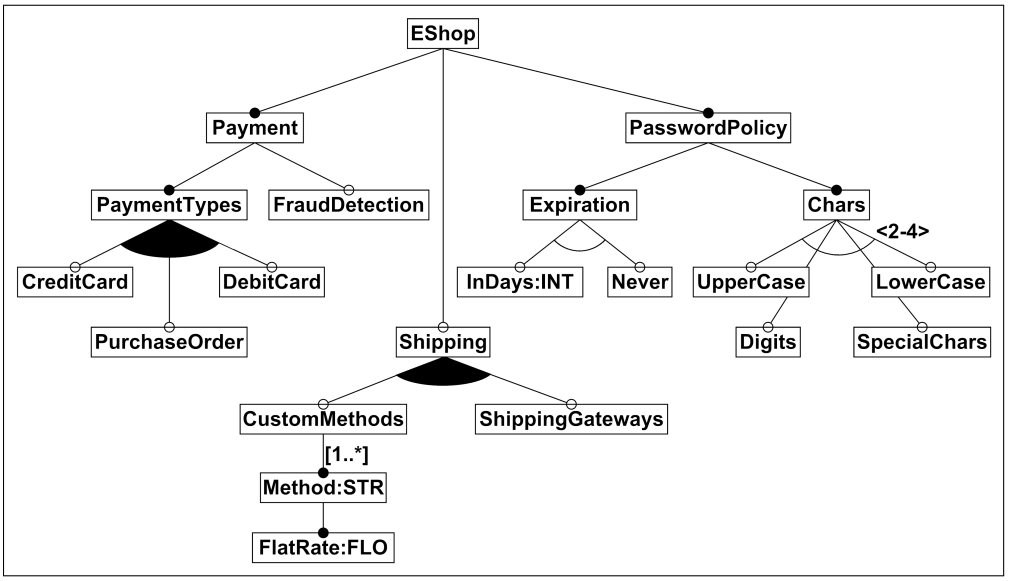
Cuadro 2.1: Notación utilizada por el *plug-in* de *Eclipse*, *fmp*, en los modelos de características

La versión del *plug-in* "*Feature Modeling Plug-in*" (*fmp*) utilizado para todos los diagramas mostrados en esta memoria, ha sido la 0.7.0[18].

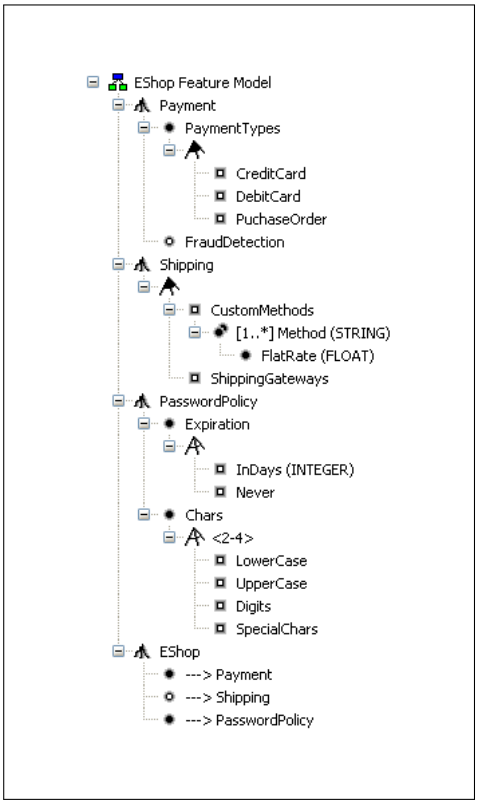
Esta nueva versión aporta grandes mejoras en cuanto a la gestión de las restricciones respecto a su precedecesora, la versión 0.6.6, aunque ya no usa *XPath* para la definición de las mismas; en su lugar, hace uso de un árbol jerárquico para visualizar las restricciones al estilo del modelo de características, y asistentes para la definición de las restricciones entre características. Es importante destacar que la comprobación de incoherencias y errores en las restricciones es más sencilla con la versión 0.7.0.

A continuación se muestra un ejemplo de la interfaz del *plug-in*, tanto de un diagrama del modelo de características como de la configuración de dicho modelo creados con la herramienta, en la ilustración de la figura 2.19. También se muestra el modelo de características en notación estándar en la subfigura para que se vean las correspondiencias de los diferentes elementos.

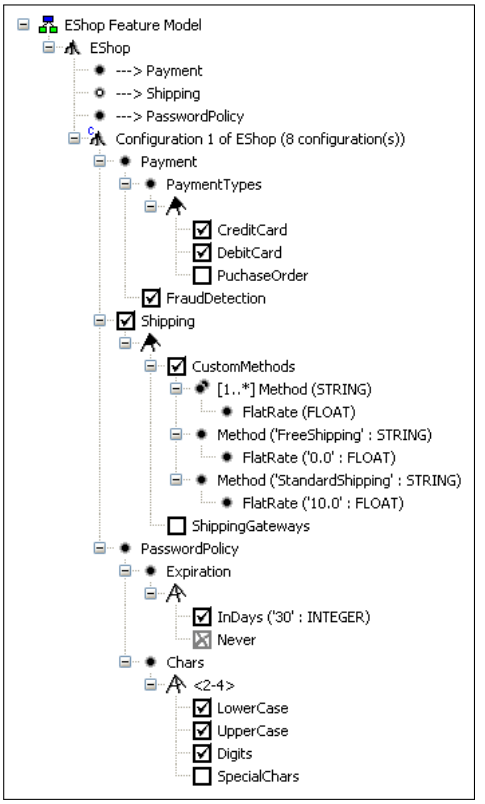
[18]Tanto el *plug-in* (que además está bajo la licencia *open source*, *Eclipse Public License* 1.0) como la documentación y algunos ejemplos y artículos están disponibles para su libre descarga en la dirección *web*: http://gsd.uwaterloo.ca/projects/fmp-plugin/



(a) Modelo de características en notación estándar



(b) Modelo de características en notación del plug-in fmp



(c) Configuración del modelo del modelo de características

Figura 2.19: Ejemplo del modelo de características y su configuración con *fmp*

En el apéndice B se muestra el diagrama de clases del meta-modelo utilizado para la generación de los modelos de características por la herramienta *fmp*.

En el apéndice C se puede encontrar una breve guía sobre la instalación del *plug-in*en el entorno de desarrollo *Eclipse*, así como su uso.

Capítulo 3
Caso de estudio de una línea de productos para asociaciones

"Tan a destiempo llega el que va demasiado deprisa como el que se retrasa demasiado"

William Shakespeare

3.1. Análisis del modelo de características

A lo largo de este apartado se describirán, con gran detalle, todas las características y la funcionalidad prevista en la línea de productos *software* para asociaciones, utilizando el modelo de decisión presentado anteriormente, el modelo de características.

Dado el número de características contemplado (más de 280), e incluso con la posibilidad de añadir bastantes más[1], se ha dividido el modelo de características en ramas, representando cada una de ellas un "gran" concepto dentro de la línea, es decir, una parte importante del quehacer diario de una asociación que podría necesitarse (no debe olvidarse que se está tratando una línea de productos, cuyo principal objetivo es la gestión de la variabilidad con lo que no todas las asociaciones van a necesitar o van a hacer uso de toda la funcionalidad descrita) en un producto para la gestión de la asociación.

El análisis del modelo de características que se presenta a continuación es la guía que debe acompañar a todos los involucrados en el proceso de desarrollo de la línea de productos *software* para asociaciones, ya que como se ha mencionado en el capítulo 2, el modelo de decisión es una de las piezas clave en toda línea de productos tanto por la representación de toda la funcionalidad de la línea en el modelo como por ser el "manual" de comprensión de la línea.

Una interpretación equivocada de una característica del modelo, una configuración de un conjunto de características en un instante no adecuado,... lleva a la obtención de una aplicación que tendrá fallos,

[1] Es posible que en sucesivas lecturas de las descripciones de este apartado se puedan extraer nuevas características, aunque no se debe olvidar que el modelo de características representa conceptos, funcionalidades y características, y el nivel de detalle del mismo no debería ser atómico en tanto que estaría invadiendo el dominio de diseño y, sin embargo, el modelo de características pertenece al dominio de análisis aunque determine fases posteriores, incluida la codificación, compilación e incluso la ejecución

y que se hubiesen minimizado si se hubiese realizado una descripción adecuada del modelo de características. Por ello es muy importante describir todas las características del modelo de características y las funcionalidades que agrupan para tomar las decisiones adecuadas en el momento oportuno y para mejorar y corregir el modelo de características.

Algunas de las características de cada una de las ramas, descritas en subapartados independientes, pueden requerir unas restricciones que se han descrito textualmente al final de cada subapartado en un cuadro. En el cuadro, la nomenclatura utilizada para la escritura de las características ha sido la eliminación de espacios entre palabras y la escritura en mayúsculas de la primera letra de cada palabra a unir.

Estas restricciones son muy importantes, ya que si no se tienen en cuenta, se podrían producir inconsistencias o incoherencias a la hora de la toma de decisiones, problemas en la configuración (especialmente cuando se haga con algún programa con el objetivo de automatizar esta fase) y por supuesto, afectará negativamente en sucesivas fases del desarrollo de la línea de productos *software*.

Es decir, cualquier configuración realizada sobre el modelo de características debe validarse contra estas restricciones. Por ejemplo, el *plug-in* de *Eclipse*, *fmp*, en su version 0.7.0, permite comprobar automáticamente la validez de las restricciones sobre características obligatorias y optativas, características de grupo de tipo OR y XOR; sin embargo, no es posible comprobar restricciones sobre tipos de datos concretos de características (INTEGER, FLOAT, STRING, FEATURE,...), referencias a otras características (no olvidando que las referencias sólo pueden ser a características raíces del diagrama) y características clonables.

Esto lleva a que si se hace uso en cualquier diagrama de los elementos que no pueden ser verificados por el *plug-in*, la comprobación deberá ser manual, con el consiguiente aumento de tiempo, posibilidad de error e incluso omisión de algunas restricciones.

Los cuadros de restricciones que hay al final de la descripción de cada rama describen las restricciones existentes entre características de la propia rama y características de la rama descrita y otras ramas, a excepción de la rama de la característica `Portal web`. Esto se debe a la división gráfica realizada del modelo de características, utilizando referencias para facilitar la descripción. Por ello, las restricciones también deben ser interpretadas de forma jerárquica, partiendo de la rama de la característica principal, `Portal web`, y siguiendo las referencias a otras ramas y examinando las restricciones que estas planteen.

No ha de olvidarse que la selección de una característica opcional en cualquier punto del modelo de características conlleva la selección de todas las características opcionales “padre” hasta alcanzar una característica que sea obligatoria.

En los cuadros de restricciones se observa que algunas restricciones tienen en el lado izquierdo el símbolo $\gg$. Este símbolo indica que la restricción se presenta entre dos características pertenecientes a la misma rama.

En el cuadro 3.1 se muestra la notación utilizada para la representación de las relaciones de restricción entre dos características de forma abreviada.

RELACIÓN	SIGNIFICADO
A $\rightarrow$ B	A recomienda a B
A $\Rightarrow$ B	A requiere[2] de B
A $\not\Rightarrow$ B	A es independiente de B
A $\rightleftarrows$ B	A excluye a B

Cuadro 3.1: Notación utilizada en las relaciones de restricción entre dos características

En la ilustración de la figura 3.1, se muestra el esquema principal del modelo de características. La característica raíz (*root*) es la denominada como `Portal web`, ya que, aunque no se ha comentado, los productos *software* que generará la línea de productos tendrán como entorno de funcionamiento a la *web*. El resto de características correspondenderán a lo que antes se ha denominado como "gran" concepto (uno por cada característica) o "rama".

Figura 3.1: Esquema general de características de la línea de productos para asociaciones

3.1.1. Portal *web*

La característica `Portal web` es la más importante de todo el modelo de características de la línea de productos *software* para asociaciones, ya que, como se ha indicado al comienzo del apartado, esta característica es la que describe toda la funcionalidad de la línea, de modo que el resto de características descritas en este apartado descienden[3] de alguna de las ramas descritas en esta característica.

El portal *web* reune toda la funcionalidad posible que se puede realizar con la aplicación y es una primera base para el diseño de la aplicación en tanto que define las características que como mínimo debe haber en toda aplicación, así como las restricciones que existen entre características opcionales

[2] "Requiere" es equivalente a "incluye" ya que a efectos prácticos en la selección de la característica A, es necesario seleccionar la característica B para poder asegurar la consistencia de la funcionalidad de la característica A

[3] Cuando lea en el texto característica referenciada deberá sustituir la referencia por la rama de la característica correspondiente junto con las subcaracterísticas que desciendan o dependan de ella

de la misma rama o de ramas diferentes, en caso de seleccionar alguna de estas características en la configuración del producto concreto.

La ilustración de la figura 3.2 muestra la jerarquía de características de la característica (aunque siendo más exactos habría que denominarla como concepto) `Portal web`.

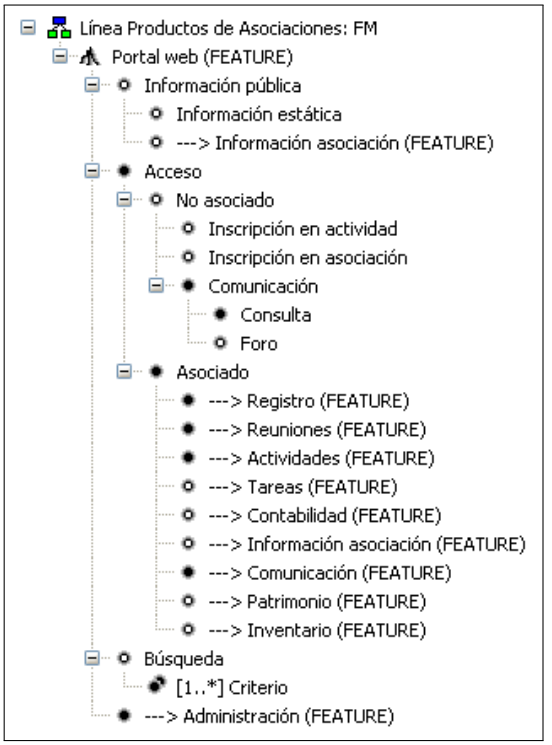


Figura 3.2: Esquema de la característica `Portal web`

3.1.1.1. Información pública

Aunque el objetivo del portal *web* sea el uso de las diferentes herramientas, que se dipongan, por parte de los asociados para todo lo relacionado con la asociación, también se podrá aprovechar para ofrecer dos tipos de información de ámbito general (para cualquier público, tanto asociados como no asociados), aunque habrá que seleccionar dicha funcionalidad en la configuración del producto concreto a obtener:

- Información sobre la propia asociación, modelada en la subcaracterística opcional `Información estática`, como por ejemplo la historia, los estatutos, las instalaciones, los órganos de gobierno, las actividades que se suelen realizar,... y que cambia con poca frecuencia dado que es informativa (por ejemplo, puede haber información de las actividades pasadas que se han organizado pero no indica que se vayan a organizar de nuevo).
- Noticias y avisos de la asociación, modelado en la referencia a la subcaracterística opcional `Información asociación` (3.1.7). Por ejemplo, se podrán mostrar noticias de una nueva

composición de la Junta Directiva de la asociación, o de un acuerdo con una institución para la organización futura de una actividad concreta, o incluso la organización de una actividad a la que podrá asistir cualquier persona. Y como ejemplo de avisos, puede ser el fin de plazo de la inscripción en una actividad, un cambio de última hora en una actividad, o el cambio de horario de atención al público de la asociación.

Toda la información proporcionada en el portal *web* sobre noticias o avisos no requerirá de registro alguno, ya que sólo se mostrará aquella cuyo destinatario sea la subcaracterística `No asociado` (3.1.7.1), y estará siempre disponibles, a excepción de los avisos, que dejarán de mostrarse en la fecha indicada en la subcaracterística `Fecha caducidad` (3.1.7.1).

3.1.1.2. Acceso

El objetivo principal del portal *web* es proporcionar un acceso centralizado a todas las herramientas de gestión de la asociación, así como a las funcionalidades de interés para los asociados.

Es imprescindible establecer un acceso que diferencie a los asociados entre sí, ya que no todos tendrán acceso a las mismas herramientas ni tampoco lo tendrán acceso al mismo nivel de funcionalidad que ofrezca por cada una de las herramientas. También es posible que se desee proporcionar un acceso limitado a los no asociados, debido a las necesidades de la asociación para la organización de actividades o para la inscripción en la asociación.

Los usuarios de cualquier producto obtenido con la línea de productos *software* van a ser los asociados, y así se ha modelado en la subcaracterística obligatoria `Asociado`. Opcionalmente, la aplicación podrá ofrecer a los no asociados información propia de la asociación, para la que se haya establecido su visibilidad por parte de los no asociados; esta otra parte se ha modelado en la subcaracterística opcional `No asociado`.

Los asociados van a disponer, en cualquier producto, de las siguientes funcionalidades, representadas como características obligatorias[4]:

- **Registro** (3.1.2): Funcionalidad necesaria para la creación y salvaguarda de los usuarios de la aplicación, independientemente de su condición de asociados o no, junto con algunas preferencias personales (lo que comúnmente se denominará como "perfil".

  El registro también incluye, aunque a modo ilustrativo en tanto que se representaría mejor mediante restricciones, las condiciones bajo las cuales es obligatorio dicho registro.

- **Reuniones** (3.1.3): Funcionalidad necesaria para simular la gestión tradicional de las reuniones en una asociación mediante el "libro de actas".

  También permite buscar un acta concreta así como realizar rectificaciones y visados (aunque el Presidente de la Asociación será el único que podrá visar) a las actas.

- **Actividades** (3.1.4): Gestión de las actividades, tanto la organización[5] de las propias actividades como la inscripción en las mismas (incluyendo el pago en caso de que haya actividades

[4] Como ya se indicó al comienzo del apartado, cada una de esas funcionalidades se ha representado mediante una referencia en el plugin *fmp* de *Eclipse* para facilitar tanto la comprensión como la descripción de cada una de ellas. Cada uno de los subapartados que siguen a éste, describen con gran detalle la expansión de cada una de las referencias del diagrama de características de `Portal web` mostrando su diagrama de características correspondiente

[5] Esta característica incluye entre sus subcaracterísticas, una referencia a la rama encabezada por la característica `Tareas` (3.1.5)

con un coste para el participante) por parte de los participantes que se hayan definido en la actividad.

También se podrá llevar un control de asistencias, generar diplomas y hacer encuestas.

- **Comunicación** (3.1.8): La aplicación ha de proporcionar un sistema de comunicación interno básico, y eso se representa y modela en esta rama. También será posible, si se selecciona en la configuración, la comunicación por correo electrónico.

  Otra de las funcionalidades proporcionadas por esta rama son las consultas a la asociación o a algún estamento dentro de la asociación, sin necesidad de estar registrado en la aplicación, es decir, sin tener por qué ser usuario.

  Si se considera necesario un foro, esta rama se encargará de ello.

Los asociados también tendrán acceso a las siguientes funcionalidades, representadas como características opcionales, siempre que se seleccionen en la configuración del producto concreto que se desee obtener y las restricciones del modelo de características validen:

- **Tareas** (3.1.5): En una asociación es muy común el reparto de trabajo entre asociados, lo cual facilita la organización de una actividad, reunión,... sin embargo, sin un control de ese reparto de trabajo, esa facilidad inicial se puede volver en contra. Esta rama se encarga de facilitar la organización y gestión de las diferentes tareas, asociándolas a una actividad, comisión, proyecto,...

  Es posible llevar un control del estado de la tarea e incluso asociar un fichero a las tareas. También se puede disponer de un "bloc de notas".

  Esta característica es opcional porque no todas las asociaciones pueden tener tanta complejidad en la organización interna, y por lo tanto, no necesiten ayuda externa.

- **Contabilidad** (3.1.6): Al igual que con el libro de actas, la contabilidad pretende simular el "libro de cuentas" donde han de realizarse todas las anotaciones de "entradas" y "salidas" de dinero. Aprovechando la disponibilidad de un sistema informático, se proporciona la posibilidad de generar resúmenes, incluso enviarlos.

  Es posible que una asociación ofrezca una serie de servicios por los que facture una cantidad (por ejemplo, una asociación podría editar una revista y vender espacios publicitarios en la misma). De la emisión de facturas o recibos se encargará esta característica.

  También puede ocurrir que la asociación tenga a personal en nómina al que deba pagarle un sueldo, y, como en la facturación, la rama de la contabilidad se encarga de ello.

  Esta característica es opcional porque es posible que la contabilidad de la asociación sea tan simple no se necesite.

- **Información asociación** (3.1.7): Toda información genérica, o de ámbito general que la asociación desee proporcionar tanto a los asociados como a los no asociados será gestionada en esta rama.

  La información se proporcionará en forma de noticias, anuncios o avisos, no distinguiéndose entre usuarios individuales aunque sí se podrá distinguir entre grupos de usuarios con el mismo nivel de acceso.

  Esta característica es opcional porque puede ocurrir que una asociación no necesite realizar

ningún aviso, anuncio o noticia de esta forma, o porque quiera hacerlo mediante un mensaje personal o por correo electrónico.

- **Inventario** (3.1.9): Funcionalidad necesaria para el control y gestión del inventario de bienes tangibles de la asociación. Se podrán generar listados de los bienes de acuerdo a unos criterios que se establezcan.

  Esta característica es opcional porque puede no ser necesario el disponer de un inventario (o bien que la asociación no quiera).

- **Patrimonio** (3.1.10): Funcionalidad necesaria para la gestión del patrimonio (bienes inmuebles) de la asociación, con el objetivo de llevar también un control de los gastos asociados a un determinado bien inmueble.

  Esta característica es opcional porque puede no ser necesario llevar un control del patrimonio, bien porque no tenga patrimonio, o bien porque sea sólo una sede, o bien porque resulte más sencillo o menos laborioso hacerlo de otra forma.

La subcaracterística opcional `No asociado` reúne el conjunto de funcionalidades a los que tendrá acceso un usuario que no pertenezca a la asociación, pero que se haya registrado en la aplicación. Se puede observar que la funcionalidad que tiene es mínima, ya que al no pertenecer a la asociación no tiene porqué ver información que no le corresponde.

Todo usuario no asociado puede comunicarse mediante mensajes personales en forma de consultas; opcionalmente, si el administrador del sistema lo considera, podrá tener acceso al foro o algún subforo con la posibilidad de publicar mensajes.

Si los no asociados pueden participar en alguna actividad, entonces habrá que seleccionar en la configuración la subcaracterística opcional `Inscripción en actividad`. Sólo podrán inscribirse en aquellas actividades en las que entre los destinatarios estén los no asociados.

Del mismo modo, se puede permitir el asociacionismo a través de la aplicación, en lugar de tener que desplazarse a la sede de la asociación. No obstante, es probable, que aunque se permita asociarse de esta forma, la persona interesada tenga que formalizar la inscripción de alguna forma; por ejemplo en el caso de que tenga que demostrar que cumple una serie de requisitos que la asociación exija para poder asociarse.

3.1.1.3. Búsqueda

La subcaracterística opcional `Búsqueda` permite realizar búsquedas de información de acuerdo a un criterio dado (por ejemplo, una palabra o frase que forme parte de un texto o de un título) por todo el portal.

El alcance de las búsquedas dependerá del nivel de acceso del usuario que realice la búsqueda. Por ejemplo, un usuario no asociado que realice una búsqueda desde el portal *web* con la palabra clave "acta" no obtendrá ningún listado de actas.

3.1.1.4. Administración

La subcaracterística `Administración`, que además es una referencia a la rama de la característica `Administración`(3.1.11), proporciona el conjunto de herramientas necesarias para el mantenimiento

y control de la aplicación de gestión de la asociación, entre ellas, el control de acceso de los usuarios, el repositorio de ficheros, el registro de actividad y las copias de seguridad. También permite realizar una configuración general de la misma (por ejemplo, el nombre del sitio, plantillas gráficas,...).

3.1.1.5. Restricciones

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN	**CONTEXTO**	
`InformacionAsociacion` (3.1.1.1)	◄►	`NoAsociado` ∧ ¬`Asociado` (3.1.7.1)
A ⇒ B	La información de la parte pública sólo puede aquella cuyo destinatario sea `NoAsociado`	
B ⇏ A	La selección del destinatario de una información es independiente de su ubicación en el portal *web*	
`InformacionAsociacion` (3.1.1.1)	◄►	`UnaDireccion` ∧ ¬`GrupoDirecciones` (3.1.7.4)
A → B	Es recomendable que la información de la parte pública se pueda enviar a una única dirección de correo electrónico (pero no a un grupo de direcciones)	
B ⇏ A	La posibilidad de envío de una información es independiente de su ubicación en el portal *web*	
`InformacionAsociacion` (3.1.1.1)	◄►	`Busqueda` (3.1.7.5)
A → B	Es recomendable que se puedan realizar búsquedas en el portal *web* de la información pública	
B ⇏ A	La posibilidad de buscar una información es independiente de su ubicación en el portal *web*	
`NoAsociado` (3.1.1.2)	◄►	`NoAsociados` (3.1.11.3)
A → B	El acceso de no asociados a la aplicación requiere que los administradores lo hayan habilitado en la interfaz de administración	
B ⇏ A	La selección de la característica `No asociados` es independiente de que el portal web disponga de un acceso para los no asociados	
`InscripcionActividad` (3.1.1.2)	◄►	`InscripcionActividad` (3.1.2.2)
A → B	Es recomendable que el usuario no asociado que se inscriba en una actividad se registre en la aplicación	
B ⇏ A	La obligación de registro en la aplicación para una actividad es independiente del tipo de usuario que lo haga	
`InscripcionAsociacion` (3.1.1.2)	◄►	`MembresiaAsociacion` (3.1.2.2)
A ⇒ B	Un usuario con acceso de "no asociado" ha de poder disponer los medios para asociarse si se le permite hacerlo por sí solo	
B → A	Es recomendable que se proporcione la funcionalidad de registro desde la aplicación a los no asociados si es obligatorio el registro para asociarse mediante la aplicación	

Continúa en la página siguiente

Viene de la página anterior

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN **CONTEXTO**		
`Foro` (3.1.1.2)	◄►	`NoAsociadosOtros` (3.1.8.3)
A ⇒ B	Se ha de proporcionar acceso a los foros de no asociados si se les permite acceder al foro	
B ⇏ A	El acceso a no asociados en el foro es independiente de que realmente lleguen a poder acceder[6]	

Cuadro 3.2: Restricciones adicionales de la característica `Portal web`

3.1.2. Registro

La característica `Registro` describe toda la funcionalidad necesaria para el almacenamiento permanente de cierta información de los usuarios de la aplicación, de modo que en sucesivos accesos a la aplicación no tengan que proporcionar esa información porque ya esté almacenada. Sólo tendrán que acreditar su identidad ante la aplicación para garantizar que son los "dueños" de la cuenta de usuario que quieren utilizar.

Esta característica también debe definir los criterios por los que será necesario realizar el registro en la aplicación.

La ilustración de la figura 3.3 muestra la caracterización del conjunto de características encabezadas por la característica `Registro`.

3.1.2.1. Información de usuario

La característica obligatoria `Información de usuario` representa la información que se recogerá de un usuario para almacenarla posteriormente en la aplicación. Dependiendo de la funcionalidad que se permita que tenga un usuario, se pedirá más o menos información. Lo habitual en estos casos, es recoger la información mediante un formulario, que no se ha modelado mediante una característica porque el método de recogida de datos es responsabilidad del diseño e implementación y no del modelo de decisión.

Como mínimo se han de almacenar las credenciales de acceso, y proporcionar mensajería en la aplicación mediante mensajes personales. Con los credenciales de acceso, el usuario podrá utilizar la aplicación hasta donde el administrador del sistema haya definido de acuerdo a las políticas de seguridad de la asociación y podrá intercambiar mensajes con el resto de usuarios.

Si se considera necesario, se podrán almacenar otra serie de datos personales, representados en la subcaracterística `Datos personales`, y si el usuario a registrar va a ser un nuevo asociado, es obligatorio el almacenamiento de información requerida por la asociación.

[6] No tiene mucho sentido proporcionar la funcionalidad de acceso a los no asociados y luego no permitirles usarlo, sin embargo es válido en el modelo de características

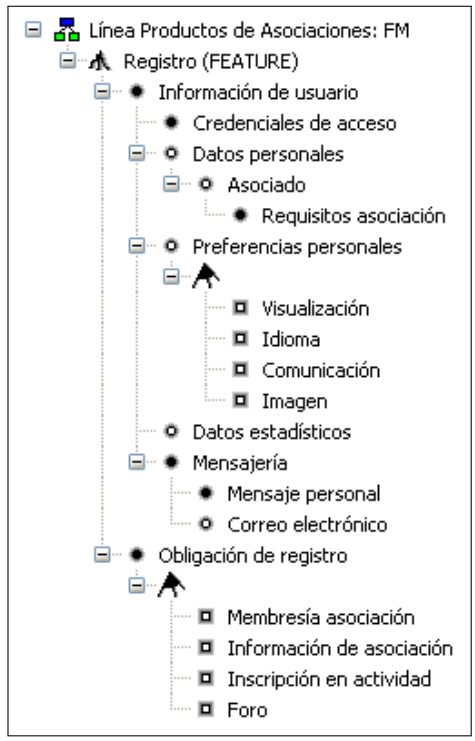


Figura 3.3: Esquema de la característica `Registro`

La elección de la característica opcional `Preferencias personales` en tiempo de compilación, lleva necesariamente a elegir qué preferencias podrá seleccionar y modificar el usuario en su perfil de la aplicación. A destacar, de acuerdo al resto de características del modelo sujeto de análisis, están:

- `Visualización`: elección un estilo para la interfaz gráfico de un conjunto de estilos instalados por el administrador
- `Idioma`: elección de la configuración del idioma de la interfaz de la aplicación (siempre que el administrador lo haya configurado así, en caso contrario no se podrá escoger ningún idioma)
- `Imagen`: posibilidad de incrustar una imagen de un tamaño determinado en el perfil de modo que se muestre al visualizar la parte pública del perfil del usuario
- `Comunicación`: información asociada al método de comunicación por defecto en la aplicación (por ejemplo, la dirección de correo electrónico)

Puede ser interesante para la elaboración de estadísticas, el pedir datos demográficos o de otra índole de interés para la asociación con el objeto de adaptarse y conocer el perfil de los usuarios; para ello, basta con seleccionar la subcaracterística opcional `Datos estadísticos` que permite disponer de esta funcionalidad.

Una subcaracterística obligatoria bastante importante de la característica `Información de usuario` es `Mensajería`, la cual aportará la funcionalidad necesaria para poder enviar y recibir mensajes

en el entorno de la aplicación. El funcionamiento de esta característica será disponer de un buzón para cada usuario, de modo que cuando se genere y envíe un mensaje, éste se guarde en el buzón del destintario (logicamente sólo el destinatario podrá acceder a su buzón y por tanto leer los mensajes que haya recibido).

Opcionalmente, en tiempo de desarrollo, se podrá activar el envío por correo electrónico si se necesita para el producto concreto que se esté configurando.

3.1.2.2. Obligación de registro

Dado que la aplicación tiene partes de acceso público (y dentro del acceso público, secciones que pueden estar activas si se ha seleccionado la característica que gestiona la funcionalidad en la configuración) y partes de acceso privado, es decir, secciones exclusivamente para asociados (y dentro de estas, secciones para comisiones o departamentos y la Junta Directiva), se debe indicar en la configuración en tiempo de compilación qué opciones van a obligar a realizar un registro en la aplicación para poder acceder a ellas.

En tiempo de ejecución se podrá configurar, de entre las disponibles, las opciones que deberán estar activas.

Las características agrupadas en la característica de grupo `Obligación de registro` de tipo OR son las siguientes:

- `Membresía asociación`: Todo usuario de la parte pública de la asociación que desee asociarse, podrá hacerlo a través de la aplicación (probablemente mediante un formulario), y esta característica guiará al usuario a través de todo el proceso.

  El registro como asociado no implicará el acceso automático a las secciones a las que los ya asociados tengan acceso; quedará pendiente de revisión de la información proporcionada, e incluso se podrá pedir información adicional mediante un mensaje personal.

- `Información de asociación`: Ciertas noticias o anuncios/avisos podrán requerir de un registro en la aplicación para poder tener un completo acceso al resto de la información de la noticia o anuncio/aviso.

  Por ejemplo, un aviso que tenga ficheros adjuntos podría incluir el registro en la aplicación para evitar abusos en la descarga del fichero, o simplemente para poder realizar un seguimiento de los usuarios que acceden a dicho fichero.

- `Inscripción en actividad`: La inscripción en una actividad puede requerir de un registro previo, en tanto que la asociación requiera más información que un nombre y una forma de contacto.

  Por ejemplo, cuando una actividad tenga un coste, será necesario tener información más detallada de la persona que va a participar en esa actividad por cuestiones legales, y probablemente también por cuestiones administrativas.

- `Foro`: Aunque es posible que se permita el acceso al foro para la lectura de las discusiones que se planteen, si un usuario desea responder, deberá registrarse para evitar dar más trabajo del necesario a los moderadores del foro con mensajes publicitarios o no solicitados, y también para asociar las respuestas a un usuario concreto y no a uno anónimo.

3.1.2.3. Restricciones

	CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
	RELACIÓN	**CONTEXTO**	
≥	`MembresiaAsociacion` (3.1.2.2)	◄►	`Asociado` (3.1.2.1)
	A ⇒ B	Para asociarse, la persona interesada debe proporcionar una serie de datos personales (y cumplir los requisitos que la asociación haya podido establecer) que no se exigne a los usuarios registrados en la aplicación no asociados	
	B ⇏ A	El almacenamiento de datos personales y cumplimiento de los requisitos de la asociación es independiente de que una persona se pueda asociar mediante la aplicación	
	`Visualizacion` (3.1.2.1)	◄►	`PlantillaGrafica.Usuario` (3.1.11.2)
	A ⇒ B	No será posible configurar ninguna plantilla para cada usuario si no se permite en la configuración del sistema aparte de la plantilla genérica para todos los usuarios	
	B → A	Si se permite escoger a los usuarios una plantilla gráfica determinada, entonces es recomendable que dispongan de tal opción en su perfil	
	`Idioma` (3.1.2.1)	◄►	`Idioma.Usuario` (3.1.11.2)
	A ⇒ B	No será posible configurar ningún idioma para cada usuario si no se permite en la configuración del sistema aparte del idioma común para todos los usuarios	
	B → A	Si se permite escoger a los usuarios un idioma concreto, entonces es recomendable que dispongan de tal opción en su perfil	
	`Imagen` (3.1.2.1)	◄►	`Importar` (3.1.11.4)
	A ⇒ B	Para asociar una imagen a un perfil de usuario se ha de importar previamente mediante la interfaz de la aplicación	
	B ⇏ A	La selección de la característica `Importar` es independiente del lugar donde se pueda utilizar	
	`CorreoElectronico` (3.1.2.1)	◄►	`CorreoElectronico` (3.1.8.1)
	A ⇒ B	Para que los usuarios puedan enviar mensajes por correo electrónico, el administrador deberá haber habilitado la característica `CorreoElectronico`	
	B ⇏ A	El envío por correo electrónico sólo depende de la existencia de un servicio de transporte de correo electrónico, y es independiente del uso que se vaya a hacer en la aplicación	
	`Foro` (3.1.2.2)	◄►	`Foro` (3.1.8.3)
	A ⇒ B	Si se obliga a registrarse para acceder al foro es porque existe dicha funcionalidad	
	B ⇏ A	La existencia de foros en la aplicación es independiente del modo de acceso a los mismos	

Cuadro 3.3: Restricciones adicionales de la característica `Registro`

3.1.3. Reuniones

La característica `Reuniones` integra la gestión de las reuniones que se lleven a cabo en la asociación. Sólo se contemplarán las reuniones "oficiales", es decir, la aplicación almacenará y gestionará las reuniones convocadas de acuerdo a los preceptos establecidos en los Estatutos de la asociación, ya que a todos los efectos, serán las únicas reuniones que tengan validez legal, tanto para los propios asociados como para las decisiones tomadas en ellas.

Toda reunión tiene un órden del día, en el que se indican los asuntos a tratar. Los comentarios de los asuntos tratados junto con el listado de los asistentes a la reunión conforman el acta de la reunión, que es lo que la aplicación almacenará. Toda acta podrá ser consultada por cualquier asociado, indendientemente de que haya asistido o no a la reunión, o de que tuviese derecho a voto o no.

La ilustración de la figura 3.4 muestra el conjunto de características encabezadas por la característica `Reuniones`.

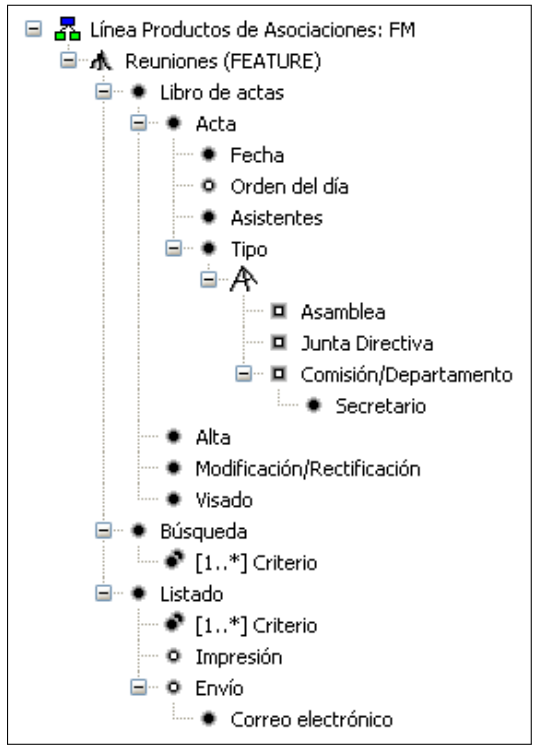


Figura 3.4: Esquema de la característica `Reuniones`

3.1.3.1. Libro de actas

La subcaracterística `Libro de actas` es el nombre simbólico que se ha dado en el diagrama de características de la característica `Reuniones` para describir la necesaria presencia, puesto que la característica es obligatoria, del equivalente en la aplicación al tradicional libro de actas en papel, que

se suele utilizar para tener prueba documental de los temas tratados y decisiones tomadas.

Por lo tanto, el libro de actas no será más que una colección de actas; cada una de esas actas tendrá una fecha, unos asistentes, un tipo de reunión dependiendo del órgano que se reúna. Ese órgano podrá estar compuesto por todos los miembros que conforman la asociación (Asamblea), un grupo de asociados nombrados para dirigir los asuntos de la asociación (Junta Directiva), o un grupo de asociados encargados de llevar a cabo uno o varios temas más concretos (Comisión/Departamento), y los temas tratados junto con los resultados de las votaciones realizadas, si el tema tratado requería tomar alguna decisión.

Es posible que también se desee almacenar el orden del día, representado en la subcaracterística opcional `Orden del día`, si bien no es necesario, ya que el propio acta incluye de forma implícita el orden del día al describir los temas tratados.

Puesto que el libro de actas no va a existir como tal (por escrito), se ha de traspasar a la aplicación todo el proceso de elaboración de un acta, e incluso la enmendación. Consecuentemente, todo acta deberá ser dado de alta, y podrá ser enmendado mediante la subcaracterística obligatoria `Modificación/Rectificación`.

En un libro de actas, todo acta está firmado por el secretario de la asociación (o persona en la que éste delegue) y posteriormente es visado por el presidente quien da el visto bueno con su firma; de esta forma adquiere validez ante cualquier asociado y entidad que pudiese requerir prueba documental de un tema tratado o una decisión tomada.

En la aplicación, este proceso de visado se representa mediante la subcaracterística `Visado`, y cuya implementación[7] debe garantizar tanto la autenticidad de las firmas involucradas como la integridad del acta.

En este punto es importante destacar la subcaracterística obligatoria `Secretario`, hija de la característica `Comisión/Departamento`, y que representa al Secretario de la comisión o departamento que se reúna y oficialice esa reunión en forma de acta en la aplicación. El Secretario de esta Comisión o Departamento es una persona designada entre los miembros de la comisión o departamento, que se encarga de tomar nota de las decisiones llevadas a cabo. Este tipo de reuniones no requerirán[8] visado por parte del presidente de la asociación.

3.1.3.2. Búsqueda y Listado

Ya se ha indicado que todo asociado tiene derecho a ver cualquier acta de cualquier órgano de la asociación, independientemente de que forme parte de dicho órgano o no. Para ello, la subcaracterística obligatoria `Listado` permite obtener un listado de actas de acuerdo a uno o varios criterios, modelado en la subcaracterística obligatoria `Criterio` con cardinalidad `[1..*]`. Opcionalmente se podrán imprimir las actas o bien enviarlas por correo electrónico, siempre que se seleccionen las características opcionales involucradas en la configuración específica.

[7] Hoy en día, la tecnología existente permite realizar esta implementación mediante el uso de certificados electrónicos con relativa sencillez

[8] No se ha intentado modelar esta peculiaridad de la aplicación en el modelo de características porque se trata de una explicación aclaratoria sobre el proceso que ha de llevarse a cabo para hacer realidad la funcionalidad que sí se ha modelado en el modelo

Otra subcaracterística de interés es la búsqueda de actas, porque las personas que forman cada uno de los órganos de la asociación no son siempre las mismas, y muchas veces necesitan conocer las posiciones adoptadas sobre un tema; además cualquier asociado puede también necesitar buscar un acta concreto, y al igual que en la subcaracterística `Listado`, ha de hacerse, al menos, mediante un criterio.

3.1.3.3. Restricciones

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN CONTEXTO		
`CorreoElectronico` (3.1.3.2)	◄►	`CorreoElectronico` (3.1.8.1)
A $\Rightarrow$ B	Un acta no se puede enviar por correo electrónico si no existe dicha posibilidad en la característica `EnvioMensajes`	
B $\not\Rightarrow$ A	El envío por correo electrónico sólo depende de la existencia de un servicio de transporte de correo electrónico, y que queda fuera de este modelo de características	

Cuadro 3.4: Restricciones adicionales de la característica `Reuniones`

3.1.4. Actividades

Las actividades son el eje central de toda asociación, en tanto que a través de ellas las personas participantes desarrollan un fin o interés común y se enriquecen entre todos. Por lo tanto, una asociación sin actividades de ningún tipo no tendría sentido en cuanto a su existencia, pues sus miembros asociados nunca llegarían a compartir aquello que se supone tienen en común y motivo por el que se han asociado.

Es decir, la característica que modela toda la gestión de actividades de una asociación ha de ser obligatoria en cualquier producto de la línea, aún cuando no se llegue a utilizar porque una asociación concreta decida hacerlo con algún otro sistema o metodología.

También, debido a la gran variedad de asociaciones y por tanto formas o métodos de gestión de las actividades que realizan, esta rama es la que sufrirá mayores cambios en sucesivas revisiones de la línea de productos según se elaboren nuevos productos, depuren errores de productos obtenidos y amplíe la funcionalidad en base a los comentarios de usuarios de productos ya desplegados.

La característica `Actividades` se encarga de facilitar y simplificar toda la gestión y la organización de las actividades que una asociación pueda llevar a cabo, tanto para sus propios asociados como para personas ajenas a la asociación. Esta característica también permitirá la inscripción en las diferentes actividades organizadas; es decir, no sólo integra la gestión de la organización de una actividad si no que también incluye la inscripción en esa actividad y la gestión de los inscritos.

La ilustración de la figura 3.5 muestra el diagrama de características de la característica `Actividades`.

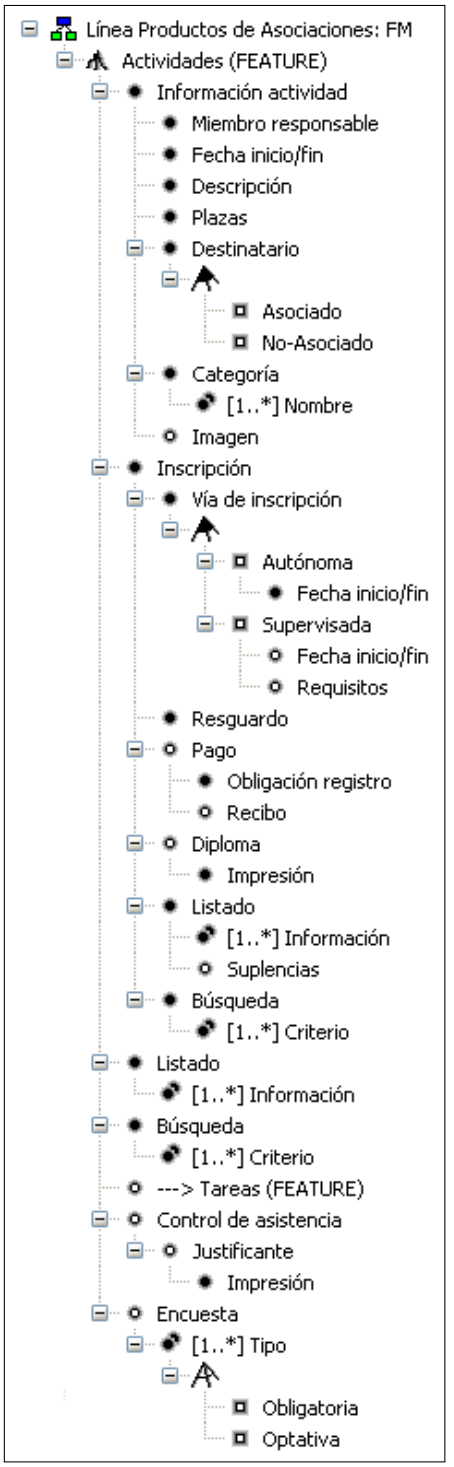


Figura 3.5: Esquema de la característica `Actividades`

3.1.4.1. Información actividad

Puesto que se van a gestionar actividades, se ha de modelar la información que representa a una actividad. Así, la subcaracterística obligatoria `Información actividad` representa toda la información relevante de una actividad.

Toda actividad ha de tener un miembro de la asociación encargado de dicha actividad, un número máximo de posibles asistentes, asi como una descripción[9] que incluirá todos los datos relevantes de la actividad: horario, requisitos, precio (si lo hubiese), fecha de comienzo y fin de la actividad,...

Una subcaracterística muy importante es `Destinatario`, y que es una característica de grupo de tipo OR, porque permite distinguir entre actividades exclusivas para asociados, para no asociados o para ambos. Como ya se ha indicado antes, una actividad debe tener un público determinado, no puede no tener destinatario, por eso se modela la relación de tipo OR. Su configuración se decidirá en tiempo de ejecución para cada una de las actividades que se gestionen con la aplicación.

Opcionalmente, se podrá asociar a la descripción de la actividad una imagen.

Por último, otra subcaracterística de interés es `Categoría`, con la que se podrán clasificar las distintas actividades, para facilitar una posterior búsqueda o listado. Esta subcaracterística tiene una característica hija clonable `Nombre` con cardinalidad `[1..*]` que representa cada una de las categorías que podrá haber; por lo tanto, se trata también de una característica cuya configuración será decidida en tiempo de ejecución, pues cada asociación tendrá un criterio determinado.

3.1.4.2. Inscripción

No sólo es interesante gestionar las diferentes actividades, sino que también resulta útil llevar la gestión de las inscripciones en las actividades, especialmente cuando el volumen de actividades o de partipantes es elevado.

La subcaracterística obligatoria `Vía de inscripción` representa la funcionalidad necesaria para permitir inscripciones autónomas, aquellas que podrá realizar cualquier persona, asociada o no, mediante la aplicación y la ayuda que ésta pueda proveer, sin intervención de nadie más que la propia persona que desea inscribirse en la actividad; o inscripciones supervisadas, aquellas que podrá realizar cualquier persona, asociada o no, pero mediante la intervención de un usuario autorizado del sistema.

Es decir, dado que el producto que se obtendrá mediante la línea de productos es una aplicación *web*, una inscripción autónoma es aquella que se realiza con una navegador *web* desde cualquier punto conectado a la red y una inscripción supervisada es aquella que habrá de realizarse con la presencia de un usuario de la asociación, con lo cual es probable que haya que desplazarse a la sede de la asociación para formalizar la inscripción.

Las inscripciones supervisadas pueden ser interesantes porque haya una serie de requisitos que deba cumplir la persona participante, y por lo tanto haya que verificarlos, o bien simplemente porque se desea entregar alguna documentación que no se puede proporcionar a través de la red o por cualquier otra razón.

[9] Este es uno de los muchos casos en los que por abreviar no se modela cada uno de los elementos que se incluirá en la descripción y que serían comunes a la característica padre

Las inscripciones autónomas tienen la peculiaridad de tener una fecha de comienzo y fin del periodo de inscripciones, que no tiene nada que ver con las fechas de comienzo de inicio y fin de la actividad (por ejemplo, se podría pensar que la fecha de fin de inscripciones podría ser el día antes al comienzo de la actividad o incluso el mismo día de comienzo).

Se trata de fechas que establecerá el responsable de la actividad. Las inscripciones supervisadas podrán tener opcionalmente de la misma subcaracterística, pero de nuevo, podrán ser fechas diferentes a la anterior.

Para cualquier inscripción en cualquier actividad, independientemente de la pertenencia a la asociación o de la forma de inscripción, se generará un resguardo que servirá como justificante de la inscripción en la actividad ante la asociación.

Se han de poder realizar búsquedas de las inscripciones mediante al menos un criterio, modelado como la subcaracterística clonable `Criterio` con cardinalidad `[1..*]`; y también obtener listados en los que se pueda escoger la información que se desee que aparezca.

En el caso de los listados, opcionalmente se podrá escoger en la configuración del producto concreto, la subcaracterística opcional `Suplencias`, para poder obtener listados de suplentes.

Las actividades que se organicen podrán ser gratuitas o no; y en este último caso, habrá de activarse en la configuración la selección de la subcaracterística opcional `Pago`, que proporcionará la funcionalidad necesaria para el control y confirmación de los pagos. En el diagrama de características propuesto se ha establecido que toda inscripción que suponga un pago incluya el registro obligatorio en la aplicación.

Hasta ahora cualquier persona podía inscribirse en una actividad, pero no tenía porqué estar registrada en la aplicación como un usuario más; sin embargo, el hecho de que tenga que abonar una cantidad por una actividad supone que se deba crear un usuario para esa persona. Opcionalmente, la aplicación permitirá la generación de recibos que justifiquen la realización del pago, una vez lo confirme la asociación.

Algunas actividades, por su carácter, pueden incluir la obtención de un diploma acreditativo de la participación en dicha actividad. Por eso, se ha modelado la subcaracterística opcional `Diploma`, que incluye la característica obligatoria `Impresión`, y a través de la cual se podrá generar e imprimir un diploma para una determinada inscripción.

3.1.4.3. Listado y búsqueda

La subcaracterística `Búsqueda` permitirá la consulta de una o varias actividades en todo el conjunto de actividades existentes, de acuerdo a uno o más criterios. Y el listado, de forma similar a la búsqueda, permitirá obtener información concreta de un subconjunto de actividades dependiendo de la información solicitada.

3.1.4.4. Tareas

Esta subcaracterística opcional es una referencia a la rama encabezada por la característica `Tareas` (3.1.5). Con ello se pretende describir y modelar que, aunque la aplicación no incluya en la configuración de la característica principal `Portal web` (3.1.1) la rama de la característica `Tareas`, porque

no se necesite en el producto concreto toda la funcionalidad de las tareas, sí se requiere la funcionalidad de las tareas aplicada exclusivamente a las actividades, y concretamente, a la organización de las tareas.

3.1.4.5. Control de asistencia

Puede llegar a ser necesario un control de la asistencia en una determinada actividad organizada porque la asociación haya podido llegar a algún acuerdo con alguna institución o entidad por el que la participación será recompensada por esa institución o entidad de alguna forma a los participantes que lo acrediten convenientemente y puedan probar que cumplen unos requisitos mínimos (de asistencia u otra índole) a dicha actividad.

Por este motivo se incluye `Control de asistencia` como una subcaracterística opcional, la cual incluye a su vez, opcionalmente también, la generación de un justificante acreditativo de los requisitos mínimos (lo habitual serán requisitos mínimos de asistencia a la actividad). En caso de que no se seleccione la generación de este justificante, la justificación de asistencia será presentada por la asociación a la entidad o institución en una lista o enviada por correo electrónico, en lugar de hacerlo el propio participante con el justificante.

3.1.4.6. Encuesta

Durante el estudio del dominio, algunas asociaciones consideraron de interés la realización de encuestas sobre las actividades por algún medio automático. Por ello, se ha modelado toda la gestión de encuestas en la subcaracterística opcional `Encuesta`.

Debido a que las encuestas que realizaban algunas de las asociaciones entrevistadas no tenían muchos elementos comunes en cuanto a contenido, formato y procedimientos (pues algunas hacían encuestas sobre la calidad de las actividades, otras encuestas sobre posibles actividades,...), se ha optado por modelar esa variabilidad mediante una característica clonable `Tipo` con cardinalidad `[1..*]`, de modo que sea la asociación, en tiempo de ejecución, la que establezca el formato de la encuesta que desea generar, así como su carácter: opcional u obligatorio.

Aunque el carácter obligatorio de una encuesta es posible que deba estar ligado a la entrega del diploma ya que de otro modo pocos asistentes a la actividad harán la encuesta de forma voluntaria pues su realización no les reportará nada; sin emabargo, esto queda a criterio de la asociación.

3.1.4.7. Restricciones

	CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
	RELACIÓN **CONTEXTO**		
≽	`Diploma` (3.1.4.2)	◄►	`ControlAsistencia` (3.1.4.5)
	A → B	Si se van a generar diplomas para los asistentes a una actividad es recomendable disponer de un control de asistencias para poder establecer criterios de derecho a diploma	
	B ↛ A	El control de asistencia sobre los asitentes a una actividad no requiere de que se les proporcione diplomas	

Continúa en la página siguiente

Viene de la página anterior

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN	**CONTEXTO**	
`Imagen` (3.1.4.1)	◄►	`Importar` (3.1.11.4)
A ⇒ B	Para adjuntar una imagen a la información de una actividad se ha de importar previamente mediante la interfaz de la aplicación	
B ⇏ A	La selección de la característica `Importar` es independiente de los tipos de ficheros a importar	
`NoAsociado` (3.1.4.1)	◄►	`InformacionAsociacion` (3.1.1.1)
A → B	No tiene sentido permitir que una actividad sea para no asociados y no se anuncie en la parte pública del portal *web*	
B ⇏ A	La información pública del portal *web* es independiente de que se organicen actividades o no	
`Asociado` (3.1.4.1)	◄►	`InformacionAsociacion` (3.1.1.2)
A ⇒ B	El anuncio de una actividad para asociados obliga a que se pueda anunciar	
B ⇏ A	Las noticias y avisos/anuncios no tienen en cuenta el contenido por lo que es indiferente que sean para una actividad	
`Autonoma` (3.1.4.2)	◄►	`InscripcionActividad` (3.1.2.2)
A → B	Para que un usuario se pueda inscribir en una actividad es recomendable que se incluya el registro en la aplicación aunque no es necesario	
B ⇏ A	La obligación de registro en la aplicación por una inscripción en una activdad es independiente de que esa inscripción sea autónoma o supervisada	
`Pago` (3.1.4.2)	◄►	`Contabilidad` (3.1.1.2)
A ⇒ B	Si una actividad lleva consigo un determinado coste, es imprescindible llevar un registro de ese pago mediante la característica `Contabilidad`	
B ⇏ A	La funcionalidad de contabilidad en la aplicación no tiene porqué incluir que se tenga que poder realizar actividades con coste	
`Tareas` (3.1.4.4)	◄►	`Actividad` (3.1.5.1)
A ⇒ B	En caso de que se decida organizar las actividades en forma de tareas, la categoría de las tareas deberá ser necesariamente la subcaracterística `Actividad`	
B → A	La categoría de una tarea es independiente de se haya seleccionado la subcaracterística `Tareas` en la rama de la característica `Actividad`, sin embargo es recomendable su selección	

Cuadro 3.5: Restricciones adicionales de la característica `Actividades`

3.1.5. Tareas

Una tarea es una o más acciones que han de llevarse a cabo, por parte de uno o más asociados, para completar un trabajo. En el estudio del dominio explicado en el apartado 2.4.1, se observó que una de las "quejas" comunes en las asociaciones era la falta de control acerca de quién hacía qué, de

modo que muchas veces había tareas que no se hacían o bien se tardaba mucho en hacer porque no se podía llevar a cabo un seguimiento[10] de las mismas.

En consecuencia, había que esperar a una reunión para saber acerca del estado de las diferentes tareas asignadas o bien contactar directamente (por teléfono o correo electrónico) con el responsable de la tarea.

La funcionalidad de las características fue propuesta por una de las asociaciones entrevistadas, y posteriormente fue revisada y caracterizada en el diagrama de características, encabezado por la característica `Tareas`, que se muestra en la ilustración de la figura 3.6.

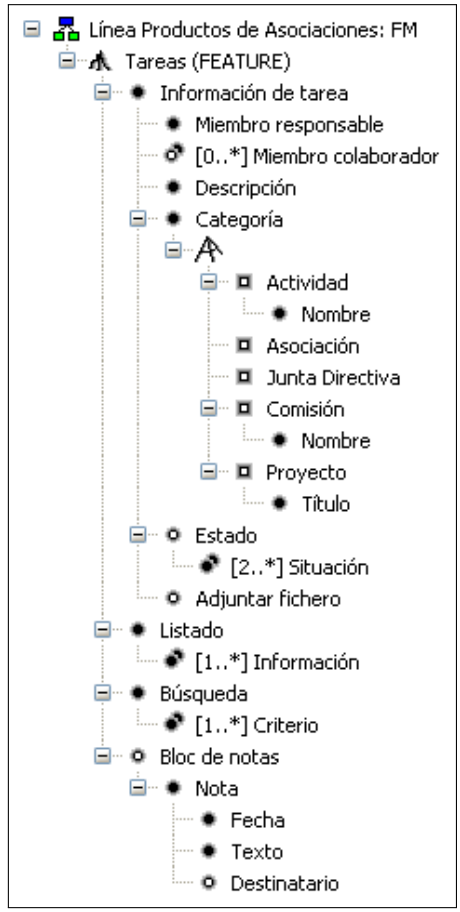


Figura 3.6: Esquema de la característica `Tareas`

[10] Seguimiento en el sentido de poder conocer los cambios no en tiempo real pero sí al poco de producirse dichos cambios

3.1.5.1. Información tarea

Como cabe esperar de acuerdo a lo explicado anteriormente, uno de los conjuntos de características más importantes de esta "rama" será el que agrupe a los elementos que representan una tarea y que la definen en la aplicación.

La subcaracterística `Información de tarea` modela los detalles que definirán a una tarea. Toda tarea tendrá un miembro responsable de la misma, que se encargará de llevar a cabo la tarea, y en el caso de que hubiese miembros colaboradores, modelado como la subcaracterística clonable `Miembros colaboradores` con cardinalidad `[0..*]`, también se encargaría de coordinar el reparto de acciones necesarias para completar la tarea.

Junto con una descripción, también se habrá de clasificar la tarea de acuerdo a un conjunto de categorías, modelado como un grupo de subcaracterísticas de tipo XOR, ya que toda tarea está asociada a uno de los elementos del grupo de subcaracterísticas, y aunque podría haber una tarea que afectase a dos o más categorías, siempre es originada por una categoría y es ahí donde ha de clasificarse.

Puede ser necesario para alguna tarea la inclusión de un fichero; por ejemplo, una tarea clasificada en la subcaracterística `Actividad`, y cuya descripción se refiera a la búsqueda del presupuesto del alquiler de un local para la celebración de una actividad, puede dar como resultado que el presupuesto se formalice en un fichero que se importe a la aplicación, para que el responsable de la actividad lo consulte, una vez observe que el estado de la tarea ha cambiado o que se le ha notificado, en lugar de recibirlo por otros medios. Por ello se incluye la subcaracterística opcional `Adjuntar fichero`.

También puede ser interesante llevar el control del estado de cada tarea mediante la aplicación; para ello, basta seleccionar en la configuración la subcaracterística opcional `Estado` y definir las diferentes situaciones que puede tener una tarea.

Como mínimo hay que definir dos situaciones, una ha de indicar que la tarea está pendiente, y la otra que la tarea se ha completado; no obstante, se podrán definir situaciones intermedias si así se considera necesario para un producto concreto. El estado sólo podrá ser modificado por el responsable de la tarea cuando lo considere adeacuado.

3.1.5.2. Listado, Búsqueda

La posibilidad de buscar tareas de acuerdo a un criterio y la obtención de listados de tareas seleccionando la información que aparecerá en ellos, es absolutamente imprescindible ya que es la única forma de conocer las tareas asociadas a cada categoría y por lo tanto, poder llevar un control de las tareas, así como su estado, en caso de que se haya seleccionado en la configuración.

3.1.5.3. Bloc de Notas

La subcaracterística opcional `Bloc de notas` es una simulación muy simplificada del equivalente en papel, de modo que cualquier usuario puede crear una nota en la que escribe lo que considere necesario (se supone que será algo relacionado con alguna tarea), y posteriormente podrá borrar la nota cuando ya no le haga falta.

En caso de que se seleccione la subcaracterística opcional `Destinatario`, la nota estará dirigida al usuario indicado y cuando éste la lea, también podrá eliminarla.

3.1.5.4. Restricciones

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN CONTEXTO		
`AdjuntarFichero` (3.1.5.1)	◄►	`Importar` (3.1.11.4)
A ⇒ B	Para adjuntar un fichero a una tarea primero se ha de importar mediante la interfaz de la aplicación	
B ⇏ A	La selección de la característica `Importar` no tiene ninguna dependencia con la caracterísitca `AdjuntarFichero`	

Cuadro 3.6: Restricciones adicionales de la característica `Tareas`

3.1.6. Contabilidad

La contabilidad reúne la funcionalidad básica para poder llevar a cabo la contabilidad de una asociación, incluyendo una posible actividad de facturación de servicios o incluso la contratación de personal por parte de la asociación.

La característica `Contabilidad` no pretende emular toda la funcionalidad que se ofrece en multitud de paquetes comerciales, gratuitos y *open source*. Simplemente se ha de proporcionar una mínima funcionalidad, lo suficientemente potente como para que el tesorero de la asociación pueda llevar a cabo las tareas contables de la asociación, y los asociados puedan consultar los gastos e ingresos que pudiese haber.

La funcionalidad de la característica `Contabilidad` se encuentra caracterizada y representada en la ilustración de la figura 3.7.

3.1.6.1. Libro de cuentas

La subcaracterística obligatoria `Libro de cuentas` representa la funcionalidad principal de la característica `Contabilidad`, ya que es la que permite llevar a cabo todas las tareas de control de los ingresos y gastos de la asociación; por lo tanto, se trata de una característica común en todos los productos de la línea.

Esta subcaracterística es similar a la subcaracterística `Libro de cuentas`, en tanto que también proporciona una simulación, aunque en este caso del libro de cuentas. En todo libro de cuentas se ha de anotar todo ingreso o gasto (es decir, hay que llevar un control de todo el dinero en el que la asociación lo recibe o lo entrega de alguna forma), estas anotaciones pueden ser de dos tipos, "haberes" (ingresos o dinero entrante) y "debes" (gastos o dinero saliente).

Las características que lo representan, `Haber` y `Debe`, son subcaracterísticas pertenecientes a la característica de grupo de tipo XOR, cuya configuración se realizará en tiempo de ejecución, concretamente en el momento en el que se seleccione añadir una nueva anotación en el libro de cuentas.

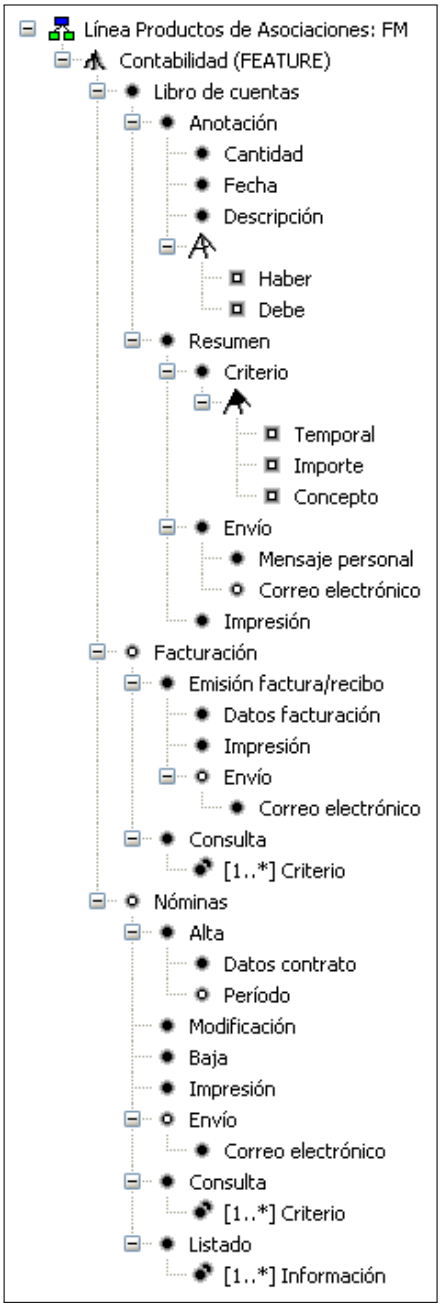


Figura 3.7: Esquema de la característica `Contabilidad`

Toda anotación tendrá una fecha, el importe del ingreso o del gasto y una descripción que detalle el concepto, para poder localizar el documento que justifique la anotación realizada.

Tan importante es poder registrar las anotaciones como obtener resúmenes de ingresos y gastos. Para poder obtener un resumen, ha de especificarse en tiempo de ejecución al menos uno de los tres criterios establecidos en la subcaracterística `Criterio`: `Temporal`, `Importe` o `Concepto` para restringir las anotaciones que se desea consten en el resumen.

Los resúmenes se podrán imprimir y también enviar mediante mensajes personales o por correo electrónico, aunque en este último caso habrá de seleccionarse la característica opcional `Correo electrónico` en la configuración del producto concreto.

3.1.6.2. Facturación

Es posible que una asociación ofrezca algún servicio por el que cobre y deba entregar un justificante, por lo tanto estaría realizando una facturación. Por lo tanto, la subcaracterística `Emisión factura/recibo` se marca como obligatoria; aunque si no se selecciona la característica opcional `Facturación`, no se incluirá la funcionalidad correspondiente puesto, que depende de esta.

La emisión de una factura o un recibo no es más que la transposición de una anotación en el libro de cuentas de un haber en la que se incluyen los datos de la persona o entidad facturada. Toda factura se ha de poder imprimir y opcionalmente se podrá enviar por correo electrónico.

La subcaracterística `Consulta` permite realizar una búsqueda concreta de las facturas de acuerdo a uno o más criterios indicados. En realidad, la consulta no es más que una versión concreta de la subcaracterística `Resumen` de la característica `Libro de cuentas`, con unos criterios predefinidos y no modificables, para acotar la consulta a ingresos cuyo concepto sea un servicio realizado por la asociación.

3.1.6.3. Nóminas

Al igual que la subcaracterística `Facturación`, esta otra subcaracterística, `Nóminas`, centraliza la funcionalidad referente a las contrataciones[11] de personal que realice la asociación.

Cada nueva contratación supone un alta de una nómina en la que se indicarán los datos del contrato y el período del mismo. De este modo se automatiza la creación de las anotaciones periódicas del sueldo en el libro de cuentas. Lógicamente las nóminas se podrán modificar y dar de baja, así como imprimir o enviar por correo electrónico siempre que se seleccione esta última opción en la configuración concreta.

Las subcaracterísticas obligatorias `Consulta` y `Listado` permiten realizar por un lado búsquedas de nóminas almacenadas de acuerdo a uno o varios parámetros, y por otro lado la obtención de listados de acuerdo a uno o más criterios dados en el momento de realización de la operación para reducir la información que se obtendrá en el listado.

[11] La contratación de servicios correspondería a una anotación en el libro de cuentas indicando en el concepto el detalle del servicio y la factura correspondiente

3.1.6.4. Restricciones

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN CONTEXTO		
`CorreoElectronico` (3.1.6.1) | `Envio` (3.1.6.2) | `CorreoElectronico` (3.1.6.3)	◄►	`CorreoElectronico` (3.1.8.1)
$A \Rightarrow B$	Un resumen, factura o nómina no se pueden enviar por correo electrónico si no se ha activado el envío de mensajes por correo electrónico en la rama de `Comunicación` (3.1.8)	
$B \not\Rightarrow A$	El envío por correo electrónico sólo depende de la existencia de un servicio de transporte de correo electrónico, por lo que es independiente de lo que se pueda enviar a través de él o del lugar en el que se pueda utilizar	

Cuadro 3.7: Restricciones adicionales de la característica `Contabilidad`

3.1.7. Información asociación

La característica `Información asociación` reúne toda la funcionalidad referente a la información genérica que la asociación desee proporcionar tanto a sus asociados, como a los no asociados. Esta información será proporcionada en forma de noticias y anuncios. Por ejemplo, para informar sobre una próxima actividad que se vaya a organizar se utiliza una noticia, y para comunicar a los asociados una reunión se utiliza un aviso/anuncio.

Por lo tanto, esta característica proporciona información a un grupo de usuarios no específico, sólo se distingue entre grupos de usuarios y no usuarios individuales.

La ilustración de la figura 3.8 muestra la jerarquía de características de la rama de la característica `Información asociación`.

3.1.7.1. Información

La subcaracterística obligatoria `Información` describe a la entidad que representa la información a proporcionar, tanto a la propia información que se desea transmitir como la meta-información (o también meta-datos[12]) asociada a ella.

Toda información tiene asociada una meta-información a destacar, una fecha de publicación y una categoría. La categoría se podría haber modelado como una característica clonable o bien como un grupo de características de tipo OR en el caso que se conociese de antemano el espacio de categorías existentes (y dejando la libertad suficiente como para que una información se pueda asignar a diferentes categorías).

[12]Resulta complicado tratar el tema de la información en un sistema informático con cierta abstracción pero en un lenguaje no muy técnico debido a que cualquier sistema informático trata información, y por tanto se produce un juego de palabras. En el contexto que se plantea aquí, la información se refiere a conocimiento relevante para alguien

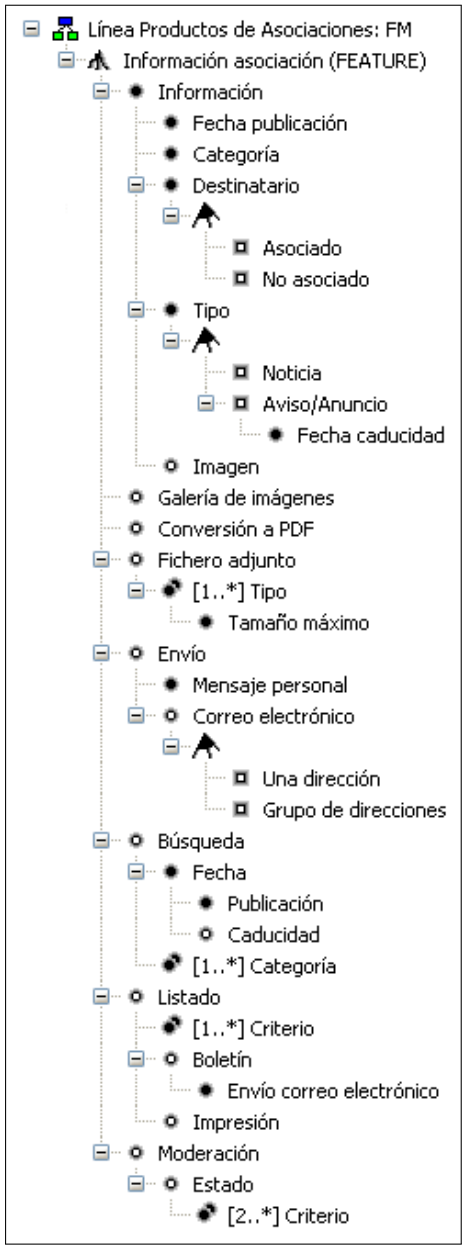


Figura 3.8: Esquema de la característica `Información asociación`

Sin embargo se ha modelado como una característica unitaria, ya que es proporcionada por el usuario responsable de la información, y además en forma de etiquetas (*tags*); es decir, en lugar de establecer un conjunto de categorías, se anima a que sea el usuario el que introduzca palabras relevantes para definir la categoría.

Como ya se ha indicado, la información tiene que tener un destinatario, por ello la subcaracterística obligatoria `Destinatario`, aunque es posible ofrecer una información (en tiempo de ejecución) tanto para asociados como para no asociados.

Y del mismo modo, se ha de escoger el tipo de información a publicar, bien una noticia o un aviso/anuncio; en el último caso, ha de incluirse obligatoriamente una fecha de caducidad.

Toda información podrá tener opcionalmente una imagen asociada al propio texto de la noticia o anuncio/aviso, aunque tendrá que cumplir una serie de requisitos que vendrán dados por los requisitos especificados en la plantilla gráfica del sistema que el administrador haya configurado en el producto concreto.

3.1.7.2. Galería de imágenes

La característica opcional `Galería de imágenes` es un tipo particular de noticias en donde se agrupan una colección de noticias (y también se clasifican mediante la subcaracterística `Categoría` de la característica `Información`) que no tienen más información que una imagen y a lo sumo un pie de imagen en la subcaracterística `Descripción`.

Por lo tanto, la galería de imágenes requiere que la subcaracterística `Imagen` de la característica `Información` esté habilitada.

3.1.7.3. Conversión a PDF y Fichero adjunto

Puede ser necesaria la conversión de un aviso/anuncio o una noticia a un formato estático, fijo y relativamente estándar, como puede ser el formato PDF (*Portable Document Format*); o también la inclusión de ficheros adjuntos porque la información que contengan no sea compatible con el medio en el que se usa la aplicación (web).

Sin embargo, en el caso de que se proporcione la funcionalidad de adjuntar ficheros, habrá de definirse qué tipo de ficheros se va a permitir, y para cada uno de ellos, el tamaño máximo que la aplicación admitirá.

Las características `Conversión a PDF` y `Fichero adjunto` son características opcionales que habrán de ser seleccionadas en la configuración del producto concreto, la primera en tiempo de desarrollo y la segunda en tiempo de ejecución.

3.1.7.4. Envío

Las noticias y avisos/anuncios podrán enviarse si se selecciona la característica opcional `Envío` en la configuración del producto en desarrollo. El envío deberá poder realizarse de forma obligatoria mediante mensajes personales, si bien, esto exige que el remitente de la noticia o aviso/anuncio sea un usuario válido en la aplicación.

No obstante también se podrá enviar por correo electrónico una noticia o un aviso/anuncio, siempre que se seleccione la subcaracterística `Correo electrónico` en la configuración en tiempo de desarrollo, a una dirección o a un grupo de direcciones.

En el caso de que el destinatario sea un grupo de direcciones, el usuario remitente de la información deberá ser un usuario autorizado a ello para evitar envíos masivos (se recomienda echar un vistazo a las restricciones para poder concretar estas condiciones).

3.1.7.5. Búsqueda y Listado

La característica opcional `Búsqueda` contiene la funcionalidad que permite realizar búsquedas de noticias y/o avisos/anuncios. Para realizar las búsquedas, como mínimo ha de permitirse hacerlo mediante la fecha de publicación, pues tanto las noticias como los avisos tienen una fecha de publicación, y opcionalmente mediante la fecha de caducidad (aunque esta última sólo será aplicable a los anuncios/avisos).

Es imprescindible indicar, junto con la palabra o palabras claves de la búsqueda, una o más categorías (en tiempo de ejecución) para poder realizar la búsqueda (y así acotar los resultados y reducir el tiempo de búsqueda).

Puede resultar interesante, especialmente para los anuncios/avisos, la posibilidad de generar listados, seleccionando en la configuración la característica opcional `Listado` para la que necesariamente ha de indicarse qué información se ha de incluir en el listado, y que se ha modelado en la subcaracterística clonable `Información` con cardinalidad `[1..*]`.

Opcionalmente, será posible imprimir el listado. También, si se desea se podrán generar boletines de noticias o avisos/anuncios para ser enviados por correo electrónico, aunque sólo podrá hacerlo un administrador autorizado.

3.1.7.6. Moderación

Aunque tanto las noticias como los avisos/anuncios solo pueden ser puestos por usuarios autorizados es conveniente que exista la posibilidad de moderarlos con la idea de corregir, por ejemplo, la categoría o resumir un anuncio si es demasiado extenso.

Para moderar se pueden establecer estados, siempre que se escoja en la configuración del producto en desarrollo la subcaracterística opcional `Estado`. La cardinalidad de la subcaracterística clonable que representa los estados, `Criterio`, es `[2..*]` porque debe haber, al menos, dos estados posibles a escoger, publicado y no publicado; podrá haber otros estados, pero si se opta por moderar, como mínimo debe poderse escoger entre esos dos.

3.1.7.7. Restricciones

	CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
	RELACIÓN CONTEXTO		
⩾	GaleriaImagenes (3.1.7.2)	◄►	Imagen (3.1.7.1) ∧ Noticia ∧ ¬AvisoAnuncio (3.1.7.1)
	A ⇒ B	Las galerías de imágenes están compuestas de noticias que contienen una imagen, por lo que es necesario que se cumplan ambas condiciones	
	B ⇏ A	El tipo de una noticia o el que lleve asociada una imagen es independiente de que forme parte de una galería de imágenes	
⩾	Busqueda (3.1.7.5)	◄►	Listado (3.1.7.5)
	A → B	Es recomendable poder generar listados a partir de los resultados de una búsqueda	
	B → A	La generación de listados se realizará más rapidamente si se pueden hacer búsquedas	
	Asociado | NoAsociado (3.1.7.1)	◄►	NoAsociados (3.1.11.3)
	A → B	Es recomendable que se active el control de acceso para los no asociados y de este modo limitar el acceso a las noticias o avisos/anuncios	
	B ⇏ A	El acceso por zonas es independiente del destinatario de una noticia o aviso/anuncio	
	FicheroAdjunto (3.1.7.3) | Imagen (3.1.7.1)	◄►	Importar (3.1.11.4)
	A ⇒ B	Para adjuntar un fichero o anuncio a una noticia o aviso/anuncio primero se ha de importar mediante la interfaz de la aplicación	
	B ⇏ A	La selección de la característica Importar es independiente del lugar y forma en que se use	
	CorreoElectronico (3.1.7.4) ∧ Boletin (3.1.7.5)	◄►	CorreoElectronico (3.1.8.1)
	A ⇒ B	Una noticia, aviso/anuncio o boletín se podrá enviar por correo electrónico, si se ha habilitado la característica CorreoElectronico	
	B ⇏ A	El envío por correo electrónico sólo depende de la existencia de un servicio de transporte de correo electrónico, es independiente de que se publiquen noticias o avisos/anuncios, o que se generen boletines	
	Busqueda (3.1.7.5)	◄►	Busqueda (3.1.1.3)
	A → B	Es recomendable activar la búsqueda en la característica Portal web para que se puedan realizar búsquedas en las noticias o anuncios/avisos que se publiquen bajo la subcaracterística Información asociación	
	B ⇏ A	La búsqueda en el portal *web* es independiente de que haya noticias o avisos/anuncios	

Cuadro 3.8: Restricciones adicionales de la característica Información asociación

3.1.8. Comunicación

La característica `Comunicación` describe el intercambio de mensajes entre los usuarios de la aplicación y con la asociación, tanto de forma "privada" (mediante el envío de información en forma de mensaje dirigido a un usuario o grupo de usuarios concretos) como de forma "pública" (mediante el envío de información en forma de mensaje a un conjunto de usuarios).

La comunicación se divide en dos características obligatorias, `Envío de mensajes` y `Consultas`, y una característica opcional, `Foros`.

Por cuestiones de privacidad, los mensajes entre usuarios no deberán ser registrados[13] en el registro de actividad de la rama de la característica `Administración`.

La ilustración de la figura 3.9 muestra el diagrama de características de la rama de la característica `Comunicación`.

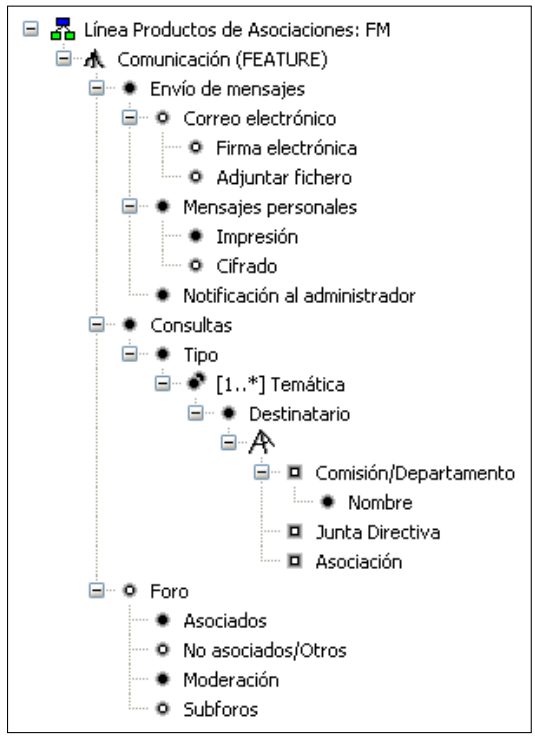


Figura 3.9: Esquema de la característica `Comunicación`

3.1.8.1. Envío de mensajes

El envío de mensajes permite el intercambio de información entre los usuarios de forma escrita. Esta subcaracterística permite realizar dos tipos de mensajes, `Mensajes personales` y `Notifica-`

[13] Se podría pensar en incluir una restricción de tipo excluyente pero ello llevaría a no poder disponer de la característica excluida, y lo que se plantea es más una cuestión ética y legislativa que tecnológica

`ción al administrador`.

Los mensajes personales son comunicaciones que realizan los usuarios de la aplicación entre sí sin intervención de ningún administrador del sistema u otro usuario sólo se pueden realizar desde la aplicación y no es posible consultarlos en ningún otro sitio más que en la aplicación, aunque se pueden imprimir.

Todo usuario del sistema dispone de un buzón donde se almacenan los mensajes que recibe. Es posible cifrar los mensajes personales (siempre que se seleccione la característica opcional `Cifrado`), para garantizar la confidencialidad de su contenido ya que la propia aplicación garantiza la autenticidad del remitente del mensaje, pero la clave de cifrado deberá ser comunicada al receptor del mensaje por algún medio seguro.

Un tipo especial de mensajes personales, que no requiere ser realizado por un usuario válido de la aplicación es el de la subcaracterística `Notificación al administrador`; su única utilidad es informar al administrador de la aplicación cualquier problema que surja en cualquier punto. Estos mensajes tienen un destinatario fijo, el administrador, y no es necesario proporcionar una forma de contacto en caso de que no se requiera contestación.

Los mensajes pueden tener un destino fuera de la aplicación, y en concreto una cuenta de correo electrónico, gracias a la subcaracterística `E-mail`, y siempre que se incluya en la configuración concreta. De forma similar al cifrado de los mensajes personales, es posible garantizar la autenticidad del mensaje y por tanto asegurar la procedencia, siempre que se firme electrónicamente. Si se desea, es posible adjuntar un fichero en el correo electrónico.

3.1.8.2. Consultas

Las consultas son mensajes similares a la notificación al administrador pero con el objetivo de realizar una pregunta u obtener una información concreta de la asociación, y por lo tanto ha de proporcionarse una forma de contacto, pues si no se hace, la consulta no tendría sentido.

Toda consulta ha de ser de un tipo concreto, y cada uno de esos tipos, deberá ser definido por la asociación. Como mínimo debe haber una temática; por eso la subcaracterística clonable `Temática` con cardinalidad `[1..*]`. Como ejemplo de temáticas, se podrían configurar actividades, información de la asociación, información sobre asociacionismo, propuesta de colaboración, sugerencias,...

El destinatario de la consulta, es la asociación, aunque en último término será un usuario o grupo de usuarios encargados de la temática de la consulta. La subcaracterística `Destinatario` refleja el receptor (genérico) de la consulta; en la configuración de cada temática habrá que seleccionar uno y sólo uno, de entre los tres destinatarios posibles: Comisión/Departamento, Junta Directiva y Asociación.

El aspecto del destinatario es posible que cambie dependiendo de la política que se quiera establecer; en este modelo se ha optado por una solución relativamente restrictiva en donde cada temática tiene un único destinatario. Se podría dar el caso que hubiese varios destinatarios y bastaría convertir la característica de grupo de tipo XOR a tipo OR, aunque ambas opciones permiten una ampliación de los destinatarios sin alterar drásticamente el modelo de características.

3.1.8.3. Foro

Otra forma de comunicación que se puede incluir en la aplicación concreta es un foro (en la red), también conocido como foro de mensajes, de opinión o de discusión. El objetivo de la característica opcional `Foro` es dar soporte *software* a una discusión u opinión sobre un tema planteado por un usuario de la aplicación, por escrito y para un público determinado, o para cualquier público.

Todo usuario puede plantear nuevos temas y responder temas existentes, si bien no puede modificar las respuestas de otros usuarios. Las respuestas a los temas planteados o respuestas de usuarios se pueden hacer en cualquier momento.

En la línea de productos para asociaciones, cobra sentido la existencia de un foro como punto de encuentro entre los asociados, ya que no es posible, ni factible, organizar reuniones (en las diferentes formas previstas: asamblea, junta, comisión,...) para discutir temas que no requieran la presencia explícita de los asociados o que no sea necesario tratar en un día. Por eso, en caso de seleccionar la subcaracterística `Foro`, como mínimo debe poder ser usado por los asociados, aunque también se podrá permitir que lo usen los no asociados. Aunque se permita el acceso a usuarios diferentes a los asociados o no, todo foro debe ser moderado para evitar abusos.

La subcaracterística `Subforos` es necesaria cuando se desea dividir el foro en subforos, bien por temas, por tipo de usuario o por ambos o simplemente por cuestiones organizativas para no tener todos las discusiones (o temas) en el mismo sitio.

Aunque la característica que representa la funcionalidad de un foro, `Foro`, podría tener un carácter obligatorio en el análisis del dominio, se ha modelado como opcional, porque lo habitual son las reuniones físicas donde se tratan los temas. Se proporciona dicha funcionalidad por su utilidad y por la expansión en el uso de las nuevas tecnologías que conllevará que esas reuniones se reduzcan estrictamente a las imprescindibles.

3.1.8.4. Restricciones

	CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
	RELACIÓN	**CONTEXTO**	
≥	`NoAsociadosOtros` (3.1.8.3)	◄►	`Subforos` (3.1.8.3)
	$A \Rightarrow B$	El acceso de usuarios no asociados al foro conlleva la división en subforos (incluyendo uno exclusivo para asociados y otro para no asociados) para facilitar la organización y gestión	
	$B \not\Rightarrow A$	Se puede dividir el foro en subforos sin necesidad de proporcionar acceso a los no asociados	
	`AdjuntarFichero` (3.1.8.1)	◄►	`Importar` (3.1.11.4)
	$A \Rightarrow B$	Para adjuntar un fichero a un correo electrónico primero se ha de importar mediante la interfaz de la aplicación	
	$B \not\Rightarrow A$	La selección de la característica `Importar` es independiente del lugar de la aplicación donde se use, así como de los tipos de ficheros que se puedan importar con ella	

Continúa en la página siguiente

Viene de la página anterior

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN CONTEXTO		
`NoAsociadosOtros` (3.1.8.3)	◄►	`NoAsociados` (3.1.11.3)
A → B	Es recomendable establecer un acceso por zonas, si se permite el acceso al foro por parte de usuarios no asociados para garantizar la confidencialidad	
B ↛ A	El acceso de no asociados al sistema es independiente de la existencia de un foro	

Cuadro 3.9: Restricciones adicionales de la característica `Comunicación`

3.1.9. Inventario

El inventario centraliza toda la organización de los bienes tangibles de la asociación (desde el punto de vista del conocimiento de lo que la asociación tiene y el uso que se le está dando), en oposición a la organización de los bienes no tangibles (y concretamente bienes inmuebles) modelados en la rama de la característica `Patrimonio` (3.1.10).

La ilustración de la figura 3.10 muestra el esquema de características de la rama de `Inventario`.

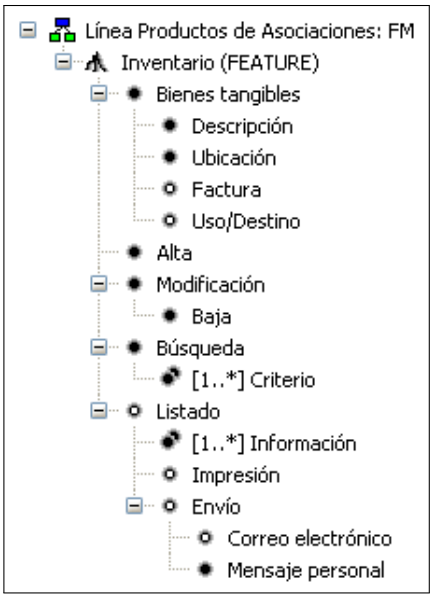


Figura 3.10: Esquema de la característica `Inventario`

3.1.9.1. Bienes tangibles

Como la organización del inventario se hace con el objetivo de elaborar y disponer de un registro de todos esos bienes, la característica `Inventario` incluye una serie de subcaracterísticas obligatorias que son necesarias para poder ofrecer una funcionalidad completa en caso de incluir el inventario en el producto obtenido en la línea de productos *software*.

La característica `Bienes tangibles` agrupa la información que se almacenará para cada elemento del inventario; como mínimo ha de incluirse una descripción y una ubicación. Opcionalmente es posible incluir el detalle del uso o destino que se da al bien y la factura asociada al bien,; no obstante, esta última subcaracterística opcional tiene restricciones que se pueden consultar al final del apartado de la rama de la característica `Inventario`.

3.1.9.2. Alta, modificación/baja y búsqueda

El almacenamiento de cada nuevo bien en el inventario se representa mediante la subcaracterística `Alta`, y la modificación de alguna información asociada a un bien determinado (por ejemplo la ubicación) se representa mediante la subcaracterística obligatoria `Modificación` que incluye a la característica `Baja`, en tanto que retirar un bien del inventario supone una modificación de la información almacenada en el inventario.

Logicamente es necesario poder realizar búsquedas de los bienes del inventario y por eso la característica obligatoria `Búsqueda`, así como la subcaracterística clonable que depende de ella `Criterio` con cardinalidad `[1..*]`, pues toda búsqueda debe hacerse con al menos un parámetro para acotarla y evitar que el número de resultados se dispare.

3.1.9.3. Listado

En caso de ser necesario o de interés para una asociación concreta, se puede escoger la posibilidad de generar listados, seleccionando en la configuración la característica opcional `Listado` para la que necesariamente ha de indicarse qué información incluir en el listado, y que se ha modelado en la subcaracterística clonable `Información` con cardinalidad `[1..*]`.

Opcionalmente, es posible imprimir el listado o enviarlo, y en este último caso, se hará mediante un mensaje personal, pero también podrá enviarse por correo electrónico siempre que se seleccione en la configuración la subcaracterística opcional `Correo electrónico`.

3.1.9.4. Restricciones

CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN **CONTEXTO**		
`Factura` (3.1.9.1)	◄►	`Importar` (3.1.11.4)
A ⇒ B	No se pueden almacenar facturas si no es posible importarlas mediante la interfaz de la aplicación	
B ⇏ A	La selección de la característica `Importar` no tiene ninguna dependencia con la caracterísitca `Factura`	

Continúa en la página siguiente

Viene de la página anterior

	CARACTERÍSTICA(S) A ◄► CARACTERÍSTICA(S) B	
RELACIÓN	CONTEXTO	
	`Envio` (3.1.9.3) ◄► `CorreoElectronico` (3.1.8.1)	
A ⇒ B	Un listado no se puede enviar por correo electrónico si no existe dicha posibilidad en la característica `EnvioMensajes`	
B ⇏ A	El envío por correo electrónico sólo depende de la existencia de un servicio de transporte de correo electrónico, y que queda fuera de este modelo de características	

Cuadro 3.10: Restricciones adicionales de la característica `Inventario`

3.1.10. Patrimonio

La característica `Patrimonio` agrupa toda la funcionalidad referente a la gestión del patrimonio que una asociación posee (por ejemplo, una sede). En este caso, patrimonio se refiere a todo bien inmueble, y por eso, la descripción de la característica obligatoria `Bienes inmuebles`.

En la ilustración de la figura 3.11 se representa la jerarquía de características de la rama de `Patrimonio`.

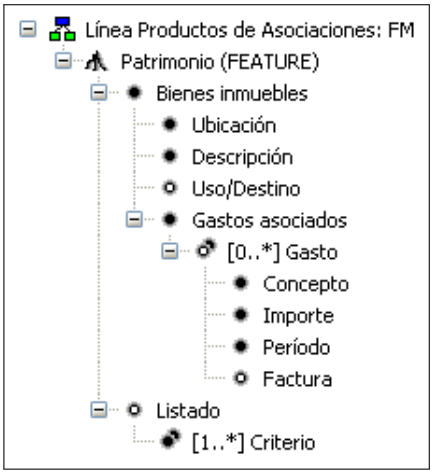


Figura 3.11: Esquema de la característica `Patrimonio`

3.1.10.1. Bienes inmuebles

Todo bien inmueble ha de tener una descripción y una ubicación, así como unos gastos, representados mediante la subcaracterística obligatoria `Gastos asociados`; sin embargo, es posible que el bien inmueble pueda tener unos gastos asociados que no sean abonados por la asociación (por ejemplo, una sede cedida por una institución pública la cual se hace cargo de los gastos de todo el edificio

en donde esté la sede cedida), por este motivo, los gastos asociados se han modelado mediante una subcaracterística clonable `Gasto` con cardinalidad `[0..*]`, y en caso de que haya algún gasto, éste incluirá el concepto y el importe de dicho gasto así como la subcaracterística `Período` que modela la frecuencia de dicho gasto.

La subcaracterística opcional `Factura` tiene como objeto modelar el almacenamiento del justificante del gasto, ya que hoy en día, el uso de facturas y recibos electrónicos está muy extendido, por lo que es posible asociar el justificante al gasto, si bien se necesita de la subcaracterística `Importar` de la característica `Repositorio de ficheros` en la rama de `Administración`.

Otra característica opcional relevante para un bien inmueble es el uso o destino que se hace de dicho bien; aunque esta característica se podría integrar en la característica obligatoria `Descripción`, se ha separado de ella con el objetivo de diferenciar la descripción y el uso; también se ha hecho esta separación pensando en una futura ampliación del modelo de características, en la que el uso o destino necesiten o requieran de una gestión más completa.

3.1.10.2. Listado

Es posible obtener un listado de bienes inmuebles, si bien se ha modelado como una característica opcional, pues no es habitual que una asociación tenga más de un bien inmueble; no obstante, es posible. En ese caso, la selección de la característica `Listado` incluye la subcaracterística clonable `Criterio` con cardinalidad `[1..*]`, que representa el criterio (o criterios) con el que se generará un listado de los bienes inmuebles.

Si se desea permitir la impresión del listado, se deberá seleccionar en la configuración del producto la subcaracterística opcional `Impresión`.

3.1.10.3. Restricciones

	CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
RELACIÓN	**CONTEXTO**		
	`Factura` (3.1.10.1)	◄►	`Importar` (3.1.11.4)
A ⇒ B	No se pueden almacenar facturas si no es posible importarlas mediante la interfaz de la aplicación		
B ⇏ A	La selección de la característica `Importar` es independiente del lugar donde se use y del tipo de ficheros que se puedan importar		

Cuadro 3.11: Restricciones adicionales de la característica `Patrimonio`

3.1.11. Administración

La característica `Administración` proporciona el conjunto de herramientas necesarias para el mantenimiento y control de la aplicación de gestión de la asociación. Las diferentes herramientas que esta característica proporciona configuran la aplicación de forma *on-line*, es decir, no es necesario deshabilitar el acceso de los usuarios a la aplicación, o incluso reiniciar la máquina que aloje

la aplicación, cuando se estén realizando labores de mantenimiento, y por lo tanto, cualquier cambio se verá reflejado de forma inmediata. Aquellos cambios que afecten a usuarios que estén en línea se activarán en la siguiente sesión de trabajo.

No obstante, las labores de mantenimiento sobre la base de datos, como la característica `Copia de seguridad` llevarán consigo un bloqueo en el acceso de escritura a la base de datos ,mientras se realice la copia de seguridad, aunque se podrá leer la información contenida y no permitirá el inicio de sesión a ningún usuario.

Las herramientas de administración, se dividen en cinco categorías (representadas mediante características): `Preferencias`, `Usuarios`, `Repositorio de ficheros`, `Registro de actividad` y `Copia de seguridad`. En la ilustración de la figura 3.12 se representa la jerarquía de características de la rama de `Administración`.

3.1.11.1. Acceso seguro

La característica `Acceso seguro` se encarga de proporcionar una vía de inicio de sesión seguro en la administración de la aplicación; por ejemplo, utilizando la capa segura SSL (*Secure Socket Layer*) sobre el protocolo HTTP (*HyperText Transport Protocol*) y así evitar ataques de autenticación como MitM (*Man in the Middle*), o también asociar la IP al inicio de sesión en las herramientas de administración para evitar una suplantación.

Esta característica es opcional ya que, si bien es importante en cualquier producto de la línea de productos *software*, su realización y disponibilidad en la línea depende de factores *hardware*, más que *software*.

Posiblemente se podría desglosar en subcaracterísticas que reflejasen los factores *software*, pero no se hace así porque no es objeto ni de este apartado ni de este Proyecto Fin de Carrera el describir las medidas de seguridad de una aplicación *software*; simplemente se pretende destacar una implementación de una seguridad adicional a la habitual comprobación de credenciales y niveles de acceso a las diferentes zonas.

3.1.11.2. Preferencias

La característica obligatoria `Preferencias` reúne la configuración genérica de la aplicación aplicable en cualquier actuación. Por ejemplo, el nombre que aparecerá en el sitio, el servidor de correo electrónico a utilizar (en caso de que se permita la comunicación por correo electrónico),...

No obstante, esta característica se podría desglosar en una característica por cada preferencia, o bien se podría clonar y para cada preferencia clonada, en el momento de la configuración, establecer la definición.

Una subcaracterística a destacar, dado su caracter opcional, es la `Plantilla gráfica`, que sería el punto de definición del diseño de la aplicación, de modo que, en caso de ser seleccionada en el momento de la configuración del modelo de características, se permitiría alterar el diseño de la aplicación mediante plantillas gráficas y en caso de que no se seleccionase, la aplicación tendría una configuración genérica del diseño de la interfaz gráfica.

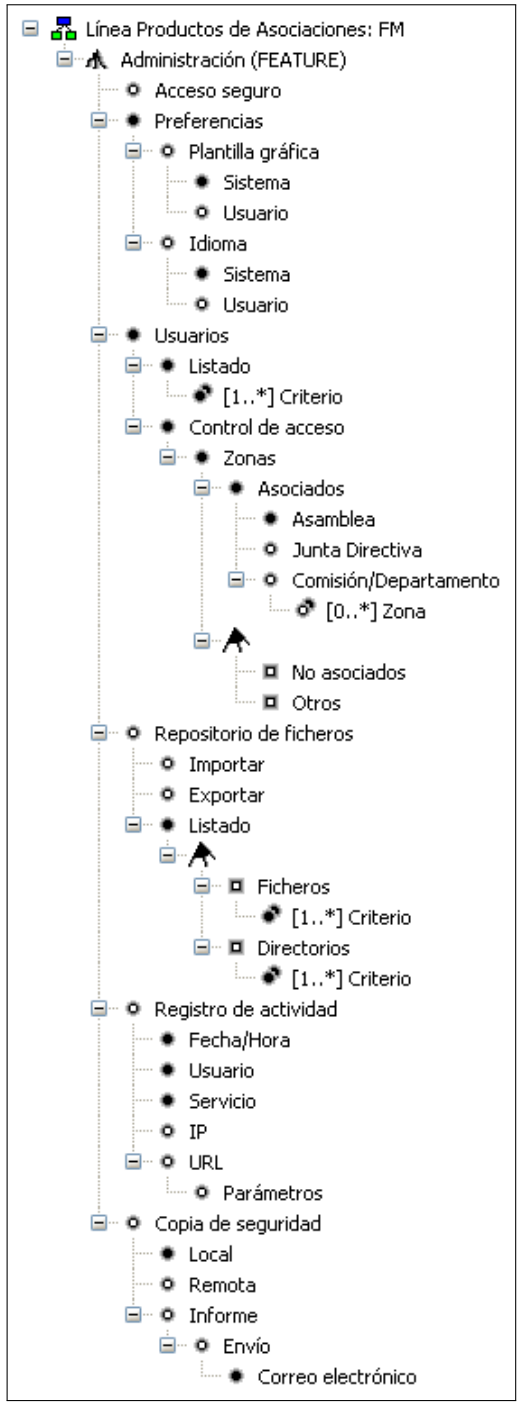


Figura 3.12: Esquema de la característica `Administración`

No hay que olvidar, que la selección de esta subcaracterística incluye la gestión del diseño de la interfaz genérica porque la característica `Sistema`, que depende de ella, es obligatoria, pero no la personalización de la interfaz gráfica para cada usuario, porque la otra característica que depende de ella, `Usuario` es opcional.

Otra subcaracterística opcional de interés es `Idioma`. En caso de ser seleccionada el administrador o administradores podrían configurar un idioma común a toda la aplicación diferente del que se aportase con la aplicación, e incluso podrían permitir a los usuarios registrados escoger el idioma que desean tener una vez hayan iniciado la sesión en la aplicación.

3.1.11.3. Usuarios

`Usuarios` se encarga principalmente de la seguridad en la aplicación para el control del acceso a las diferentes herramientas de usuario, tanto para asociados como para no-asociados. Otra de las responsabilidaddes de esta característica es la gestión de los usuarios a nivel de aplicación, es decir, permite visualizar toda la información de acceso asociada a un usuario y establecer puntos de control para resolver un problema particular de ese usuario con alguna de las herramientas o con el sistema de acceso a la aplicación.

Esta característica es obligatoria y por tanto común en toda aplicación de la línea de productos *software*, porque es indispensable una herramienta a nivel técnico para el control y gestión de los usuarios.

Además de la inclusión automática de la característica `Listado` por ser obligatoria, que a su vez incluye a la característica clonable `Criterio` con cardinalidad `[1..*]` (para obtener un listado de usuarios se exige al menos un criterio como el tipo de usuario, la fecha de registro,... para no obtener un listado completo de todos los usuarios, aunque la cardinalidad permite que haya un único criterio y sea "todos los usuarios").

Otra subcaracterística obligatoria es `Control de acceso` que lleva a cabo toda la gestión del control de acceso a la aplicación de todos los usuarios desde el punto de vista de las herramientas que se pueden usar y la información a la que se puede acceder.

El acceso se llevará a cabo mediante el establecimiento de zonas de acuerdo al tipo de usuario. Como mínimo habrá un acceso para asociadoss (representado mediante la subcaracterística obligatoria `Asociados`) y un acceso para no asociados, aunque podrá ampliarse a un tercer grupo.

Además del acceso de todos los socios a las diferentes herrramientas, es posible asignar una zona exclusiva para los miembros de la Junta Directiva ya que puede ocurrir que para determinadas herramientas se deba restringir su uso a este grupo de usuarios) o incluso a alguna comisión o departamento, y en este último caso, se podrán escoger con cardinalidad `[0..*]` las zonas de acceso, que estarán relacionadas con el trabajo que se desarrolle en la comisión o departamento correspondiente.

La creación de zonas tiene sentido cuando se desee permitir el uso de la aplicación por parte de no-asociados o se desee realizar alguna distinción entre los propios asociados (en este caso, simplificado en las características `Asamblea`, `Junta Directiva` y `Comisión/Departamento`).

3.1.11.4. Repositorio de ficheros

Con frecuencia, en una asociación es habitual almacenar información en ficheros, y puede ser necesario llevar un seguimiento mediante algún método no manual. En vez de acceder al sistema de

ficheros y tener que buscarlas mediante comandos, puede resultar más fácil utilizar una interfaz que, aunque será más limitada que un sistema operativo, se concentrará en aquellas tareas más habituales y simplificará su ejecución.

Por este motivo se incluye la característica opcional `Repositorio de ficheros`, que en caso de ser seleccionada en el modelo de decisión, incluye de forma automática uno o dos listados posibles: listados de ficheros o de directorios, los cuales a su vez se realizan mediante algún criterio (por ejemplo, por fecha de almacenamiento, usuario que creó el fichero,...), siendo dicho criterio la característica clonable `Criterio` con cardinalidad `[1..*]`.

No se activa la obligatoriedad de la importación o exportación de un fichero, representadas mediante las subcaracterísticas `Importar` y `Exportar` respectivamente, porque toda aplicación se ha de ejecutar sobre un sistema operativo, que necesariamente debe poder acceder al sistema de ficheros, por lo que es posible realizar esas operaciones en el sistema operativo directamente. Sin embargo, esto implica tener acceso al sistema operativo en donde el control de privilegios es diferente, y es posible que no se desee proporcionar a un usuario cualquier esta clase de privilegios.

3.1.11.5. Registro de actividad

Es posible, pero no imprescindible, que los usuarios administradores de la aplicación necesiten saber qué se hace con la aplicación, por parte de quién y cuándo, e incluso la información tratada. Para ello, se modela la característica opcional `Registro de actividad` que se encarga de guardar en un registro la información modelada mediante las subcaracterísticas obligatorias: `Fecha/Hora`, `Usuario` y `Servicio`; y opcionalmente la cacterística opcional `IP` para tener constancia del lugar de origen de la actividad realizada y `URL` para saber con exactitud el lugar de la aplicación al que se ha accedido (a su vez, y opcionalmente, es posible conocer los parámetros de la URL y por consiguiente la información transmitida).

Esta característica será de especial utilidad durante la fase de desarrollo de la línea de productos, ya que permite hacer un seguimiento exhaustivo de todo lo que se hace con la aplicación; por lo tanto, aunque sea de utilidad para los administradores una vez implantada la aplicación, porque les puede permitir encontrar el origen de un fallo, lo es también para los desarrolladores, y por tanto, la selección de esta característica también forma parte de las decisiones tomadas en la configuración de la fase de implementación.

3.1.11.6. Copia de seguridad

Todo sistema de gestión ha de realizar copias de seguridad de la información, ya que la pérdida de la información no tiene valor económico. Sin embargo, hoy en día existen diversas soluciones, de modo que no es necesario que una aplicación *software* incluya un sistema que permita hacer esto; sin embargo, en la línea de productos *software* para asociaciones se incluye `Copia de seguridad` como característica opcional.

En caso de seleccionar la copia de seguridad se incluye por defecto la posibilidad de hacer copias de seguridad locales (se almacenan en el mismo sistema que ejecuta la aplicación, como un fichero). También es posible hacer copias remotas (lo más óptimo para garantizar la integridad de la información), si bien esta posibilidad no siempre es posible por parte del cliente de la aplicación y por eso se ha modelado como una característica opcional.

Durante la copia de seguridad se puede elaborar un informe de todos los elementos de la copia de seguridad, modelado mediante la subcaracterística opcional `Informe`, que incluye opcionalmente el envío por correo electrónico.

La copia de seguridad debe entenderse que se realiza de todo el sistema, es decir, de la información contenida en la base de datos y de los ficheros almacenados. También es importante saber que, si bien la decisión sobre la subcaracterística `Remota` se puede hacer durante la fase de desarrollo, puede ser una decisión en tiempo de ejecución, y por lo tanto la característica estará incluida en el producto *software* siendo el usuario quien la active o no (aunque obligatoriamente siempre se guardará en el sistema local independientemente de que se escoja hacer la copia de seguridad remota o no).

3.1.11.7. Restricciones

	CARACTERÍSTICA(S) A	◄►	CARACTERÍSTICA(S) B
	RELACIÓN **CONTEXTO**		
≽	`CopiaSeguridad` (3.1.11.6)	◄►	`AccesoSeguro` (3.1.11.1)
	A → B	Es recomendable acceder a la herramienta de copia de seguridad mediante un acceso seguro pero no es imprescindible	
	B ↛ A	El acceso seguro no tiene implicación alguna en la selección de la característica `CopiaSeguridad`	
≽	`Local` (3.1.11.6)	◄►	`Exportar` (3.1.11.4)
	A → B	Es útil disponer de una extracción a través de la aplicación, de los ficheros de la copia de seguridad local	
	B ↛ A	La exportación de ficheros no tiene ninguna dependencia de la subcaracterística `Local` de `CopiaSeguridad`	
	`Envio` (3.1.11.6)	◄►	`CorreoElectronico` (3.1.8.1)
	A ⇒ B	Un informe no se puede enviar por correo electrónico si no existe dicha posibilidad en la característica `EnvioMensajes`	
	B ↛ A	El envío por correo electrónico sólo depende de la existencia de un servicio de transporte de correo electrónico, que queda fuera de este modelo de características	

Cuadro 3.12: Restricciones adicionales de la característica `Administración`

3.2. Validez del modelo de características

El modelo de características presentado en el apartado previo no es un modelo cerrado ni definitivo. Se podría decir que es una primera versión perfectamente criticable y discutible en muchos puntos porque no es lo mismo la visión de un equipo de desarrollo que la de un único desarrollador, y como ya se ha comentado en alguna ocasión, sucesivas lecturas mejorarán la caracterización de la precisión, exactitud y variabilidad de la línea de productos *software* en el modelo de características; especialmente cuando esa lectura es realizada por varias personas que tengan visiones diferentes sobre el tema (por ejemplo, que pertenezcan a asociaciones diferentes).

Por lo tanto, es necesario conocer la validez del modelo de características analizado en el apartado anterior, tanto desde el punto de vista del propio modelado de las características y las restricciones existentes entre ellas como la modificación del mismo, pues uno de los aspectos más importantes de una línea de productos es la capacidad de mejora de la misma en base a los productos obtenidos, con lo que la modificación de la línea es inherente al proceso[14] de vida de la línea.

El análisis del modelo de características se ha validado contra posibles usuarios ajenos al desarrollo. Si bien todos aquellos usuarios potenciales dieron por válido el modelo, la mayoría solía hacer comentarios sobre necesidades personales que a su juicio no estaban contempladas. Sin embargo, después de una revisión, prácticamente todas esas necesidades estaban incluidas en algún punto del modelo o bien .

Un problema que se ha detectado cuando se ha realizado esta validación es la falta de concreción de los usuarios en la crítica al modelo, ya que muchos de ellos no pensaban en lo que necesitaban en el modelo sino que se centraban en detalles de aquello que no estaba contemplado pero que no permitía identificar una funcionalidad determinada. Esto se debió a que los usuarios una vez visto el modelo intentaban aportar las sugerencias en términos del modelo en lugar de hacerlo con sus propias palabras.

En cualquier caso, la validación con los futuros usuarios potenciales fue exitosa porque aportó nuevas funcionalidades a tener en cuenta, y permitió encontrar fallos de modelado.

El modelo de características propuesto y descrito es ampliable, aunque cualquier ampliación podrá hacerse en unas ocasiones extendiendo una rama de características en el punto que proceda (por ejemplo, añadiendo unas subcaracterísticas a una característica de grupo existente), en otras ocasiones habrá que hacerlo mediante restricciones (por ejemplo, porque haya que considerar nuevas reglas a tener en cuenta), o incluso combinando ambas.

Lo habitual consistirá en tener que añadir nuevas características y a su vez esto llevará a tener que verificar las necesidades de estas nuevas características en el modelo, pues puede que alguna de esas necesidades sea una característica opcional (y por tanto, se va a generar una restricción) o bien no exista una característica que modele esa nueva necesidad, por lo que tendrá que modelarse también.

Por ejemplo, en el diagrama de características de la rama `Comunicación`, representado en la ilus-

[14] Quizás en el desarrollo basado en líneas de productos *software* esté más justificado el coste tanto económico como temporal que supone el desarrollo iterativo e incremental porque no se está desarrollando un producto concreto sino una familia de productos, con lo que la precisión que se obtiene en los primeros productos de la línea no es la misma que en productos posteriores, y por tanto, la mejora de la línea está sustentanda, en gran parte, por las opiniones y correciones de errores de productos ya obtenidos (es lo que se denomina como *feedback* o retroalimentación)

tración de la figura 3.9, es posible que una asociación determinada plantee la posibilidad de incluir el envío de mensajes de texto a través de la telefonía móvil (SMS). Añadiendo esta funcionalidad al modelo de características, el esquema de dicha rama quedaría como se muestra en la ilustración de la figura 3.13.

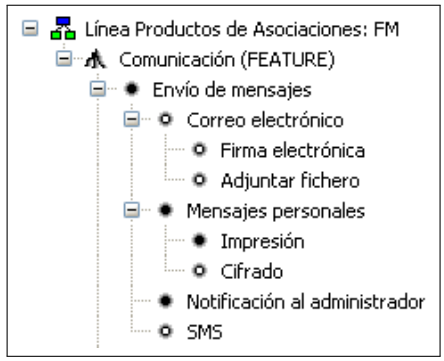


Figura 3.13: Esquema parcial de la característica `Comunicación` modificado

No obstante, este ejemplo de ampliación va a dar lugar a más modificaciones, posiblemente en la rama de la característica `Administración`, ya que se querrá llevar un control de quien puede enviar mensajes de texto y quien no. Y puesto que el servicio de mensajes de texto no va a ser gratuito para la asociación, los gastos asociados a estos mensajes podrán ser incluidos en el libro de cuentas de forma automática.

Muchos de estos comentarios se pueden modelar mediante características y restricciones, pero hay que tener en cuenta que aunque una ampliación del modelo de características pueda parecer que no afecte al resto de características, si no se hace un estudio de la influencia de las nuevas características en el modelo, puede ocurrir que no sea posible implementar el componente o conjunto de componentes que habrán de proporcionar la funcionalidad incluida en las características añadidas porque no se haya hecho una análisis adecuado.

3.3. Simplificación del modelo de características

Si bien el modelo de características analizado sería el modelo de decisión de la línea de productos *software*, el análisis que ha de realizarse posteriormente, como en todo desarrollo *software*, para obtener los diferentes elementos que, mediante la línea de productos, permitirán obtener el producto concreto de acuerdo a una configuración elegida, es una tarea que queda fuera de este Proyecto Fin de Carreradebido a la envergadura que supone, tanto en medios materiales como humanos, por la gran cantidad de funcionalidades representadas en el modelo de características.

En el capítulo siguiente se presentará un análisis de una versión simplificada del modelo de características explicado en la primera parte de este capítulo, y por tanto del dominio de análisis, que se considera suficientemente representativa para esta línea de productos *software*, con el objeto de mostrar la validez del uso de los modelos de características como elemento de representación de la variabilidad de un desarrollo *software* basado en líneas de producto.

Capítulo 4

Construcción de la línea de productos para asociaciones

"El que con perspicacia reconoce la limitación de sus facultades, está muy cerca de llegar a la perfección"

Johann Wolfgang von Goethe

4.1. Modelo de características simplificado

En el capítulo 3, se realizó un estudio pormenorizado del modelo de características de una línea de productos para asociaciones, aunque ya se indicó que en este capítulo se haría uso de una versión simplificada.

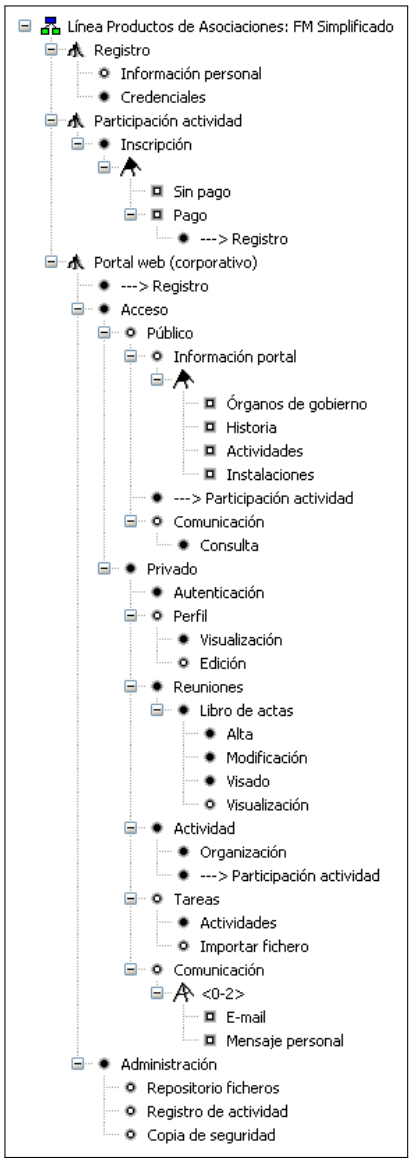
El modelo de características que servirá de base es el ilustrado en la figura 4.1. En esta versión simplificada se intentan juntar las características y funcionalidades imprescindibles en todo producto de una línea de productos para asociaciones, es decir, se presenta, mayoritariamente, características de carácter obligatorio, si bien se han reorganizado muchas características que en el modelo de características del capítulo anterior se ilustraban con más detalle y extensión, así como toda su variabilidad posible.

Junto a la ilustración del modelo de características, también se muestra un extracto (en el que se han eliminado nodos para reducir la extensión del mismo) de lo que sería la representación en XML de dicho modelo. Esta representación se ha generado con el mismo *plug-in* del entorno de desarrollo *Eclipse* con el que se ha realizado el modelo de características: *fmp* (Feature Model Plug-In).

Hay que tener en cuenta que este modelo de características (o cualquier otro), necesita de una selección en la configuración de modo que se eliminen todos los puntos de decisión que no pertenezcan al tiempo de ejecución, pues estos últimos son tomados por el usuario que utiliza la aplicación.

En el caso del modelo de características ilustrado en este capítulo se ha reducido la variabilidad para reducir la complejidad, ya que, el número de productos que la línea de productos debe poder generar, se multiplica por dos para cada característica opcional.

Por este motivo, se describirá brevemente el modelo de características simplificado realizando refe-



(a) Modelo de características

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Extracto modelo características simplificado -->

<feature min="1" max="1" name="Portal web (corporativo)"
  id="portalWeb">
  <feature min="1" max="1" name="Registro"
    id="reference1IXIregistro">
    <feature min="0" max="1" name="Información personal"
      id="reference1IXIinformacionPersonal">
    </feature>
    <feature min="1" max="1" name="Credenciales"
      id="reference1IXIcredenciales">
    </feature>
  </feature>
  <feature min="1" max="1" name="Acceso" id="informacion">
    <feature min="0" max="1" name="Público" id="publico">
      <feature min="0" max="1" name="Información portal"
        id="informacionPortal">
        <featureGroup min="1" max="4" id="group0">
          <!-- ... -->
          <feature min="0" max="1" name="Instalaciones"
            id="instalaciones">
          </feature>
        </featureGroup>
      </feature>
      <reference min="1" max="1" id="reference">
      </reference>
      <!-- ... -->
    </feature>
    <feature min="1" max="1" name="Privado" id="privado">
      <!-- ... -->
      <feature min="1" max="1" name="Reuniones"
        id="reuniones">
        <!-- ... -->
      </feature>
      <feature min="1" max="1" name="Actividad"
        id="actividad">
        <!-- ... -->
      </feature>
      <feature min="0" max="1" name="Tareas" id="tareas">
        <!-- ... -->
      </feature>
      <feature min="0" max="1" name="Comunicación"
        id="comunicacion0">
        <!-- ... -->
      </feature>
    </feature>
  </feature>
  <feature min="1" max="1" name="Administración"
    id="administracion">
    <!-- ... -->
  </feature>
</feature>
```

(b) Extracto de la representación en XML

Figura 4.1: Modelo de características, y representación en XML, del sistema simplificado

rencias al modelo de características completo donde proceda ya que puede que haya ciertas asunciones hechas en esta versión simplificada que de no ser explicadas sean malinterpretadas, y como ya se hizo incapié en el anterior capítulo, tan importante es un buen modelo de características que represente la variabilidad de una línea de productos como la descripción del mismo para la correcta toma de decisiones.

4.1.1. Resumen de la descripción

Se partirá de la característica raíz `Portal web`, ya que las características `Registro` y `Participación actividad` se han modelado por separado para evitar la repetición en el modelo pues se usan varias veces, y por ese motivo cuando aparezca la flecha ($--$ >) seguida del nombre de la característica correspondiente bastará con integrar la característica referenciada junto con el resto de subcaracterísticas que dependan de ella en ese punto.

El portal *web* está dividido en tres características obligatorias:

- **Registro**: Proporciona la funcionalidad necesaria para poder registrarse en el portal, si bien, el registro no dará acceso automático al portal *web*, ya que la cuenta deberá ser validada por el administrador del portal o persona autorizada dependiendo del registro por la vía de entrada a la opción de registro (por ejemplo, puede que haya seleccionado visualizar información de una actividad con coste y haya decidido inscribirse, lo cual provoca que deba registrarse, o simplemente puede que haya decidido inscribirse en la asociación).

  Esta característica es una referencia a la rama encabezada por la característica `Registro`.

- **Acceso**: Discriminará entre la información pública cuya visualización y acceso no requiere registro alguno por parte del usuario del portal y la información privada de la asociación, la cual requiere que el usuario no sólo se haya registrado sino que haya sido autorizado e incluido en algún grupo (asociado, asistente a actividad,...) de usuarios válidos.

- **Administración**: Funcionalidad encargada de garantizar el correcto funcionamiento de todas las herramientas del portal *web*.

La característica `Registro` está explicada en el apartado 3.1.2. Aunque la subcaracterística `Obligación de registro` se integra en el propio diagrama simplificado de modo que se asume que el registro estará disponible para las actividades que tengan un coste y la inscripción en la asociación.

La subcaracterística optativa `Preferencias personales` se omite, y la subcaracterística obligatoria `Mensajería` se integra en el diagrama simplificado en la subcaracterística `Acceso.Privado.Comunicación`.

La característica `Administración` está explicada en el apartado 3.1.11. Se omiten las subcaracterísticas `Acceso seguro` y `Preferencias`. La subcaracterística `Usuarios` se integra en el diagrama simplificado unica y exclusivamente en el control de acceso por zonas no siendo necesaria la obtención de listados.

La característica `Acceso` está también explicada en el apartado 3.1.1. Las subcaracterísticas `Información pública` y `Acceso.No asociado` se han integrado en el diagrama simplificado en la subcaracterística `Público`. La subcaracterística del diagrama simplificado `Comunicación.Consulta` está explicada en el apartado 3.1.8.2.

La subcaracterística `Reuniones` del diagrama simplificada se explica en el apartado 3.1.3, aunque la funcionalidad de interés es la que se explica en el apartado 3.1.3.1.

La característica `Actividad` del diagrama simplificado se describe en el apartado 3.1.4, aunque las subcaracterísticas de interés son: `Información actividad` (3.1.4.1), `Inscripción` (3.1.4.2)y `Pago` (3.1.4.2).

Las subcaracterísticas opcionales `Tareas` y `Comunicación` del diagrama simplificado se explican en los apartados 3.1.5 y 3.1.8 respectivamente.

No obstante, la característica `Tareas` sólo tendrá la funcionalidad necesaria para la organización de una actividad en tareas, por eso se ha modelado la subcaracterística `Tareas.Actividad`.

4.2. Casos de uso

El proceso de elaboración de los casos de uso, se ha basado en el modelo de características, y se ha omitido la especificación completa de todos los casos de uso mediante plantillas ya que en la descripción del modelo de características realizada en el capítulo 3 se deja bastante claro toda la interacción entre el usuario y el sistema, y donde no se indica expresamente es porque se considera trivial.

Sin embargo, para la obtención de los casos de uso, así como el diagrama de casos de uso, se ha aplicado la propuesta [31] realizada por Laguna y otros, descrita en el apartado 2.3, aunque con una pequeña variante.

Una vez identificados todos los casos de uso y actores dentro del dominio, se han establecido las relaciones existentes entre ellos de la manera tradicional. A continuación (y es aquí donde está la variante), en lugar de organizar todos esos casos de uso en paquetes, tal y como proponen Laguna y otros, se ha optado por realizar el diagrama de casos de uso completo para luego organizarlo en paquetes teniendo en cuenta las dependencias entre las características del modelo de características simplificado. El motivo de haber optado por esta solución se debe a la inexperiencia del autor de este Proyecto Fin de Carrera en la aplicación de esta propuesta. Por lo tanto, resultó más fácil obtener una visión general de todos los casos de uso y actores implicados para luego "trocearla" y así obtener la jerarquía de paquetes de casos de uso. Los paquetes[1] obtenidos fueron los siguientes:

- **PBase**: Paquete base. Contiene las siguientes funcionalidades:
  - Administración del sistema
  - Registro, inicio y autenticación de sesión
  - Gestión del libro de actas
  - Gestión de actividades
- **PPublico**: Registro e inscripción en actividades
- **PPublicoInformacionPortal**: Información del portal *web*
- **PPublicoComunicacion**: Consulta (general) desde el portal *web*
- **PTareas**: Organización y gestión de tareas para actividades
- **PTareasFichero**: Importación de ficheros para las tareas
- **PPerfil**: Visualización del perfil personal
- **PPerfilVisualizacion**: Edición del perfil personal
- **PReunionesVisualizacion**: Modificación de un acta
- **PComunicacion**: Envío de mensajes
- **PComunicacionEmail**: Envío de mensajes por correo electrónico

[1] Cuando se hable de paquete global deberá entenderse como la funcionalidad ofrecida por la combinación de los paquetes a los que se haga referencia teniendo presente el modelo de características y sus restricciones

- **PComunicacionInterno**: Envío de mensajes personales
- **PCopiasSeguridad**: Gestión de las copias de seguridad
- **PRegistroActividad**: Gestión del registro de actividad del sistema
- **PRepositorioFicheros**: Gestión del repositorio de ficheros
- **PInformacionPersonal**: Inclusión de información personal extra durante el registro

4.2.1. Diagramas de casos de uso

La ilustración de la figura 4.2 muestra la jerarquía de paquetes de casos de uso, de acuerdo a la propuesta [31] de Laguna y otros, mientras que la ilustración de la figura 4.3 muestra el diagrama de casos de uso obtenido en primer lugar y posteriormente "troceado".

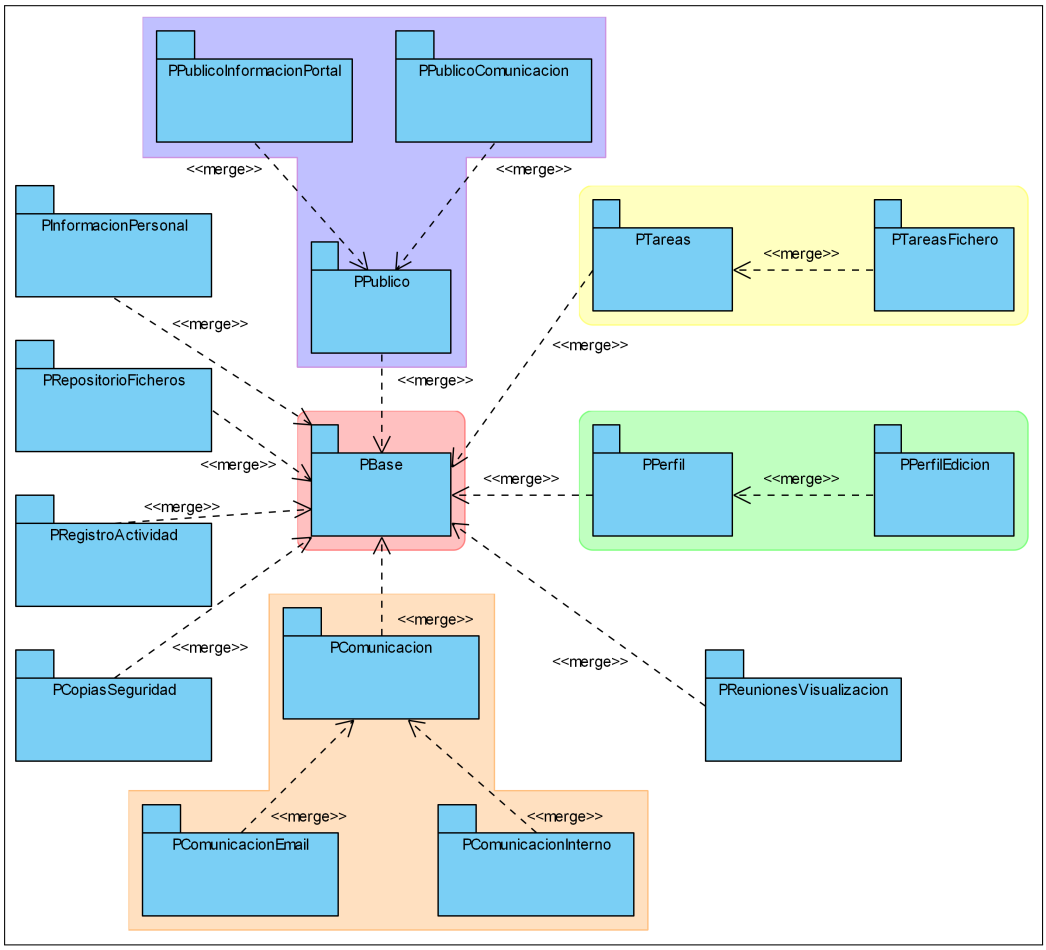


Figura 4.2: Esquema de la jerarquía de los casos de uso en paquetes

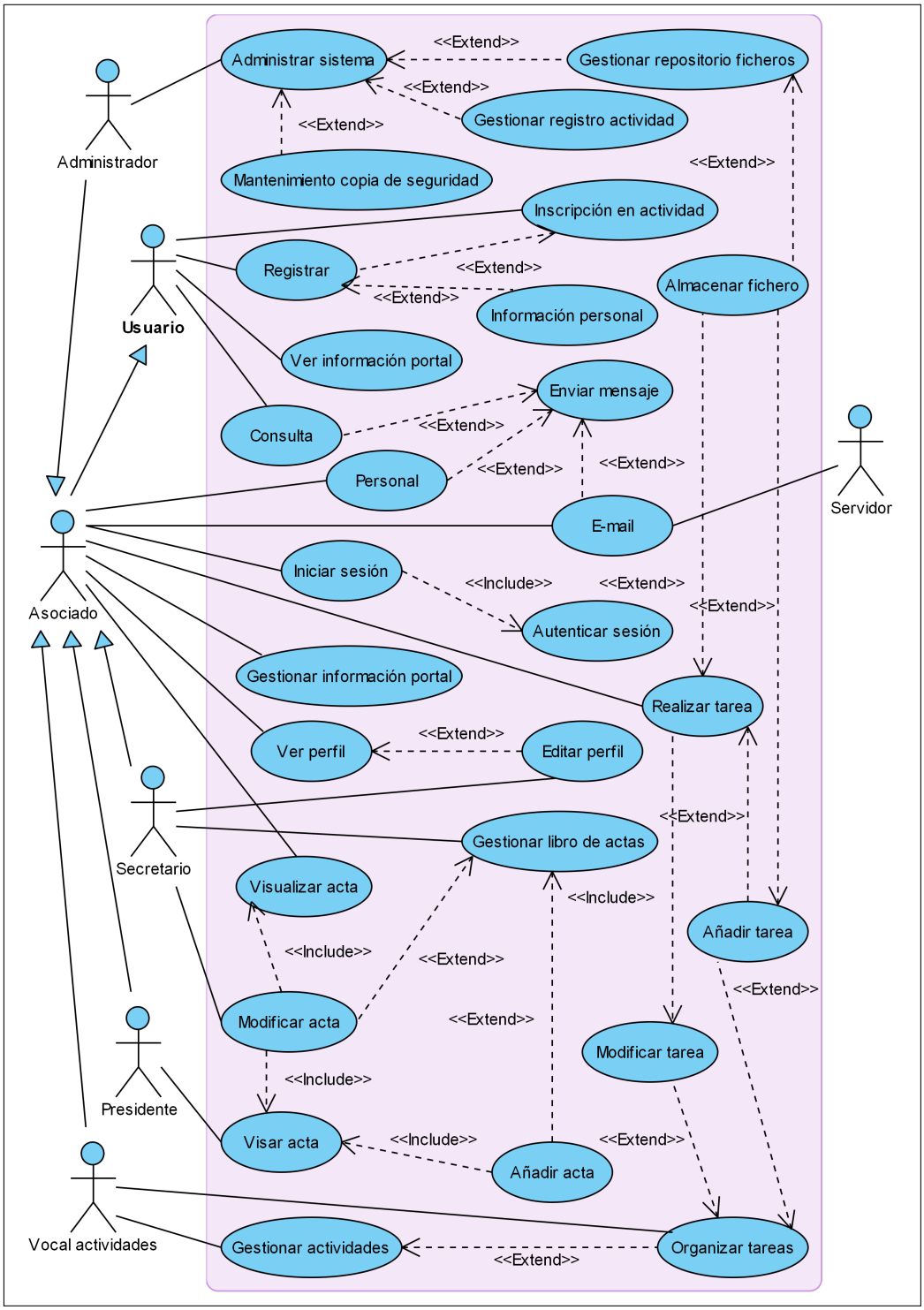


Figura 4.3: Diagrama de casos de uso del sistema simplificado

4.2.1.1. Paquete base

El paquete base reune el conjunto de características obligatorias de primer nivel y cualquier otra subcaracterística "hija" también obligatoria. Las características incluidas son las siguientes:

- `Registro`, `Registro.Credenciales`
- `Acceso`, `Acceso.Privado`, `Acceso.Privado.Autenticacion`, `Acceso.Privado.Reuniones`, `Acceso.Privado.Reuniones.LibroActas`, `Acceso.Privado.Reuniones.LibroActas.Alta`, `Acceso.Privado.Reuniones.LibroActas.Modificacion`, `Acceso.Privado.Reuniones.LibroActas.Visado`, `Acceso.Privado.Actividad`, `Acceso.Privado.Actividad.Organizacion`, `Acceso.Privado.Actividad.ParticipacionActividad`, `Acceso.Privado.Actividad.ParticipacionActividad.Inscrip-cion`
- `Administracion`

La ilustración de la figura 4.4 muestra el diagrama de casos de uso del paquete `PBase`.

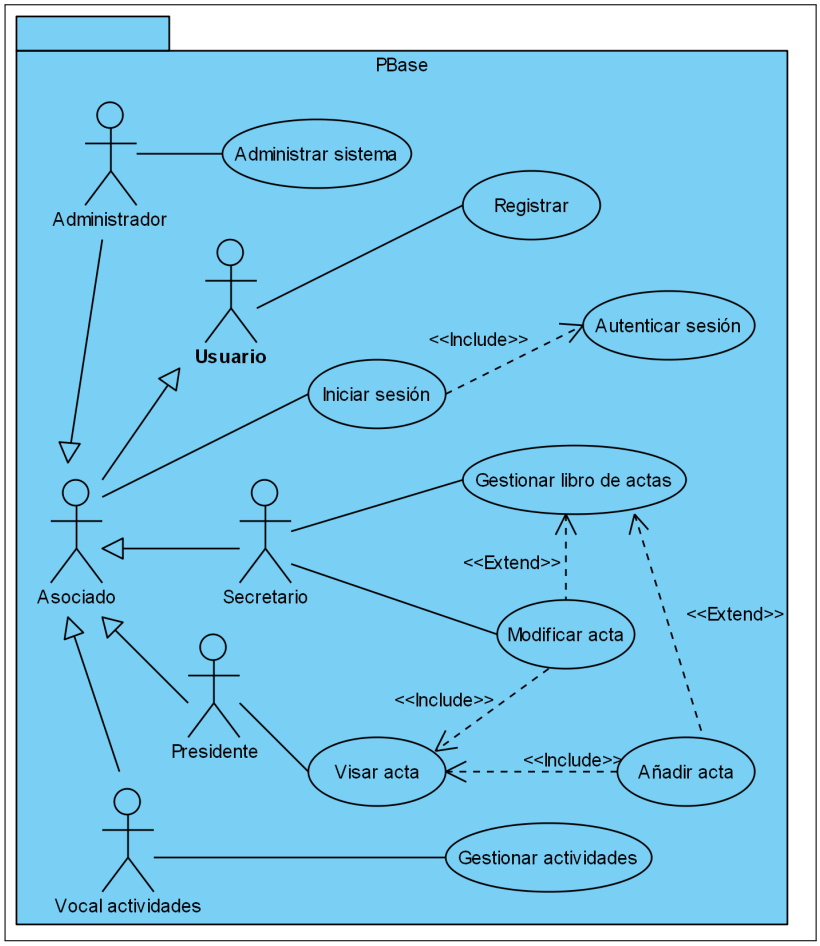


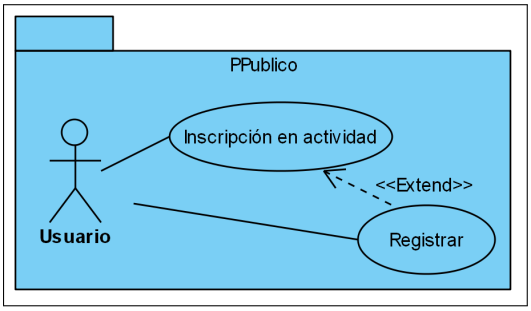
Figura 4.4: Jerarquía de casos de uso del paquete `PBase`

4.2.1.2. Paquetes público, información y comunicación del portal

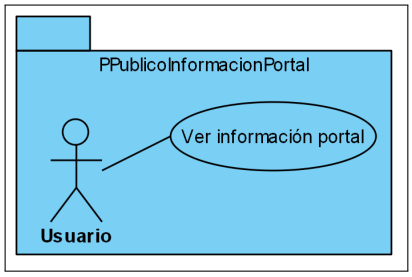
El paquete público, `Acceso.Publico`, incluye el conjunto de características que permiten inscribirse en una actividad, ver la información (estática) que se haya publicado en el portal *web* y enviar mensajes de consulta a la asociación. Las características incluidas son las siguientes:

- `Acceso.Publico.ParticipacionActividad`
- `Acceso.Publico.InformacionPortal`
- `Acceso.Publico.Comunicacion`, `Acceso.Publico.Comunicacion.Consulta`

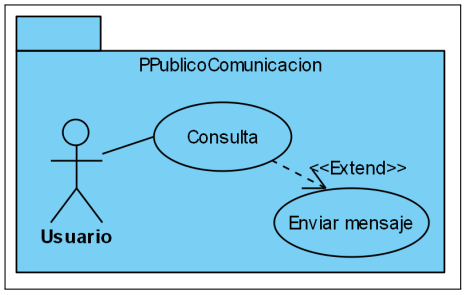
Las ilustraciones de la figura 4.5 muestran los diagramas de casos de uso de los paquetes `PPublico`, `PPublicoInformacionPortal` y `PPublicoComunicacion` respectivamente.



(a) PPublico



(b) PPublicoInformacionPortal



(c) PPublicoComunicacion

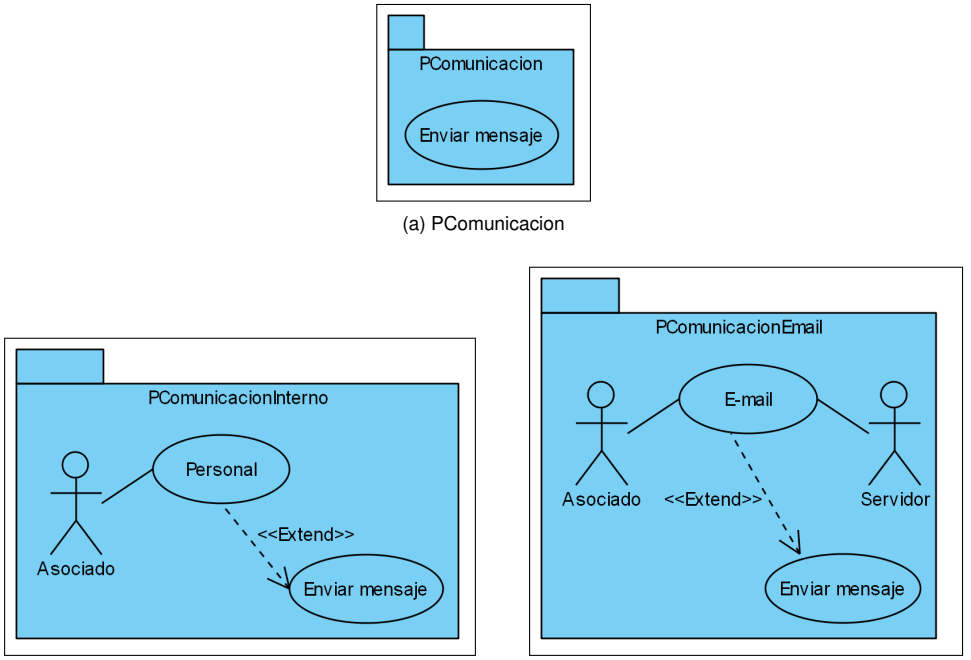
Figura 4.5: Jerarquía de casos de uso de los paquetes `PPublico`, `PPublicoInformacionPortal`, `PPublicoComunicacion`

4.2.1.3. Paquetes comunicación por mensajería personal y correo electrónico

El paquete global comunicación, `Acceso.Privado.Comunicacion`, incluye el conjunto de características que permiten comunicarse dentro de la aplicación mediante mensajes personales y también mediante correo electrónico. Nótese que la característica `Comunicación` es una característica de grupo de tipo XOR. Las características incluidas son las siguientes:

- `Acceso.Privado.Comunicacion.E-mail`
- `Acceso.Privado.Comunicacion.MensajePersonal`

Las ilustraciones de la figura 4.6 muestran los diagramas de casos de uso de los paquetes `PComunicacion`, `PComunicacionInterno` y `PComunicacionE-mail` respectivamente.



(a) PComunicacion

(b) PComunicacionInterno

(c) PComunicacionEmail

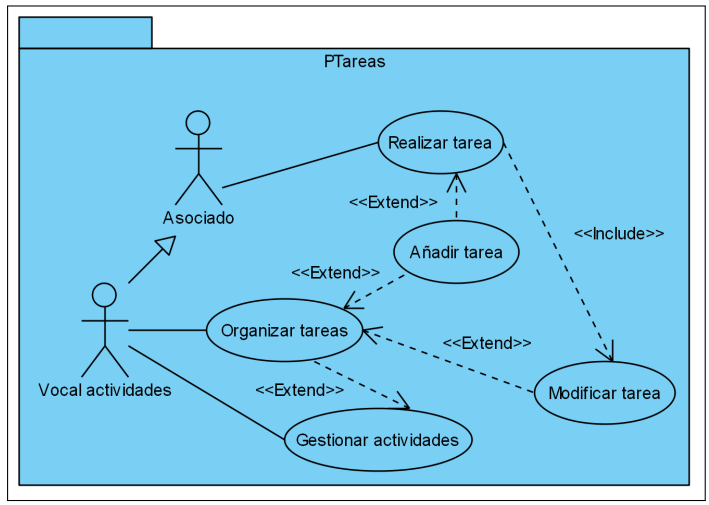
Figura 4.6: Jerarquía de casos de uso de los paquetes `PComunicacion`, `PComunicacionInterno` y `PComunicacionE-mail`

4.2.1.4. Paquetes tareas e inclusión de ficheros en tareas

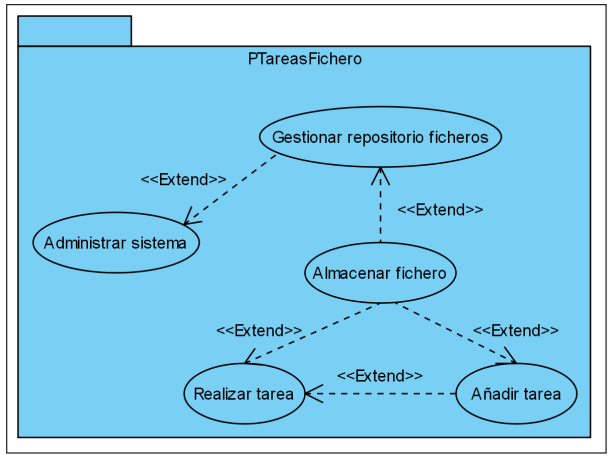
El paquete global tareas, `Acceso.Privado.Tareas`, incluye el conjunto de características que permite gestionar las tareas de una actividad e incluir ficheros para las tareas. La selección de `Tareas` incluye a `Actividades`, pero no a `Importar fichero`. En la jerarquía de paquetes, ésta última se encuentra en el paquete `PTareasFichero`. Las características incluidas son las siguientes:

- `Acceso.Privado.Tareas.Actividades`
- `Acceso.Privado.Tareas.ImportarFichero`

Las ilustraciones de la figura 4.7 muestran los diagramas de casos de uso de los paquetes `PTareas` y `PTareasFichero` respectivamente.



(a) PTareas



(b) PTareasFichero

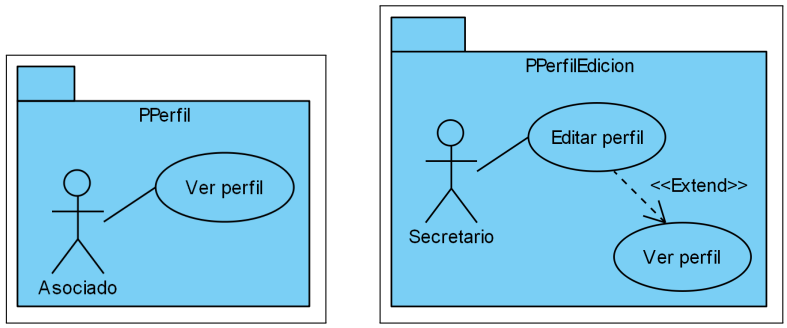
Figura 4.7: Jerarquía de casos de uso de los paquetes `PTareas` y `PTareasFichero`

4.2.1.5. Paquetes perfil y edición del perfil

El paquete global perfil, `Acceso.Privado.Perfil`, incluye el conjunto de características que permiten visualizar y editar el perfil personal. La selección de `Perfil` incluye a `Visualización`, pero no a `Edición`. En la jerarquía de paquetes, ésta última se encuentra en el paquete `PPerfilEdicion`, de modo que la inclusión de la característica `Edición` lleva a que el paquete `PPerfilEdicion` se combine mediante la operación `<<merge>>` con el paquete `PPerfil`. Las características incluidas son las siguientes:

- `Acceso.Privado.Perfil.Visualizacion`
- `Acceso.Privado.Perfil.Edicion`

Las ilustraciones de la figura 4.8 muestran los diagramas de casos de uso de los paquetes `PPerfil` y `PPerfilEdicion` respectivamente.



(a) PPerfil (b) PPerfilEdicion

Figura 4.8: Jerarquía de casos de uso de los paquetes `PPerfil` y `PPerfilEdicion`

4.2.1.6. Paquete información personal

El paquete información personal, `Registro.InformacionPersonal`, incluye a la característica que se encarga de la funcionalidad necesaria para la recogida y almacenamiento de información añadida a los credenciales de acceso.

En caso de selección de esta característica, mediante la operación `<<merge>>` se combinará el paquete `PInformacionPersonal` con su paquete base `PBase`.

La ilustración de la figura 4.9 muestra el diagrama de casos de uso del paquete `PInformacionPersonal`.

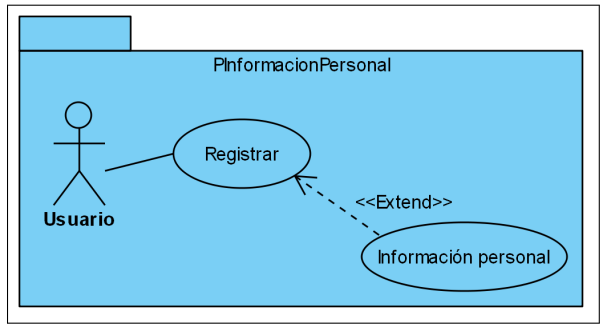


Figura 4.9: Jerarquía de casos de uso del paquete `PInformacionPersonal`

4.2.1.7. Paquete visualización de reuniones

El paquete visualización de reuniones, `Acceso.Privado.Reuniones.LibroActas.Visualizacion`, incluye a la característica que se encarga de la funcionalidad necesaria para que cualquier asociado, y no sólo aquel que ocupe el cargo de Secretario, pueda ver cualquier acta que se haya almacenado en el libro de actas.

En caso de selección de esta característica, mediante la operación `<<merge>>` se combinará el paquete `PReunionesVisualizacion` con su paquete base `PBase`.

La ilustración de la figura 4.10 muestra el diagrama de casos de uso del paquete `PReunionesVisualizacion`.

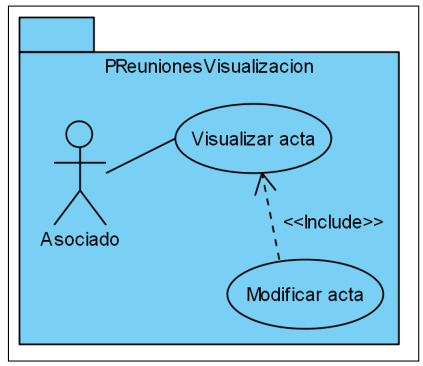


Figura 4.10: Jerarquía de casos de uso del paquete `PReunionesVisualizacion`

4.2.1.8. Paquete repositorio de ficheros

El paquete repositorio de ficheros, `Administracion.RepositorioFicheros`, incluye a la característica que se encarga de la funcionalidad necesaria para que exista un repositorio de ficheros con el que se pueda importar y exportar ficheros mediante la interfaz de la aplicación y no interactuando directamente con el sistema operativo.

En caso de selección de esta característica, mediante la operación `<<merge>>` se combinará el paquete `PRepositorioFicheros` con su paquete base `PBase`.

La ilustración de la figura 4.11 muestra el diagrama de casos de uso del paquete `PRepositorioFicheros`.

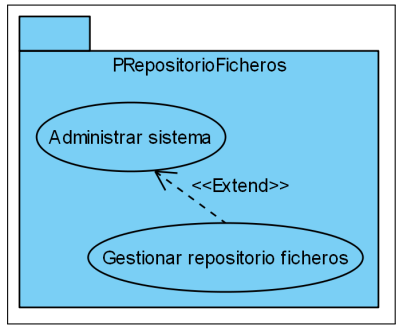


Figura 4.11: Jerarquía de casos de uso del paquete `PRepositorioFicheros`

4.2.1.9. Paquete registro de actividad

El paquete registro de actividad, `Administracion.RegistroActividad`, incluye a la característica que se encarga de la funcionalidad necesaria para poder visualizar y examinar el registro de actividad de todas las operaciones realizadas en la aplicación por cada usuario.

En caso de selección de esta característica, mediante la operación `<<merge>>` se combinará el paquete `PRegistroActividad` con su paquete base `PBase`.

La ilustración de la figura 4.12 muestra el diagrama de casos de uso del paquete `PRegistroActividad`.

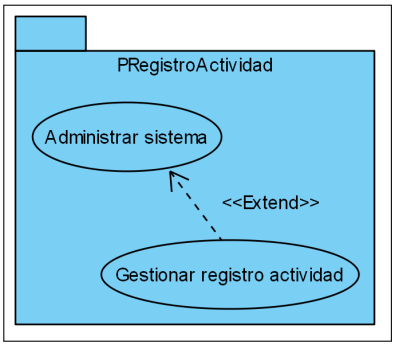


Figura 4.12: Jerarquía de casos de uso del paquete `PRegistroActividad`

4.2.1.10. Paquete copias de seguridad

El paquete copias de seguridad, `Administracion.CopiasSeguridad`, incluye a la característica que se encarga de la funcionalidad necesaria para que se pueda llevar a cabo la organización y gestión de las copias de seguridad de la aplicación mediante la interfaz de la misma.

En caso de selección de esta característica, mediante la operación `<<merge>>` se combinará el paquete `PCopiasSeguridad` con su paquete base `PBase`.

La ilustración de la figura 4.13 muestra el diagrama de casos de uso del paquete `PCopiasSeguridad`.

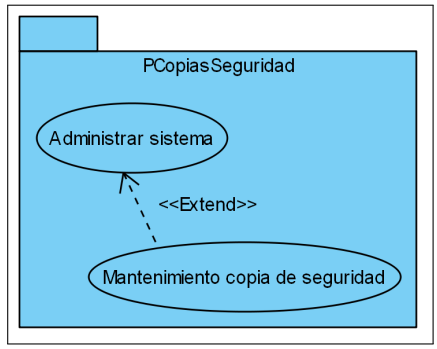


Figura 4.13: Jerarquía de casos de uso del paquete `PCopiasSeguridad`

4.3. Clases del dominio

En este último apartado del capítulo se mostrarán las clases del dominio del modelo de características simplificado aplicando de nuevo la propuesta [31] realizada por Laguna y otros.

Sin embargo, en esta ocasión, no se va a realizar la misma variación que se hizo en el apartado anterior para el modelado de los casos de uso, ya que gracias a la experiencia adquirida, y también a que la jerarquía de paquetes ya está hecha, no es necesario obtener un diagrama de clases del dominio para posteriormente "trocearlo". Además, esta otra visión de organizar las clases del dominio resulta incluso más fácil de manejar mentalmente porque al aislar la funcionalidad en paquetes, el problema también se divide y resulta más fácil atacarlo para resolverlo.

4.3.1. Diagramas de clases

La estructura que se va a seguir, va a ser la misma que en el apartado anterior. No obstante, no se va a describir los paquetes de clases ya que se ha hecho en el apartado anterior. Por lo que en caso de duda, simplemente ha de consultar la documentación para el paquete correspondiente pues el nombre del paquete no varia.

4.3.1.1. Paquete base

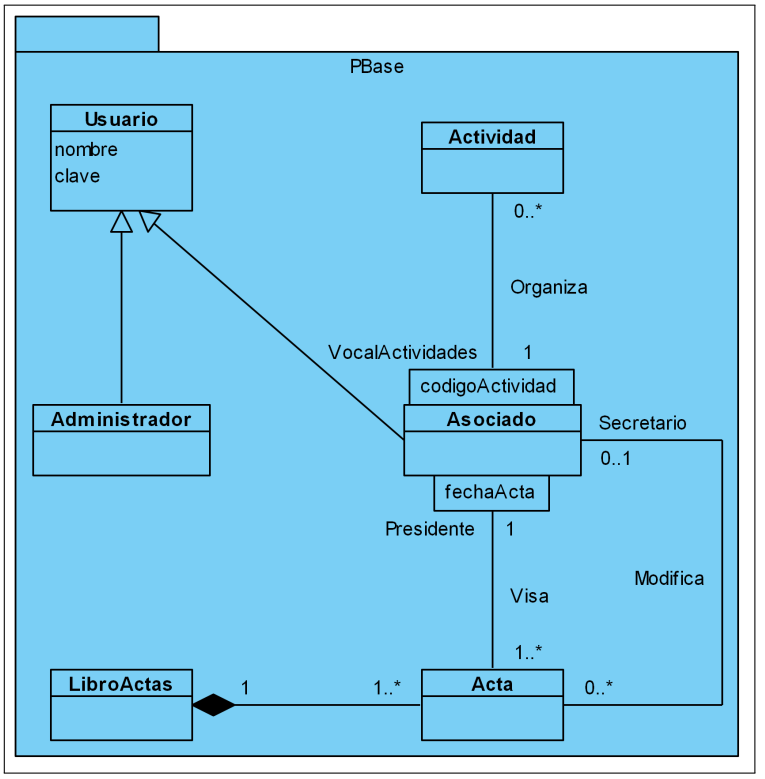
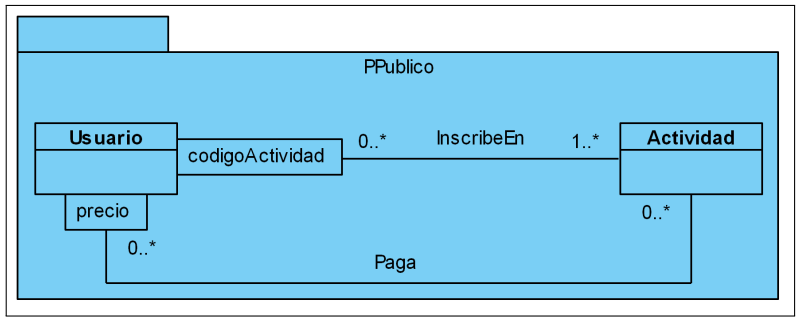
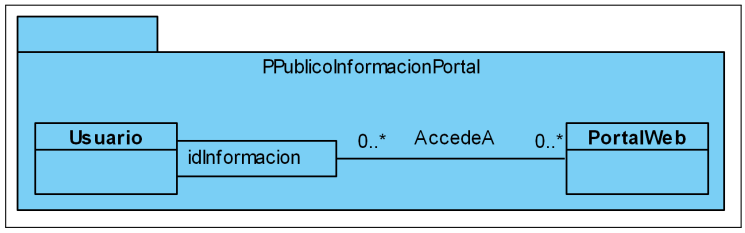


Figura 4.14: Diagrama de clases del paquete `PBase`

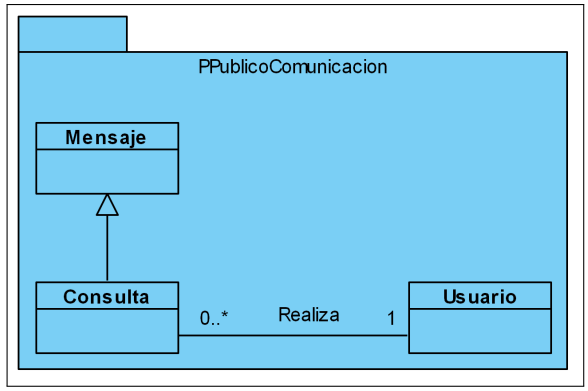
4.3.1.2. Paquetes público, información y comunicación del portal



(a) PPublico



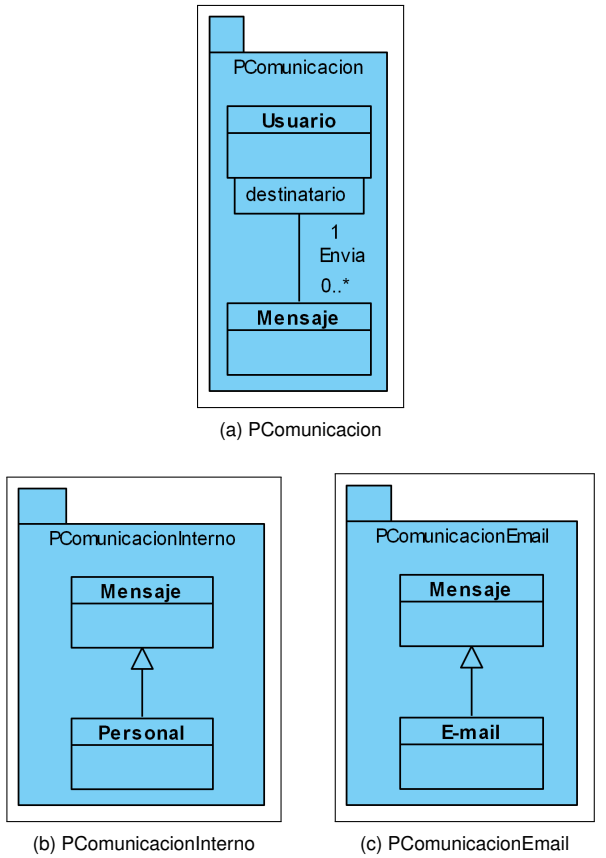
(b) PPublicoInformacionPortal



(c) PPublicoComunicacion

Figura 4.15: Diagrama de clases de los paquetes `PPublico`, `PPublicoInformacionPortal`, `PPublicoComunicacion`

4.3.1.3. Paquetes comunicación por mensajería personal y correo electrónico



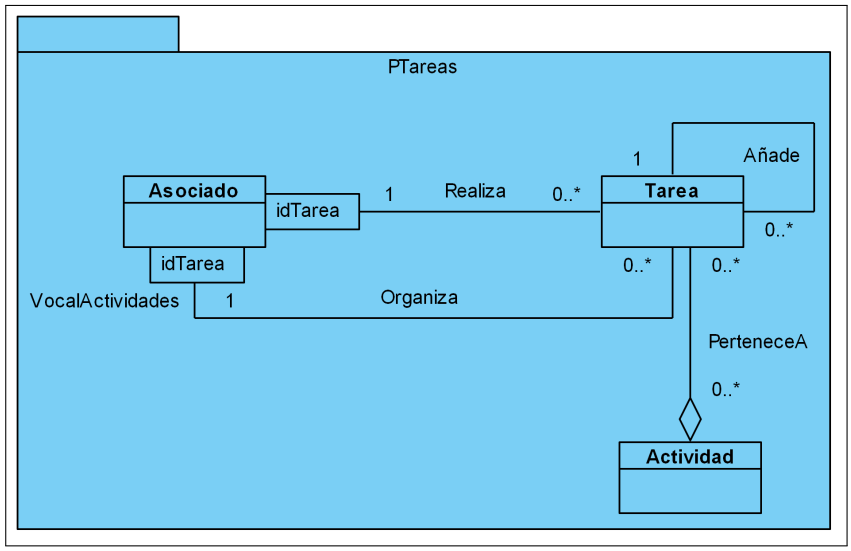
(a) PComunicacion

(b) PComunicacionInterno

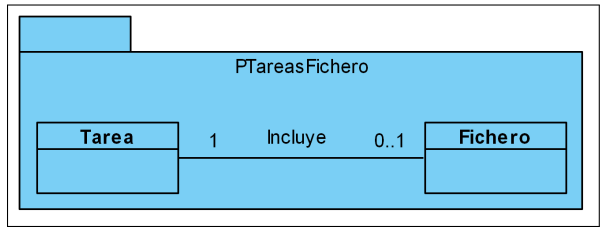
(c) PComunicacionEmail

Figura 4.16: Diagrama de clases de los paquetes `PComunicacion`, `PComunicacionInterno` y `PComunicacionE-mail`

4.3.1.4. Paquetes tareas e inclusión de ficheros en tareas



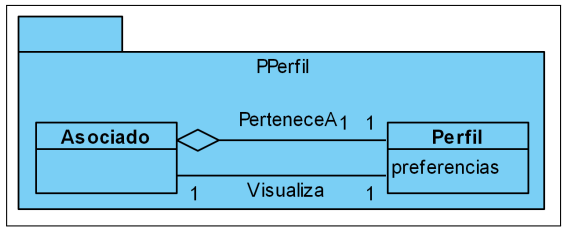
(a) PTareas



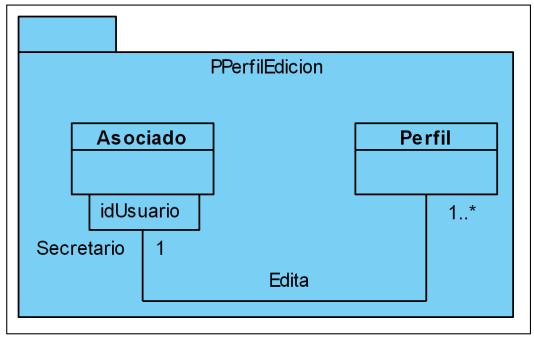
(b) PTareasFichero

Figura 4.17: Diagrama de clases de los paquetes `PTareas` y `PTareasFichero`

4.3.1.5. Paquetes perfil y edición del perfil



(a) PPerfil



(b) PPerfilEdicion

Figura 4.18: Diagrama de clases de los paquetes `PPerfil` y `PPerfilEdicion`

4.3.1.6. Paquete información personal

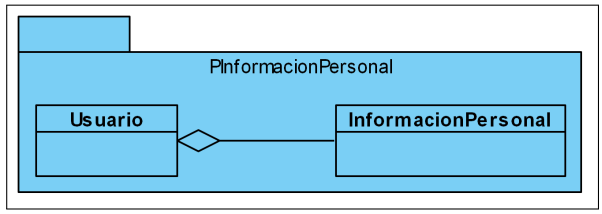


Figura 4.19: Diagrama de clases del paquete `PInformacionPersonal`

4.3.1.7. Paquete visualización de reuniones

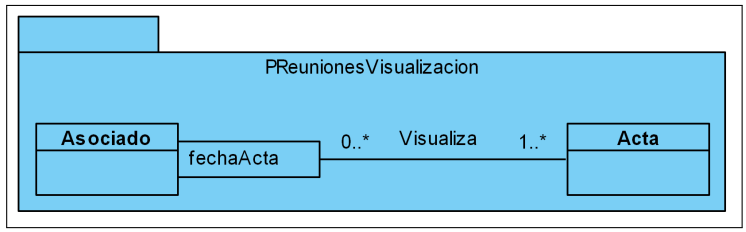


Figura 4.20: Diagrama de clases del paquete `PReunionesVisualizacion`

4.3.1.8. Paquete repositorio de ficheros

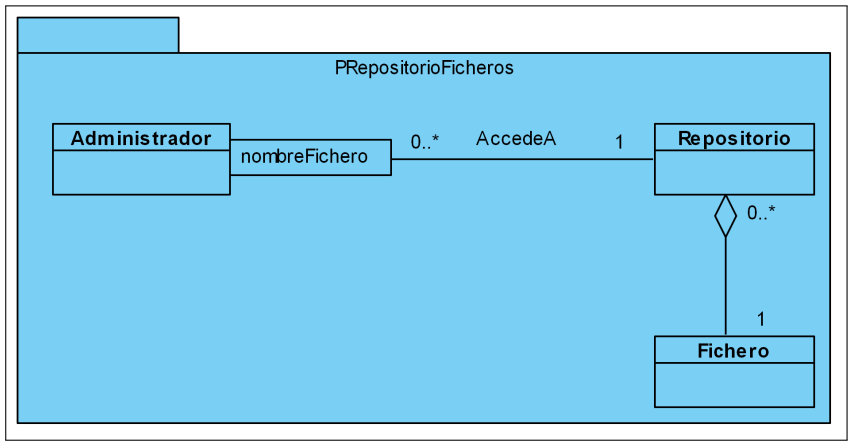


Figura 4.21: Diagrama de clases del paquete `PRepositorioFicheros`

4.3.1.9. Paquete registro de actividad

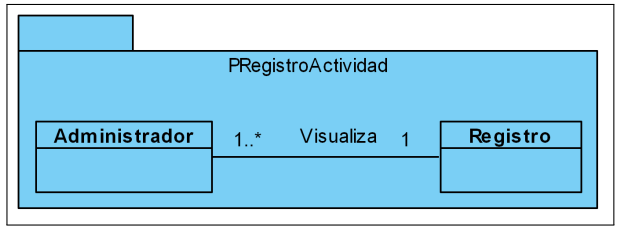


Figura 4.22: Diagrama de clases del paquete `PRegistroActividad`

4.3.1.10. Paquete copias de seguridad

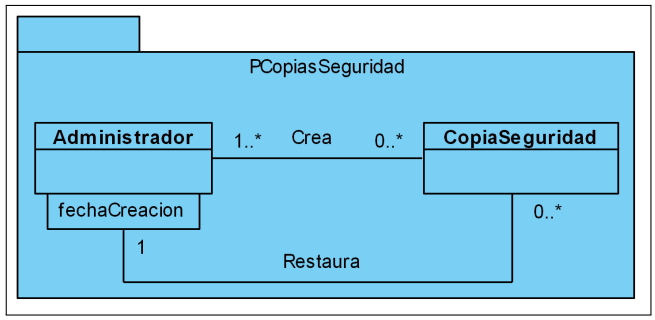


Figura 4.23: Diagrama de clases del paquete `PCopiasSeguridad`

4.3.2. Combinación de los paquetes de clases

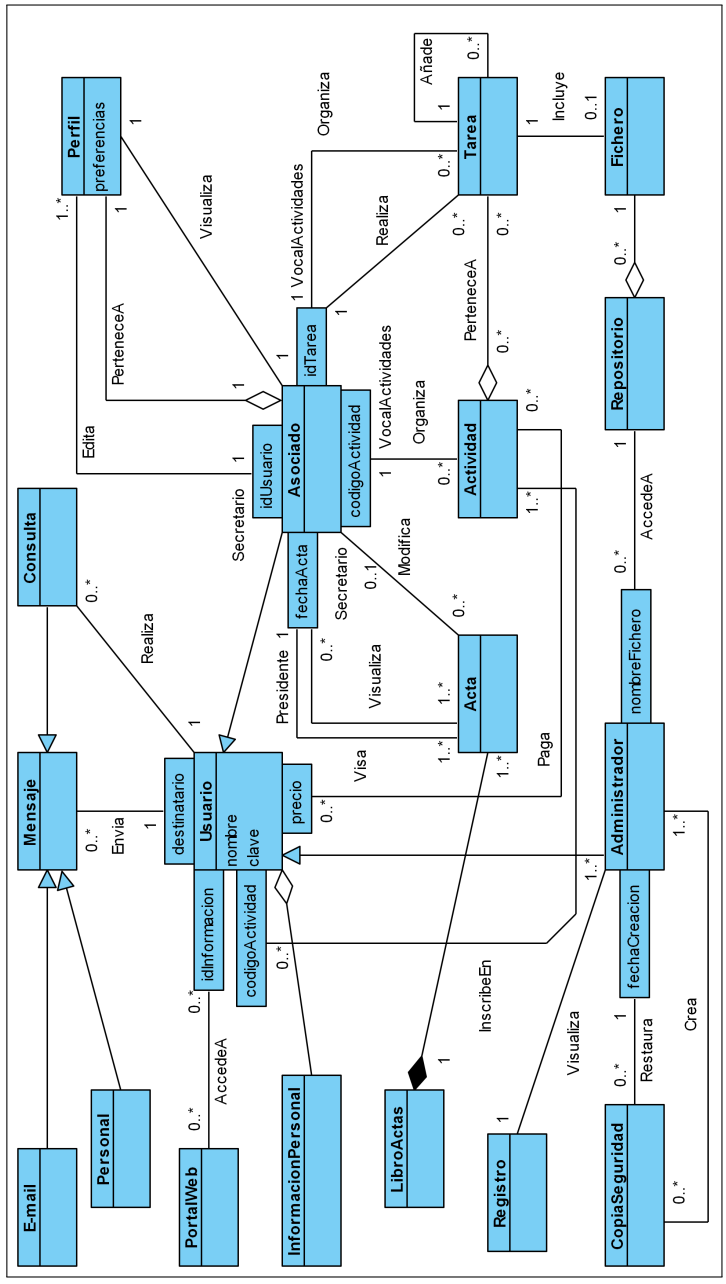


Figura 4.24: Diagrama de clases del sistema simplificado

Capítulo 5
Conclusiones y líneas futuras

"Lo importante es no cesar de hacerse preguntas"

Albert Einstein

Una vez terminado todo el trabajo se puede afirmar que el desarrollo de una línea de productos *software* no es una tarea fácil, ni tampoco es una tarea que deba ser realizada por un equipo pequeño, ya que sus mayores ventajas están en la reducción de los tiempos de producción de un nuevo producto, pero para ello la línea de productos ha de haberse construido y esta fase requiere tiempo y recursos humanos que no siempre es posible disponer, o al menos en una cantidad suficiente como para ofrecer un mínimo de calidad.

Los objetivos propuestos en la introducción se han cumplido:

- Los dos primeros objetivos propuestos, estudio del proceso de desarrollo basado en líneas de productos y estudio del modelo de características y su aplicación en el desarrollo de líneas de productos, se han cumplido. Se ha realizado un estudio del proceso y de los modelos de características.

  Por otro lado se ha elaborado una documentación teórico-práctica compacta sobre desarrollo de líneas de productos utilizando modelos de características y combinación de paquetes en UML2 gracias a la gran cantidad de bibliografía consultada de autores e investigadores diversos.

- El tercer objetivo, estudio de una línea de productos para asociaciones, también se considera cumplido ya que se ha modelado y documentado con gran precisión y detalle una línea de productos para asociaciones, partiendo casi en exclusiva de la información aportada por las asociaciones entrevistadas, pues en el momento de la elaboración del modelo no se encontró mucha documentación sobre aplicaciones que estuviesen en el dominio de aplicación ni tampoco cercanas al contexto.

  Es decir, se ha construido un modelo de características (con más de 280 características), y no sólo se ha completado su construcción sino que se ha documentado cada una de las características del modelo y especificado las posibles restricciones a tener en cuenta en la fase de configuración.

- Respecto al cuarto y último objetivo, construcción de la línea de productos con UML, una vez elaborado el modelo de características, la magnitud del mismo ha llevado a buscar una solución que redujese la variabilidad del modelo propuesto. Por ello, en base al modelo de características de la línea de productos, se procedió a realizar una simplificación del mismo eliminando

todas las características superfluas y agrupando funcionalidades. La simplificación se describió brevemente, haciendo referencia en todo momento al modelo de características original, para completar la descripción de algunas agrupaciones de funcionalidades o eliminación de características poco relevantes.

La simplificación también se ha hecho para poder aplicar la propuesta [31] de los investigadores Laguna y otros, ya que debido a la alta variabilidad de la línea de productos, el paso a UML2 podría haber llegado a ser prácticamente imposible. No hay que olvidar que el autor de este trabajo era una persona inexperta también en el proceso de combinación de paquetes propuesto por Laguna y otros, y al fin y al cabo, el objetivo era demostrar la validez de la propuesta.

Para finalizar estas conclusiones, se cree que se ha logrado presentar con éxito un caso de estudio de la aplicación de una metodología emergente en el desarrollo de *software* hoy en día, reforzando así la validez de la metodología y la propuesta de combinación de paquetes del Grupo GIRO para la transformación del modelo de características a UML2.

En este punto, hay que remarcar que todo este trabajo ha sido llevado a cabo por un desarrollador practicamente inexperto en el campo de las líneas de productos, los modelos de características y la combinación de paquetes. El trabajo no ha sido fácil, y algunas veces llegó a convertirse en un auténtico laberinto, especialmente al comienzo del proyecto debido a la falta de conocimientos y la necesidad de asimilar muchos conceptos nuevos en poco tiempo).

Por último, se desea destacar el interés investigador del trabajo ya que con frecuencia trabajos similares procuran proporcionar una aplicación funcional y con una interfaz agradable, obviando las bases teóricas y metodologías sobre las que se debe sustentar esa aplicación. Es importante poder aplicar con éxito las averiguaciones, contribuciones y propuestas de la comunidad científica, pero tanto o más importante es comprender y justificar su aplicación.

5.1. Líneas futuras

Expuestas las conclusiones, no se puede terminar el capítulo y también la memoria, sin una propuesta de líneas de trabajo futuras que se podrían seguir, ya que la finalización de un Proyecto Fin de Carrera no debería quedar nunca cerrada, y menos en el ámbito de la investigación en el que se enmarca este trabajo.

A continuación se enumeran las líneas a seguir que se consideran de interés:

- **Continuación de la construcción de la línea de productos para asociaciones en su versión simplificada**

  La continuación de la construcción de la línea de productos *software* modelada al completo es un trabajo para un equipo humano numeroso (aunque también se podrían escoger una serie de funcionalidades mínimas para obtener otra versión simplificada). Sin embargo, la propuesta simplificada sobre la que se ha realizado un primer análisis del dominio es un buen punto de ampliación en tanto que tiene una variabilidad lo suficientemente acotada como para ser diseñada e implementada.

  Incluso sería posible volver a hacer una nueva simplificación en caso necesario, aprovechando que el modelado en UML2 está hecho, por lo que simplemente sería diseñar e implementar.

  Se desea destacar que esta línea de trabajo ya está en marcha, y ya hay un alumno documentándose para poder llevarla a la práctica.

- **Mejora de la propuesta [31] de combinación de paquetes en UML2**

  La aplicación de la propuesta de Laguna y otros ha permitido un estudio detallado de la misma y en algunas ocasiones se ha observado que no satisfacía las necesidades de modelado, debido a la pérdida de información.

  El Grupo GIRO sigue investigando para subsanar estos pequeños defectos y ampliar la propuesta para que en un futuro UML integre en su meta-modelo la transformación de modelos de características mediante combinación de paquetes.

- **Construcción de herramientas de diseño de modelos de características**

  La herramienta utilizada, *Feature Model Plug-In*, para la realización de los modelos de características utiliza la notación estándar; sin embargo, la visualización de esa notación difiere de la notación original, y debido a las limitaciones tecnológicas, en ocasiones la lectura de un modelo resulta confusa.

  Una línea interesante puede ser la mejora del *plug-in* existente o la construcción de una herramienta nueva, pero que permita obtener un diagrama del modelo de características usando la notación estándar.

- **Construcción de herramientas de configuración**

  Muchas de las referencias utilizadas a lo largo de este trabajo dejan constancia de la falta de herramientas que permitan realizar una configuración con una interfaz sencilla.

  El *plug-in* de *Eclipse*, *fmp*, permite realizar la configuración de forma visual y se puede decir que es bastante aceptable; sin embargo, es un prototipo experimental en el que escasean funcionalidades básicas y otras son bastante rudimentarias e inestables.

Es importante desarrollar herramientas que faciliten esta tarea, ya que, como se ha comentado en este trabajo, la fase de configuración es la más importante, porque en ella se toman las decisiones sobre las funcionalidades que tendrá el producto que se obtendrá con la línea de productos *software*.

Bibliografía

[1] ANTKIEWICZ, MICHAL y CZARNECKI, KRZYSZTOF: «FeaturePlugin: feature modeling plug-in for Eclipse». Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange, 2004, pp. 67–72.
http://www.swen.uwaterloo.ca/~kczarnec/etx04.pdf

[2] BLAHA, MICHAEL y RUMBAUGH, JAMES: Object-Oriented Modeling and Design with UML (2nd edition). Prentice Hall (Pearson Education), 2005. ISBN 0131968599.

[3] BOOCH, GRADY; RUMBAUGH, JAMES y JACOBSON, IVAR: El Lenguaje Unificado de Modelado (2a edición). Addison Wesley (Pearson Educación), 2006. ISBN 8478290761.

[4] CENTRO NACIONAL DE REFERENCIA DE APLICACIÓN DE LAS TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN: «Sistema de Gestión de Proyectos para ONG's: Análisis Funcional». Technical report, Fundación CENATIC, Vistahermosa, 1, 3a Planta, 06200 Almendralejo (Badajoz), Spain, 2008.
http://www.cenatic.es/proyectos/blogs/media/blogs/a/Analisis_Funcional_GONG_0106.pdf

[5] CLAUSS, MATTHIAS: «Modeling variability with UML». Proc. of Net. ObjectDays 2001, Young Researchers Workshop on Generative and Component-Based Software Engineering, 2001, pp. 226–230.

[6] COHEN, SHOLOM G.; JAY L. STANLEY, JR.; PETERSON, A. SPENCER y ROBERT W. KRUT, JR.: «Application of Feature-Oriented Domain Analysis to the army movement control domain». Technical report (CMU/SEI-91-TR-028), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 1992.

[7] CZARNECKI, KRZYSZTOF: «Overview of Generative Software Development». En: Unconventional Programming Paradigms (UPP), pp. 326–341, 2004.
http://swen.uwaterloo.ca/~kczarnec/gsdoverview.pdf

[8] CZARNECKI, KRZYSZTOF y ANTKIEWICZ, MICHAL: «Mapping Features to Models: A Template Approach Based on Superimposed Variants». En: Generative Programming and Component Engineering (GPCE'05), pp. 422–437, 2005.

[9] CZARNECKI, KRZYSZTOF; ANTKIEWICZ, MICHAL; KIM, CHANG HWAN PETER; LAU, SEAN y PIETROSZEK, KRZYSZTOF: «Model-Driven Software Product Lines». En: OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented

programming, systems, languages, and applications, pp. 126–127. ACM, New York, NY, USA. ISBN 1595931937, 2005. doi: http://doi.acm.org/10.1145/1094855.1094896.

[10] CZARNECKI, KRZYSZTOF y EISENECKER, ULRICH: Generative programming: methods, tools, and applications. ACM Press/Addison-Wesley Publishing, 2000.

[11] CZARNECKI, KRZYSZTOF; HELSEN, SIMON y EISENECKER, ULRICH: «Staged configuration through specialization and multilevel configuration of Feature Models». Software Process: Improvement and Practice, 2005, **10(2)**, pp. 143–169.

[12] CZARNECKI, KRZYSZTOF y KIM, CHANG HWAN PETER: «Cardinality-Based Feature Modeling and Constraints: A Progress Report». En: Proceedings of the International Workshop on Software Factories At OOPSLA 2005, , 2005.

[13] CZARNECKI, KRZYSZTOF; KIM, CHANG HWAN PETER y KALLEBERG, KARL TRYGVE: «Feature models are views on ontologies». Software Product Line Conference, 2006 10th International, 2006, pp. 41–51.

[14] DOLOG, PETER y NEJDL, WOLFGANG: «Using UML-Based Feature Models and UML Collaboration Diagrams to Information Modelling for Web-Based Applications». En: Thomas Baar; Alfred Strohmeier; Ana Moreira y Stephen J. Mellor (Eds.), UML 2004 - The Unified Modeling Language. Model Languages and Applications. 7th International Conference, Lisbon, Portugal, October 11-15, 2004, Proceedings, volumen 3273 de LNCS, pp. 425–439. Springer, 2004.

[15] GOMAA, HASSAN: «A software design method for real-time systems». Communications of the ACM, 1984, **27(9)**, pp. 938–949.

[16] ——: Designing Software Product Lines with UML. Addison Wesley (Pearson Education), 2004.

[17] GRISS, MARTIN L.; FAVARO, JOHN y D'ALESSANDRO, MASSIMO: «Integrating Feature Modeling with the RSEB». En: ICSR '98: Proceedings of the 5th International Conference on Software Reuse, pp. 76–85. IEEE Computer Society, Washington, DC, USA. ISBN 08186-83775, 1998.

[18] GROHER, IRIS; PAPAJEWSKI, HOLGER y VOELTER, MARKUS: «Integrating Model-Driven Development and Software Product Line Engineering», 2007. Presentation at the Eclipse 2007 Summit Symposium.

[19] HALMANS, G. y POHL, K.: «Communicating the variability of a software-product family to customers». Software and Systems Modeling, 2003, **2(1)**, pp. 15–36.

[20] JACOBSON, I.; GRISS, M. y JONSSON, P.: Software reuse: architecture, process and organization for business success. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1997.

[21] JANOTA, MIKOLÁŠ y KINIRY, JOSEPH: «Reasoning about Feature Models in Higher-Order Logic». 11th International Software Product Line Conference, 2007 (SPLC 2007), 2007, pp. 13–22. doi: 10.1109/SPLINE.2007.36.

[22] JOHN, I. y MUTHIG, D.: «Tailoring Use Cases for Product Line Modeling». Proceedings of the International Workshop on Requirements Engineering for Product Lines, 2002, **2002**, pp. 26–32.

[23] KANG, KYO C.: «Feature-Oriented Development of Applications for a Domain». En: ICSR '98: Proceedings of the 5th International Conference on Software Reuse, p. 354. IEEE Computer Society, Washington, DC, USA. ISBN 0-8186-8377-5, 1998.

[24] KANG, KYO C.; COHEN, SHOLOM G.; HESS, JAMES A.; NOVAK, WILLIAM E. y PETERSON, A. SPENCER: «Feature-Oriented Domain Analysis (FODA) Feasibility Study». Technical report (CMU/SEI-90-TR-21), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 1990.

[25] KANG, KYO C.; LEE, JAEJOON y DONOHOE, PATRICK: «Feature-Oriented Project Line Engineering». IEEE Software, 2002, **19(4)**, pp. 58–65. ISSN 0740-7459. doi: http://dx.doi.org/10.1109/MS.2002.1020288.

[26] KIM, CHANG HWAN PETER: On the Relationship between Feature Models and Ontologies. MASc Thesis, Electrical and Computer Engineering Department, University of Waterloo, Canada, 200 University Avenue West, Waterloo, Ontario, Canada, 2006.
http://swen.uwaterloo.ca/~chpkim/chpkim-masc-thesis.pdf

[27] KIM, CHANG HWAN PETER y CZARNECKI, KRZYSZTOF: «Synchronizing Cardinality-Based Feature Models and Their Specializations». Model Driven Architecture-Foundations and Applications, First European Conference, ECMDA-FA, 2005, pp. 7–10.

[28] KRUEGER, CHARLES W.: «Introduction to the emerging practice Software Product Line Development». Method & Tools, 2006, **14(3)**, pp. 3–15.

[29] KRUEGER, CHARLES W.; HETRICK, WILLIAM A. y MOORE, JOSEPH G.: «Making an incremental transition to Software Product Line practice». Method & Tools, 2006, **14(3)**, pp. 16–27.

[30] LAGUNA, MIGUEL ÁNGEL y GONZÁLEZ-BAIXAULI, BRUNO: «Variabilidad, Trazabilidad y Líneas de Productos: una Propuesta basada en UML y Clases Parciales». En: XII Jornadas Ingenieria del Software y Bases de Datos (JISBD 2007), Zaragoza, Spain, , 2007.
http://giro.infor.uva.es/Publications/2007/LG07

[31] LAGUNA, MIGUEL ÁNGEL; GONZÁLEZ-BAIXAULI, BRUNO y MARQUÉS CORRAL, JOSÉ MANUEL: «Feature Patterns and Multi-Paradigm Variability Models». Technical Report 2008/01, Grupo GIRO, Department of Computer Science, University of Valladolid, Campus Miguel Delibes, 47011 Valladolid, Spain, 2008.
http://giro.infor.uva.es/Publications/2008/LGM08

[32] LARMAN, CRAIG: UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado (2a edición). Prentice Hall (Pearson Educación), 2003. ISBN 8420534382.

[33] LAU, SEAN Q.: Domain Analysis of E-Commerce Systems Using Feature-Based Model Templates. MASc Thesis, Electrical and Computer Engineering Department, University of Waterloo, Canada, 200 University Avenue West, Waterloo, Ontario, Canada, 2006.
http://gp.uwaterloo.ca/files/2006-lau-masc-thesis.pdf

[34] MADSEN, OLE LEHRMANN; MØLLER-PEDERSEN, BIRGER y NYGAARD, KRYSTEN: «Object-oriented programming in the BETA programming language», 1993.

[35] OBJECT MANAGEMENT GROUP: «OMG Unified Modeling Language (OMG UML), Infrastructure, v2.1.2». Technical Report (Formal/2007-11-03), OMG, 140 Kendrick Street, Building A Suite 300, Needham, MA 02494, USA, 2007.
http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF

[36] PÉREZ-PELLÓN VALENTÍN-GAMAZO, MARTA: Aplicación web para la Asociación Casa de Beneficencia de Valladolid. BSc Thesis, Departamento de Informática (Arquitectura, Ciencias de la Computación y Lenguajes), Universidad de Valladolid, Spain, 2007.

[37] ŠÍPKA, MILOSLAV: «Exploring the Commonality in Feature Modeling Notations». IIT. SRC 2005: Student Research Conference, 2005, pp. 139–144.
http://www2.fiit.stuba.sk/iit-src/2005/22-sipka.pdf

[38] SOMMERVILLE, IAN: Ingeniería del software (6a edición). Addison Wesley (Pearson Educación), 2006. ISBN 9702602068.

[39] VON DER MASSEN, T. y LICHTER, H.: «Modeling Variability by UML Use Case Diagrams». Proceedings of the International Workshop on Requirements Engineering for Product Lines (REPL'02), 2002, **AVAYA Labs**, pp. Technical Report: ALR–2002–033.

[40] VRANIĆ, VALENTINO: Multi-Paradigm Design with Feature Modeling. Ph.D Thesis, Faculty of Informatics and Information Technologies (Slovak University of Technology in Bratislava), 2004.
http://www2.fiit.stuba.sk/~vranic/pub/MPDfm.pdf

[41] ——: «Reconciling Feature Modeling: A Feature Modeling Metamodel». En: Matias Weske y Peter Liggsmeyer (Eds.), Proc. of 5th Annual International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World (Net.ObjectDays 2004), LNCS 3263, pp. 122–137. Springer, Erfurt, Germany, 2004.

[42] VRANIĆ, VALENTINO y MARKO, VLADIMÍR: «Dealing with Unstable Domains in Product-Line Architecture Development». En: Proc. of 9th International Conference on Information Systems Implementation and Modelling (ISIM 2006), pp. 57–64. Pøerov, Czech Republic, 2006.

[43] ——: «Developing a Product-Line Based Architecture in a Domain Under Research». En: Pavol Návrat y otros (Eds.), Tools for Acquisition, Organisation and Presenting of Information and Knowledge, Research roject Workshop (NAZOU), in conjunction with ITAT 2006, pp. 211–222. Bystrá dolina, Nízke Tatry, Slovakia, 2006.

[44] VRANIĆ, VALENTINO y ŠÍPKA, MILOSLAV: «Binding Time Based Concept Instantiation in Feature Modeling». En: Maurizio Morisio (Ed.), Proc. of 9th International Conference on Software Reuse (ICSR 2006), LNCS 4039, pp. 407–410. Springer, Turin, Italy, 2006.

[45] VRANIĆ, VALENTINO y ŚNIRC, JÁN: «Integrating Feature Modeling into UML». En: Robert Hirschfeld y otros (Eds.), Proc. of NODe 2006, LNI P-88, pp. 3–15. GI, Erfurt, Germany, 2006.

[46] ZITO, ALANNA y DINGEL, JUERGEN: «Modeling UML2 Package Merge with Alloy». En: Proceedings of the 1st Alloy Workshop (Alloy'06), Portland, Oregon, USA, 2006.

[47] ZITO, ALANNA; DISKIN, ZINOVY y DINGEL, JÜRGEN: «Package Merge in UML 2: Practice vs. Theory». En: Oscar Nierstrasz; Jon Whittle; David Harel y Gianna Reggio (Eds.), MoDELS, volumen 4199 de Lecture Notes in Computer Science, pp. 185–199. Springer. ISBN 3540457720, 2006. doi: http://dx.doi.org/10.1007/11880240_14.
http://dblp.uni-trier.de/db/conf/models/models2006.html#ZitoDD06

Apéndice A
Meta-modelo de *Feature Modeling*

"Hacemos las reglas para los demás y las excepciones para nosotros mismos"

Charles Lemesle

A continuación, se presenta un meta-modelo para el modelado basado en modelos de características, que es una propuesta [41] del investigador Vranić.

En esta propuesta, se hace uso de los elementos existentes en el modelado de características; y con ella, se intenta resolver el problema de las diferentes notaciones en uso actualmente (FODA, ODM, Czarnecki-Eisenecker o MPD_{FM}).

La propuesta está basada en las nociones de dominio, concepto y característica. Y aunque no es ningún estándar[1] en cuanto al modelado de características, muestra con gran precisión todos los elementos que se pueden incorporar a un modelo de características, así como la posibilidad de modelar la variabilidad y lo común de un dominio.

La aparación del símbolo ® en algunos diagramas, indica una referencia al diagrama correspondiente al concepto acompañado por dicho símbolo, es decir, en lugar de integrar en el mismo diagrama dónde aparece el concepto con el símbolo de referencia la representación de su diagrama, se extrae el diagrama y se denota de la forma indicada.

Las siguientes restricciones se han de tener en cuenta en las figuras que se indican:

- Figura A.2: $\neg$`Root.Node.Reference` (El nodo de una raíz no puede ser una referencia)
- Figura A.9: $\neg$`(Node` $\vee$ `Feature)` (Un enlace no puede unir a un nodo y a una característica a la vez)

[1] No hay que olvidar que todos los modelos presentados en éste apéndice hacen uso de la notación extendida de Czarnecki-Eisenecker

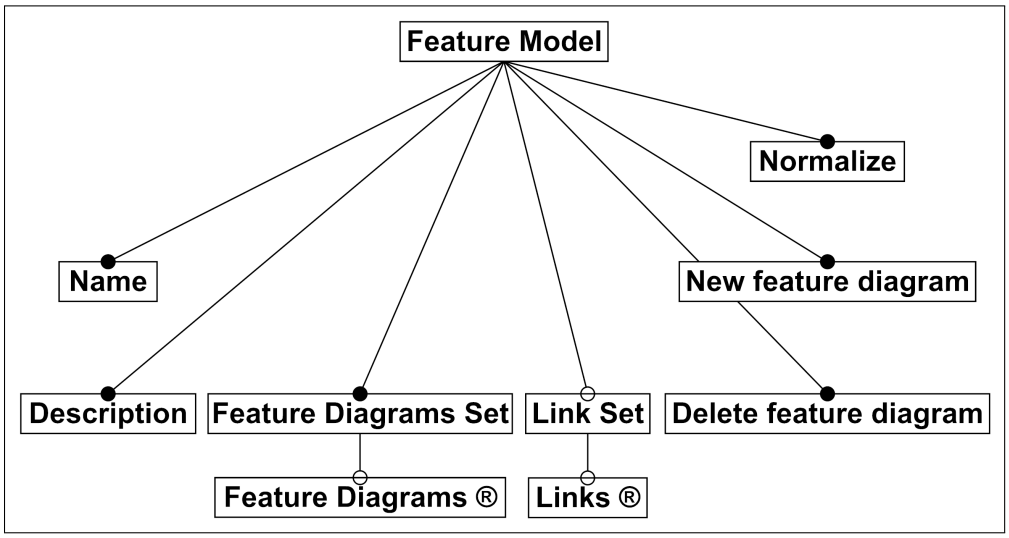


Figura A.1: Modelo de características

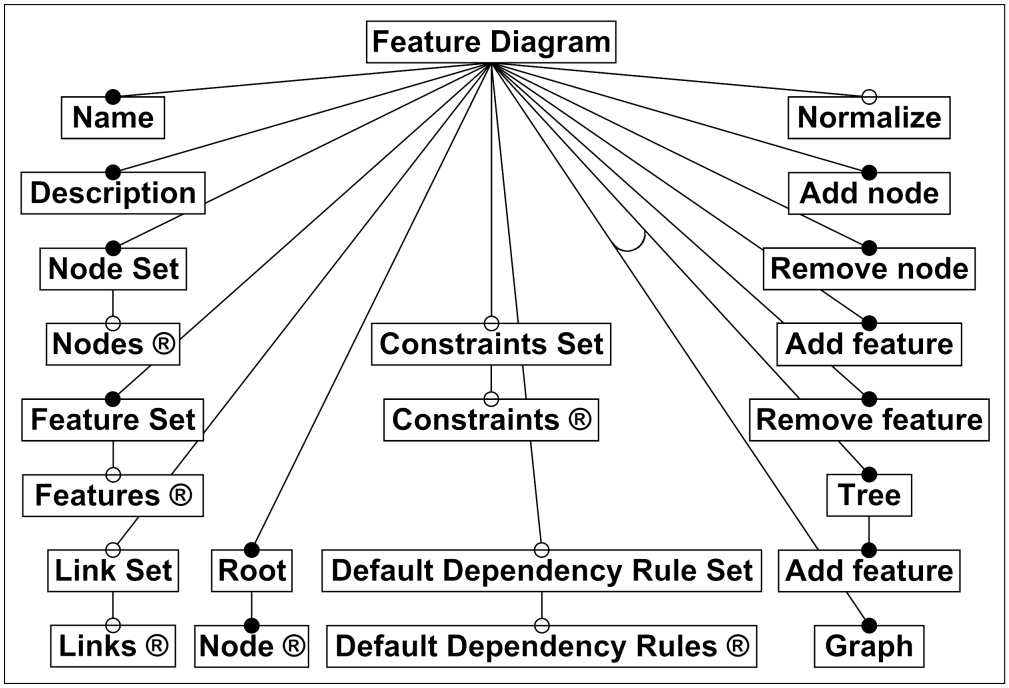


Figura A.2: Diagrama de características

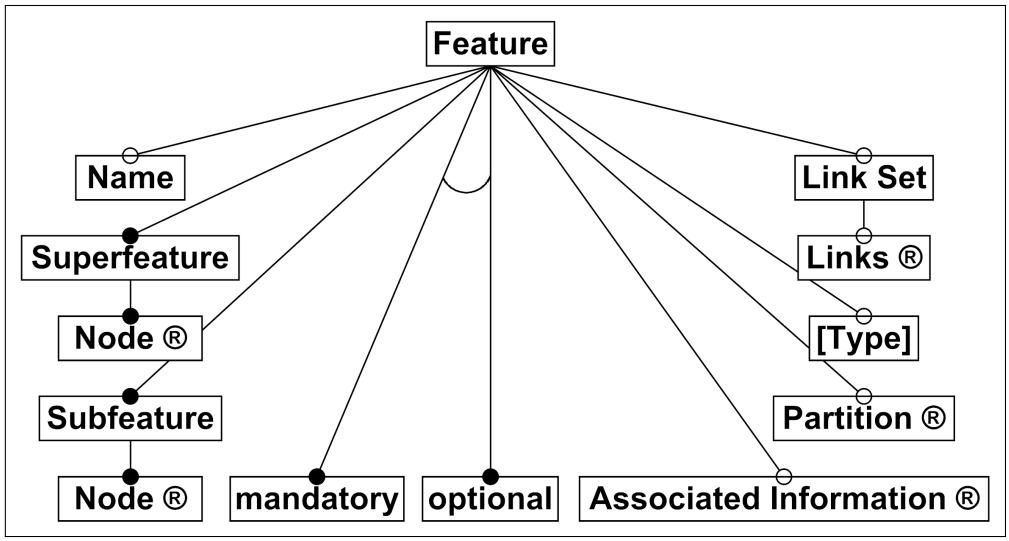


Figura A.3: Característica

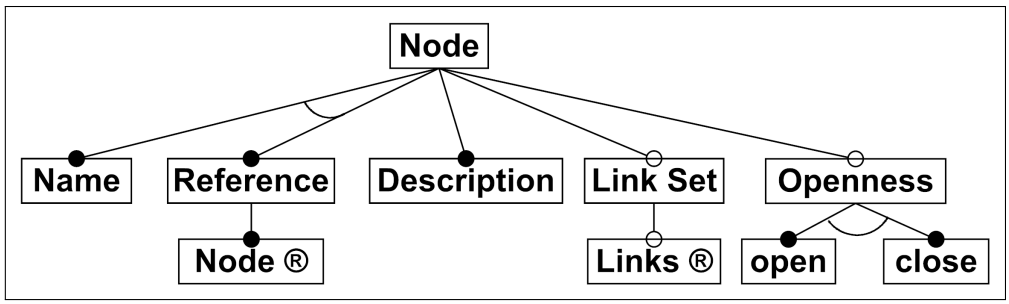


Figura A.4: Nodo

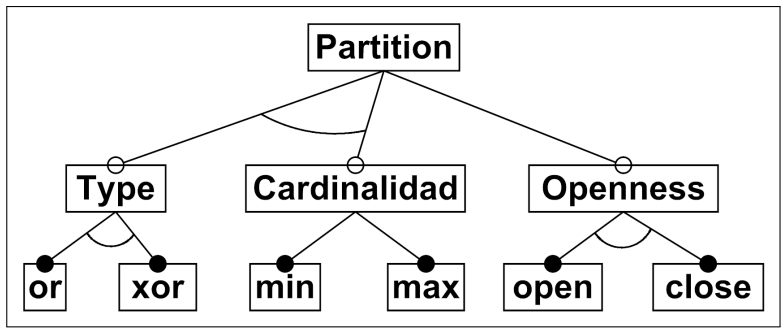


Figura A.5: Partición

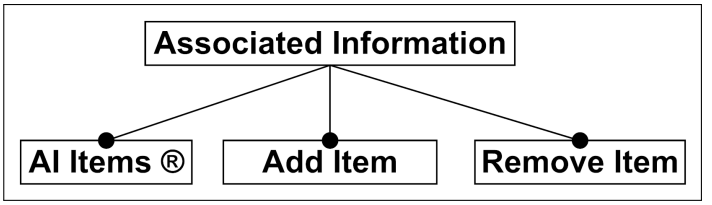


Figura A.6: Información asociada a una característica

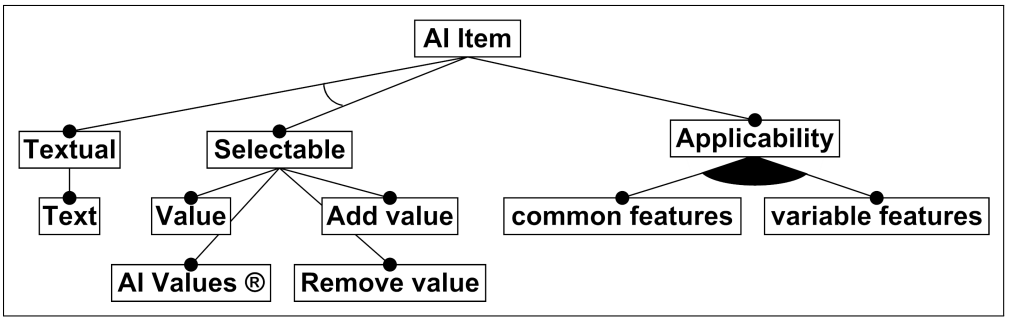


Figura A.7: Elemento básico de una información asociada a una característica

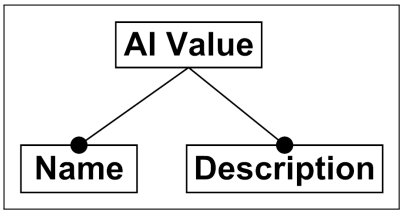


Figura A.8: Valor de un elemento básico de una información asociada a una característica

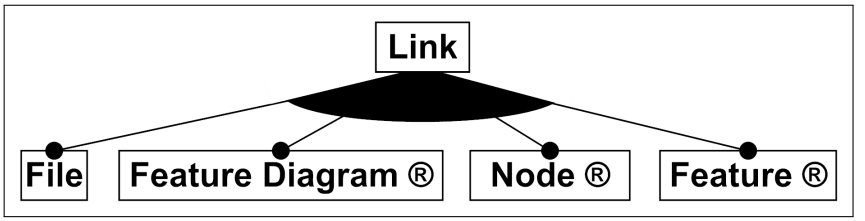


Figura A.9: Enlace

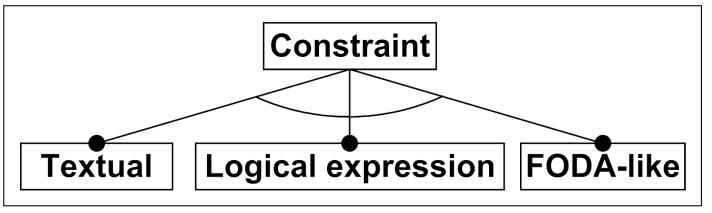


Figura A.10: Restricción

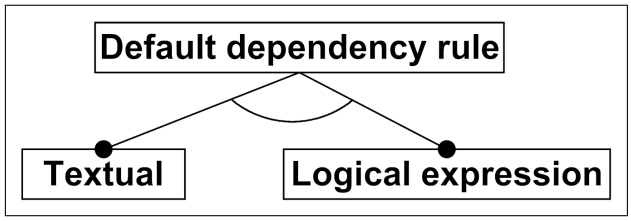


Figura A.11: Regla de dependencia por defecto

Apéndice B

Diagrama de clases del meta-modelo utilizado en *Feature Model Plug-In*

"Con constancia y tenacidad se obtiene lo que se desea; la palabra imposible no tiene significado"

Napoleón Bonaparte

A continuación, se presenta el diagrama de clases del meta-modelo de los modelos de características que se pueden generar con la herramienta *Feature Model Plug-In* realizada [1] por el equipo del investigador de la Universidad de Waterloo, Krzysztof Czarnecki.

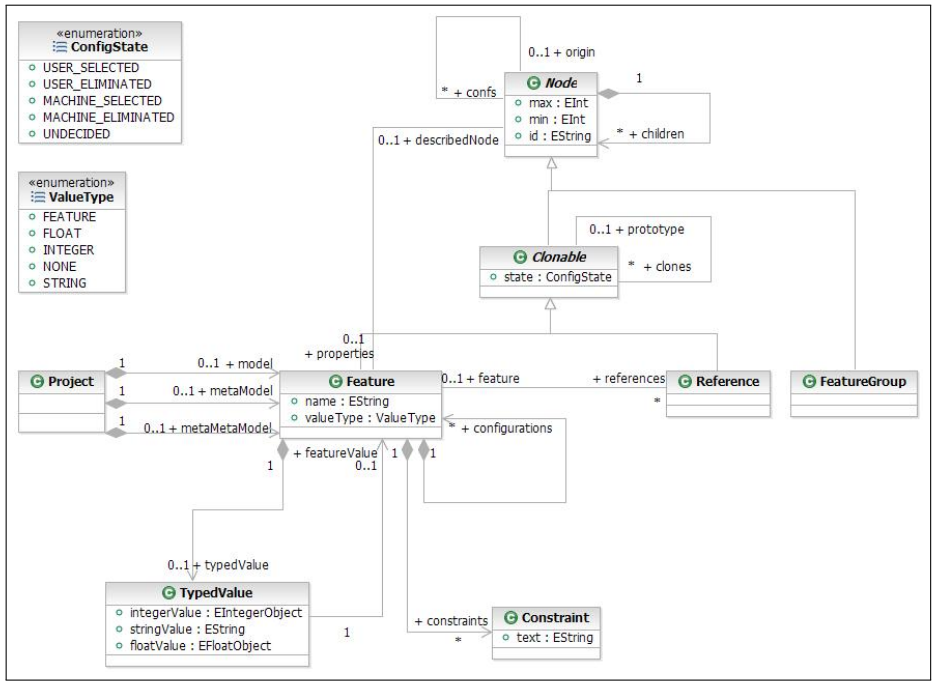


Figura B.1: Diagrama de clases del meta-modelo de *fmp*

Apéndice C

Instalación y uso en *Eclipse* de *Feature Model Plug-In*

"Todo hombre es útil a la humanidad por el simple hecho de existir"

Jean-Jacques Rousseau

C.1. Instalación de *fmp*

En este primer apartado se describe graficamente el proceso de instalación del *plug-in* de *Eclipse* utilizado para la realización de los modelos de características: *fmp*.

La instalación del entorno de desarrollo *Eclipse* y de un *plug-in* cualquiera es muy sencilla, por eso se describirá textualmente la instalación de *Eclipse* y del *plug-in* EMF[1] (*Eclipse Modeling Framework*).

Una vez disponga del paquete de instalación de *Eclipse* para la plataforma[2] en la que se vaya a instalar, simplemente descomprima el paquete en un directorio sobre el que tenga permisos de escritura, la descompresión creará un subdirectorio de nombre `eclipse` donde depositará todos los archivos necesarios (la descompresión creará cuatro subdirectorios a su vez: `configuration`, `features`, `plugins` y `readmes`). Para ejecutar *Eclipse* basta con que ejecute el programa de nombre `Eclipse` ó `eclipse` que encontrará en el subdirectorio `eclipse`.

La instalación de EMF es igual de sencilla que la del entorno de desarrollo. Una vez disponga[3] del paquete[4]del *plug-in* EMF, basta con que descomprima el contenido del paquete en el mismo directorio en el que haya instalado *Eclipse*. La organización interna del *plug-in* está diseñada para que los ficheros necesarios se coloquen en los subdirectorios adecuados (para EMF serán `features` y `plugins`).

[1] EMF es un requisito del *plug-in fmp* por lo que no se debe omitir su instalación

[2] En el disco encontrará el entorno de desarrollo *Eclipse* en versión SDK para diferentes plataformas. También se ha proporciona el código fuente de *Eclipse*

[3] En el disco encontrará los ficheros del *plug-in* EMF, se recomienda que utilice `emf-sdo-xsd-SDK-2.2.4.zip`

[4] Una de las ventajas de que *Eclipse* utilice la plataforma *Java* es la abstracción de la capa de *hardware*, por lo que el desarrollo de *plug-ins* para *Eclipse* no suele requerir de versiones específicas para cada plataforma (siempre que se usen las librerías de desarrollo proporcionadas por *Eclipse*)

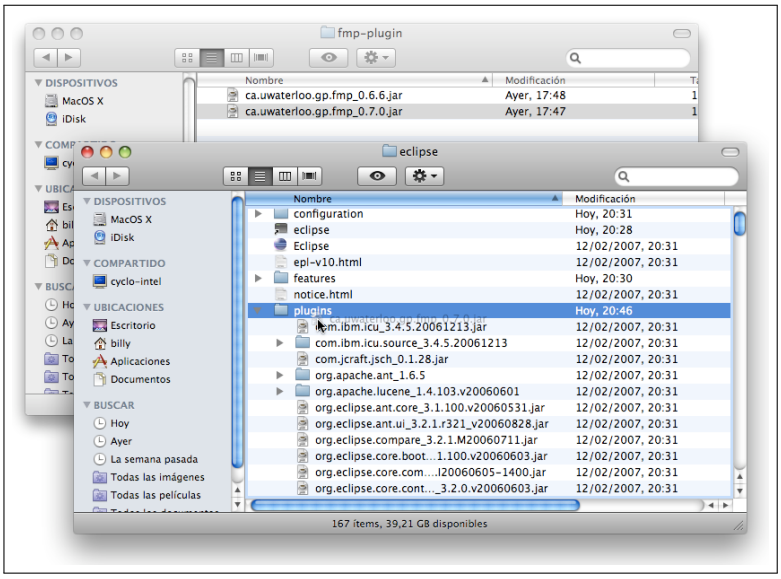


Figura C.1: Instalación de *Feature Model Plug-In*

Una vez haya abierto el subdirectorio donde haya instalado *Eclipse* y el subdirectorio `fmp-plugin` del disco, arrastre el fichero `ca.uwaterloo.gp.fmp_0.7.0.jar` al subdirectorio `plugins` (o bien copie y pegue) como se ilustra en la figura C.1. Debe quedar algo similar a la figura C.2.

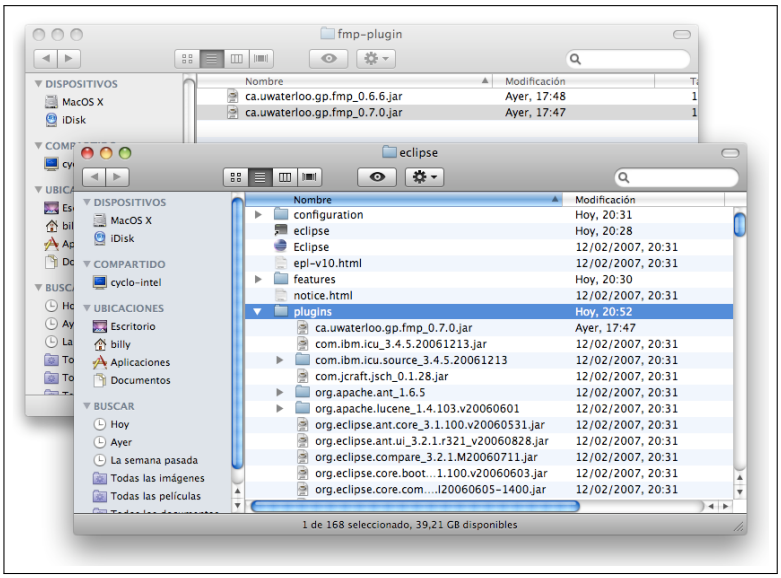


Figura C.2: Situación después de la instalación de *Feature Model Plug-In*

C.2. Uso de *fmp*

Los siguientes apartados mostrarán pequeños ejemplos de uso de la herramienta *Feature Model Plug-In*. Antes de comenzar con cada apartado se mostrará el proceso de creación de un proyecto en el que se almacenarán los ficheros de los modelos de características.

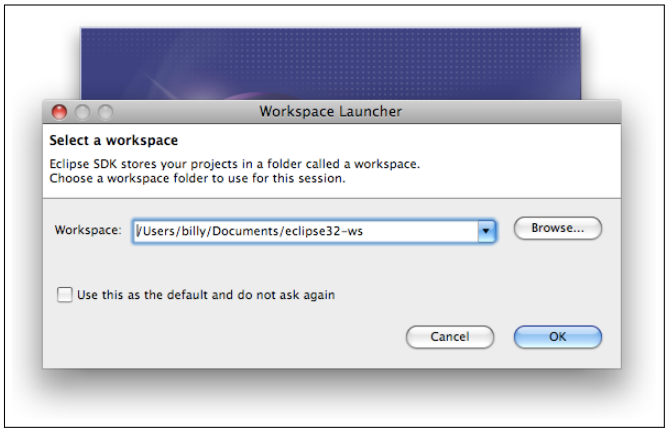


Figura C.3: Selección del directorio de trabajo de *Eclipse*

Una vez haya arrancado *Eclipse* e indicado el directorio de trabajo durante la sesión tal y como se muestra en la ilustración de la figura C.3, obtenga el menú desplegable para la selección de la creación de un proyecto pulsando en el icono de la esquina superior izquierda tal y como muestra la ilustración de la figura C.4.

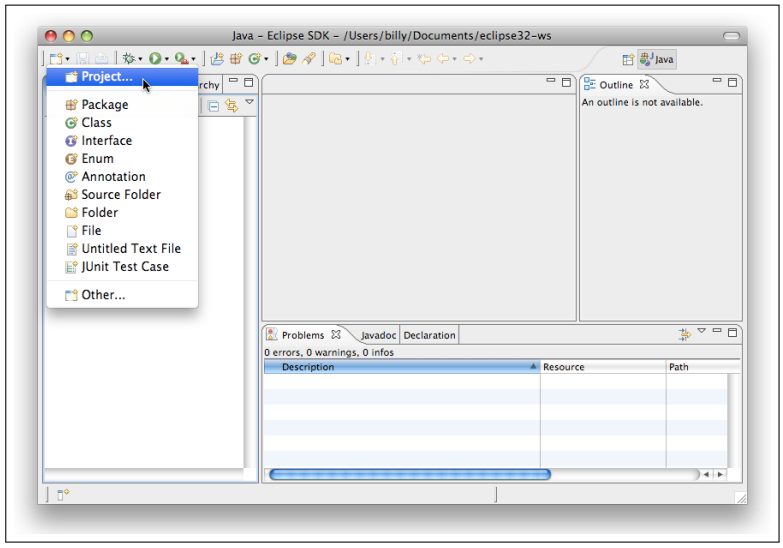


Figura C.4: Selección en el menú para la creación de un proyecto

El tipo de proyecto a escoger es irrelevante, pero se aconseja elegir un proyecto de tipo `General`, en la ilustración de la figura C.5 se muestra la elección.

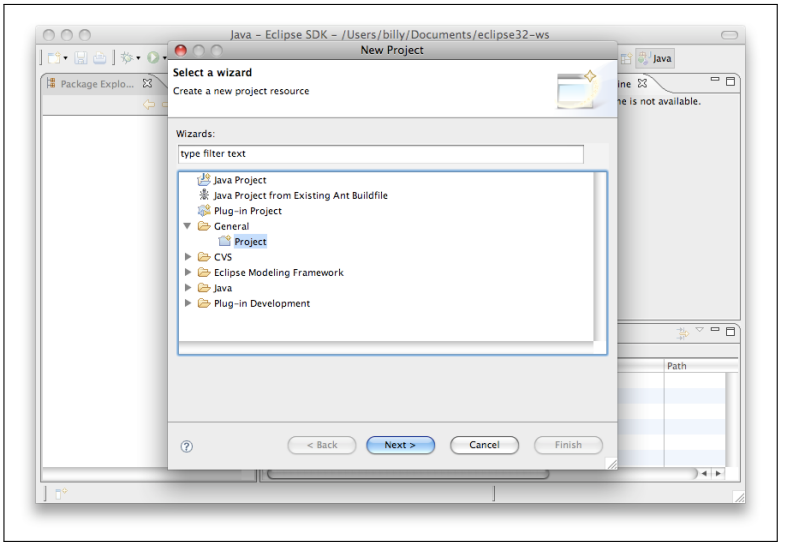


Figura C.5: Selección del tipo de proyecto a crear

Independientemente del tipo de proyecto escogido se ha de establecer un nombre identificativo para el proyecto (que lo único que hará *Eclipse* será crear un subdirectorio en el directorio de trabajo con el nombre dado, aunque si lo desea, se puede cambiar la ruta), tal y como se muestra en la ilustración de la figura C.6.

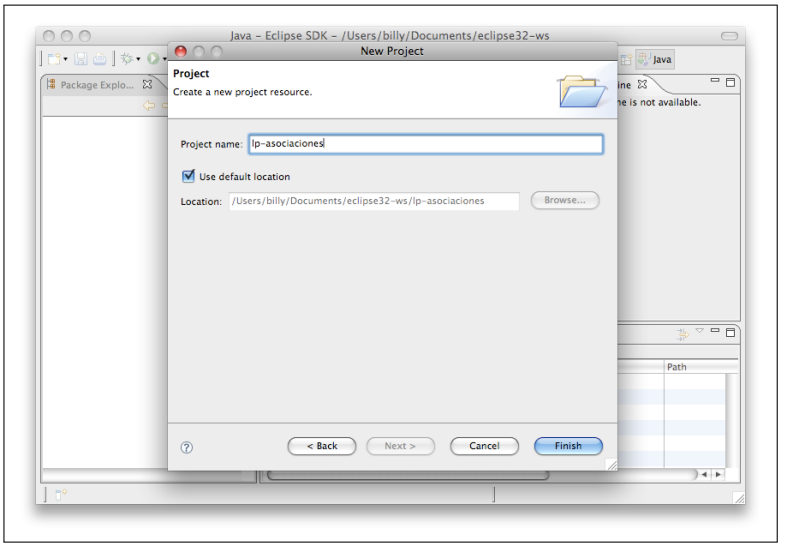


Figura C.6: Escritura del nombre del proyecto

Una vez haya finalizado la configuración del proyecto, debería tener una ventana de *Eclipse* similar a la mostrada en la ilustración de la figura C.7.

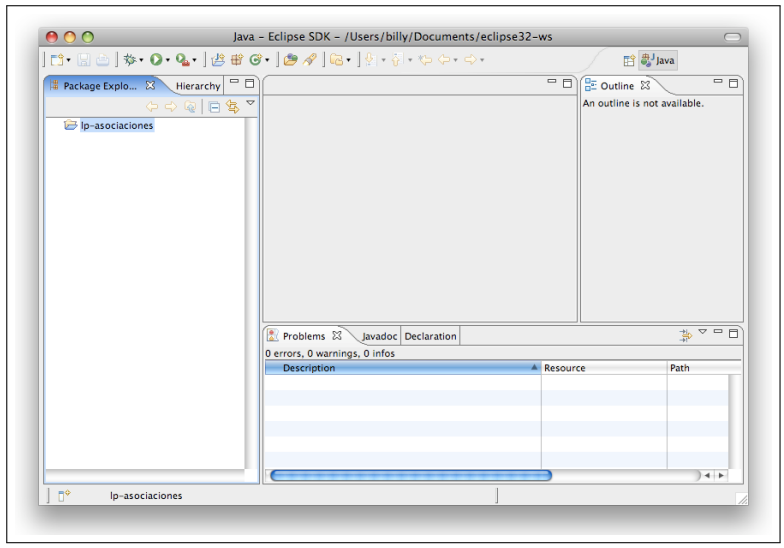


Figura C.7: Estado de *Eclipse* una vez se ha creado el proyecto

C.2.1. Gestión de un modelo de características

Para diseñar un modelo de características hay que añadir un fichero de tipo `.fmp` al proyecto, para ello, habrá que obtener el menú desplegable y seleccionar `Nuevo->Otros` (véase la figura C.8).

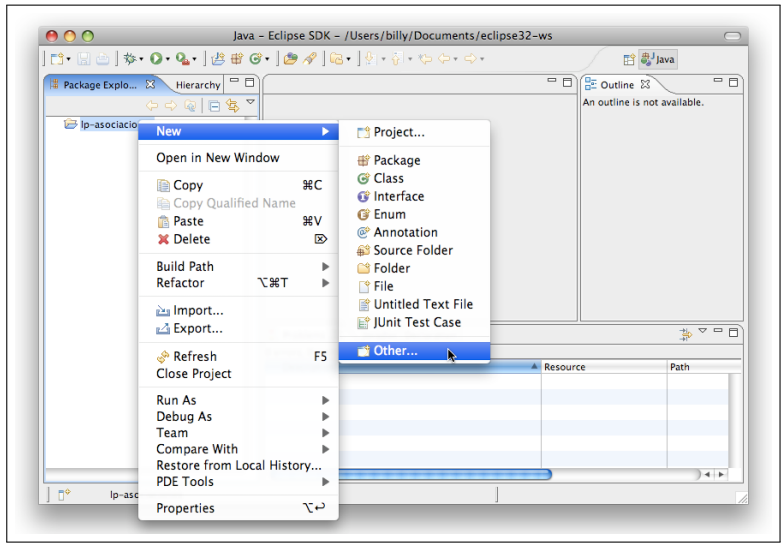


Figura C.8: Selección en el menú para la creación de un modelo de características

Para crear un modelo de características, habrá que seleccionar el tipo de fichero `Feature Model` tal y como muestra la ilustración de la figura C.9.

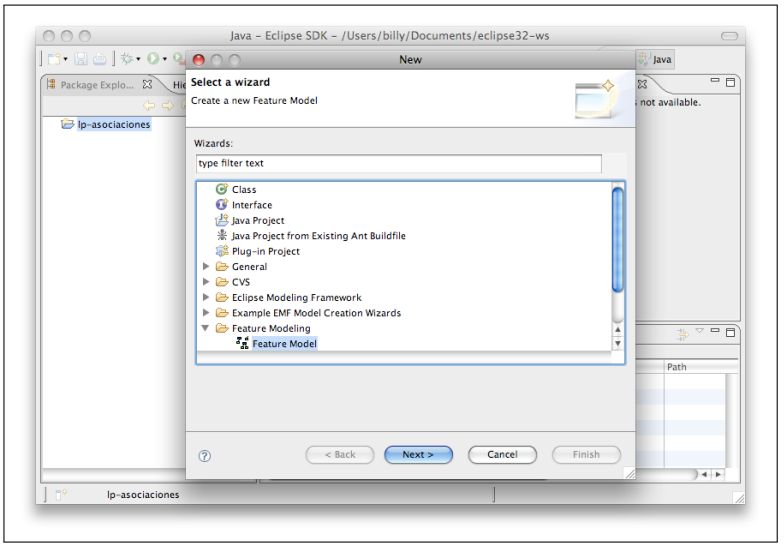


Figura C.9: Selección del tipo de fichero a añadir al proyecto

Una vez escogido el tipo de fichero a añadir, hay que dar un nombre identificativo al fichero (tenga en cuenta que el nombre debe incluir la extensión `.fmp`). La ilustración de la figura C.10 muestra el proceso.

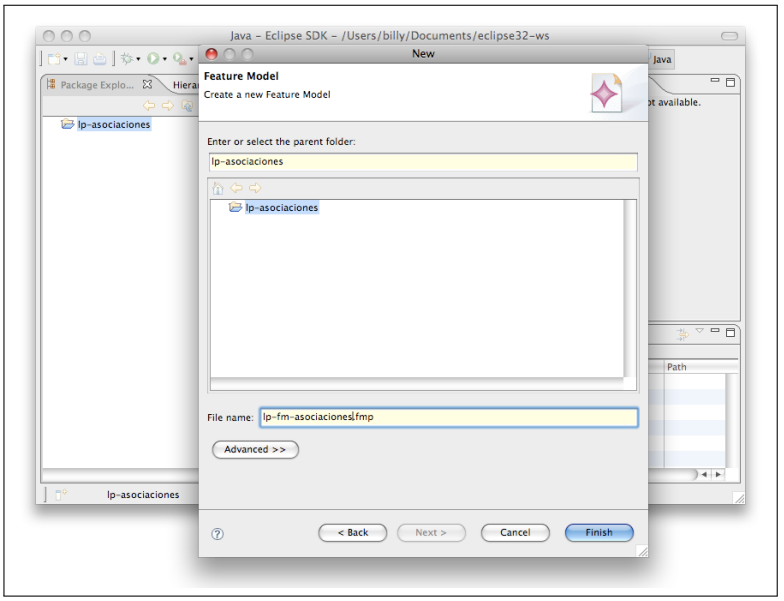


Figura C.10: Escritura del nombre del fichero

Una vez haya terminado con el asistente, debería tener una ventana de *Eclipse* similar a la mostrada en la ilustración de la figura C.11 (en este caso se ha eliminado la subventana `Outline` y se han minimizado las subventanas inferiores).

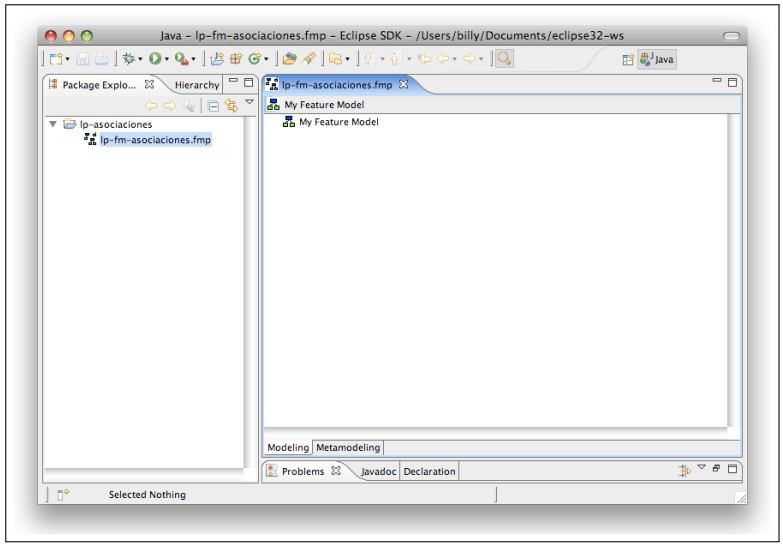


Figura C.11: Estado de *Eclipse* una vez se ha añadido el fichero

Para añadir una característica (siempre asociada a otra característica, excepto la raíz) basta obtener el menú desplegable sobre la característica padre como muestra la ilustración de la figura C.12.

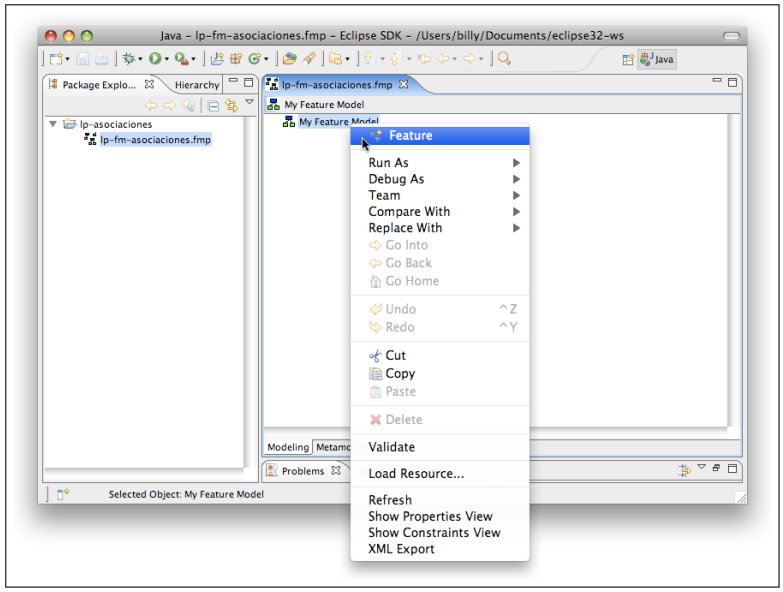


Figura C.12: Selección en el menú para añadir una característica

Toda característica debe tener un nombre identificativo y único en el modelo (en realidad deberá tener el nombre de una variable, único, y un nombre para el lector del modelo), para ello pulse dos veces sobre la característica a renombrar, y así cambiará el nombre como en la ilustración de la figura C.13.

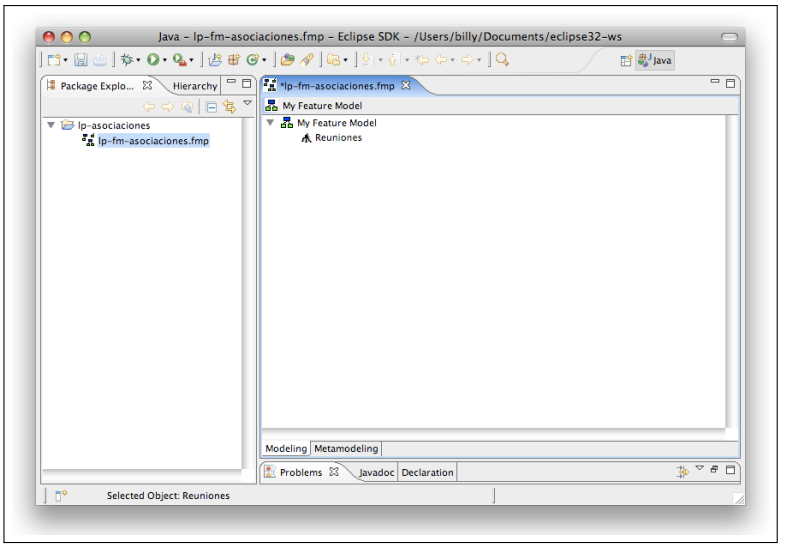


Figura C.13: Cambio de nombre de una característica

Por defecto, todas las características se añaden al modelo como opcionales, para cambiar el carácter, obtenga el menú desplegable sobre la característica de interés y seleccione `Make Mandatory` como se ilustra en la figura C.14.

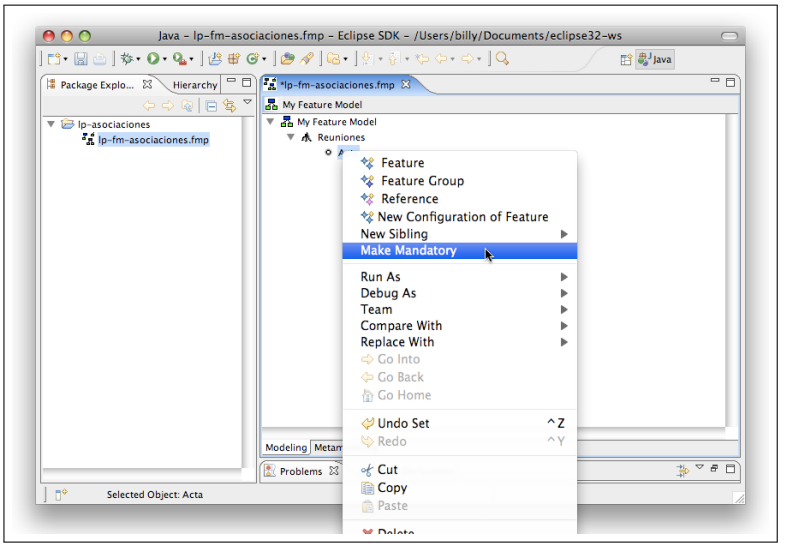


Figura C.14: Cambio de carácter optativo a obligatorio de una característica

No hay una opción directa para crear una característica clonable. El proceso se basa en la modificación de la cardinalidad de una característica existente. Por lo tanto, cree la característica que vaya a convertir, y a continuación obtenga el menú desplegable sobre la característica que acaba de crear (o una característica existente que desee convertir) y pulse `Show Properties View` tal y como se muestra en la ilustración de la figura C.15.

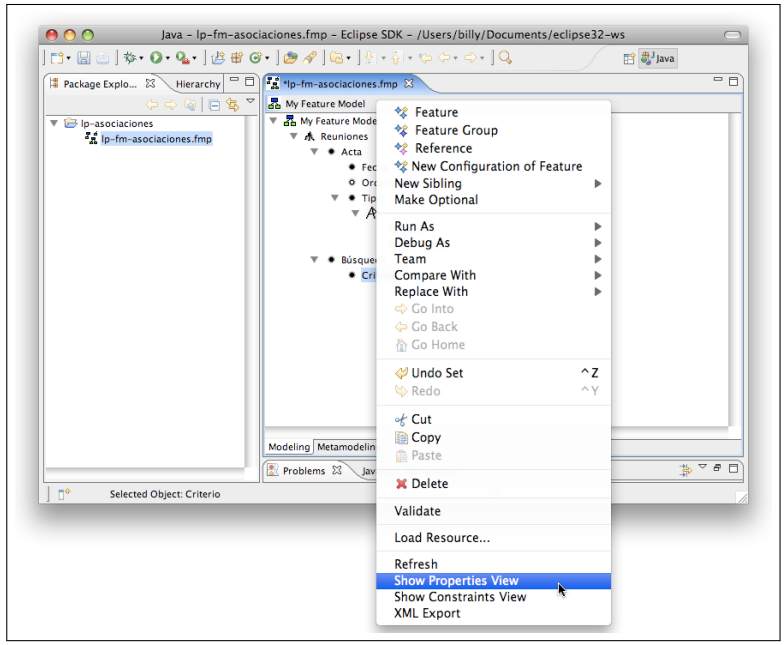


Figura C.15: Obtención del menú de propiedades de una característica

Observe también en la misma figura C.15 que desde el menú desplegable se puede seleccionar la opción `Show Constraints View` (además muestra más información de la que se refleja en el modelo gráfico), que servirá para definir restricciones sobre las características modelo de modo que el *plug-in* las valide (no olvidando las limitaciones de las comprobaciones automáticas de *fmp*).

Aprovechando la figura C.15, la opción `New Sibling` permite crear una característica, grupo de características, referencia o configuración al mismo nivel que la característica sobre la que se ha obtenido el menú desplegable.

Ttambién es posible generar características de grupo, eligiendo la opción `Feature Group`. A continuación aparecerá un símbolo que simplemente es un elemento gráfico separador (no es posible asignar ningún identificador o nombre, tampoco tiene sentido porque no es más que un arco) sobre el que deberá obtener de nuevo el menú desplegable para incluir las características que formarán parte del grupo.

Nótese, que, por defecto, las características de grupo son creadas como características de tipo XOR (un arco vacío), si desea cambiar a tipo OR (un arco negro) o establecer una cardinalidad concreta tendrá que acceder al submenú de propiedades del símbolo y modificar los parámetros `Min` y `Max`.

Para convertir una característica en clonable, basta con pulsar dos veces (al igual que hacía para cambiar el nombre de una característica) sobre el parámetro `Max` y escribir "-1". Automáticamente se producirán los cambios tal y como se muestra en la ilustración de la figura C.16.

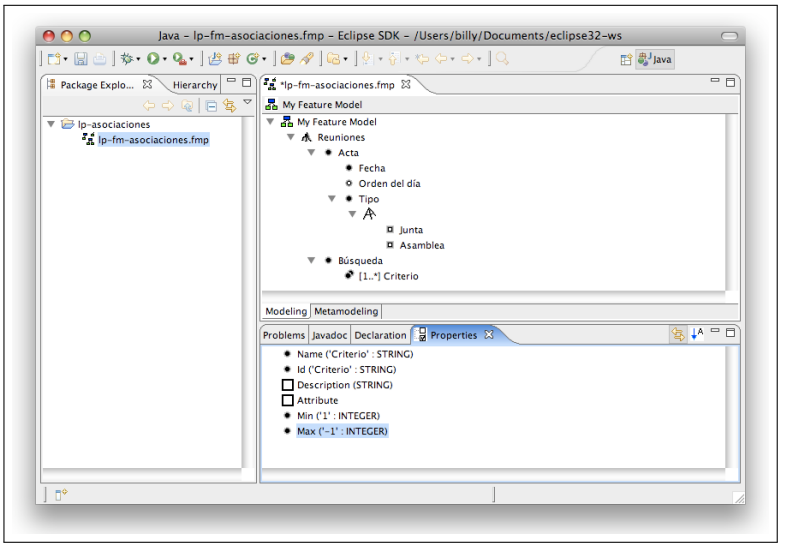


Figura C.16: Cambio de la cardinalidad de una característica en el menú de propiedades

En la ilustración de la figura C.17 se muestran las cardinalidades mínimas y máximas de las características opcionales, 0 y 1 respectivamente. Fíjese que el identificador de la característica `Id` debe ser un nombre sin acentos ni espacios entre las letras.

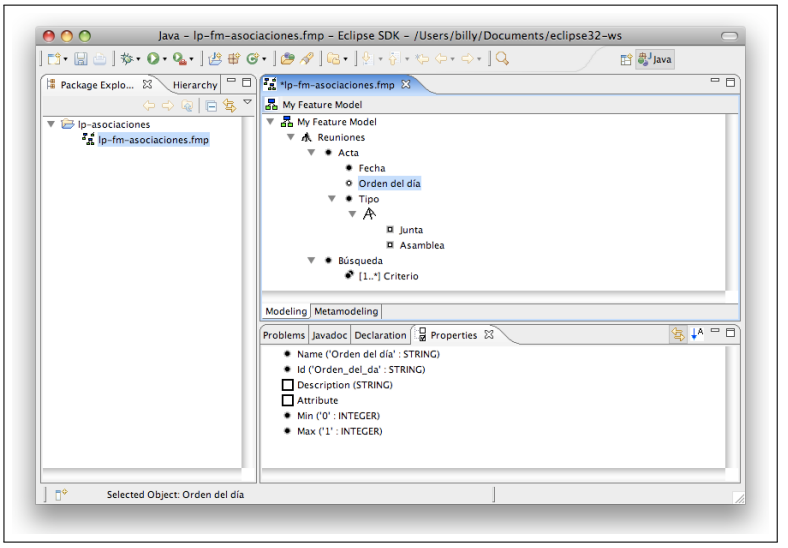


Figura C.17: Cardinalidad de una característica opcional

Para incluir restricciones, una vez seleccionado `Show Constraints View`, aparecerá el submenú que se muestra en la ilustración de la figura C.18. Obtenga el menú desplegable sobre `Additional constraints` y lance el asistente.

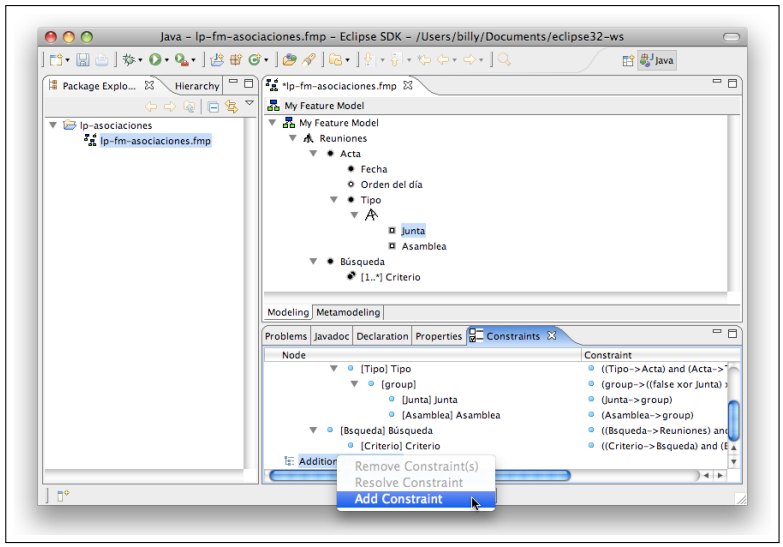


Figura C.18: Menú de propiedades de una característica

La ilustración de la figura C.19 muestra una ventana de error del *plug-in* que por desgracia suele ser bastante frecuente y poco descriptiva pues el *log* no aporta mucha más información de utilidad.

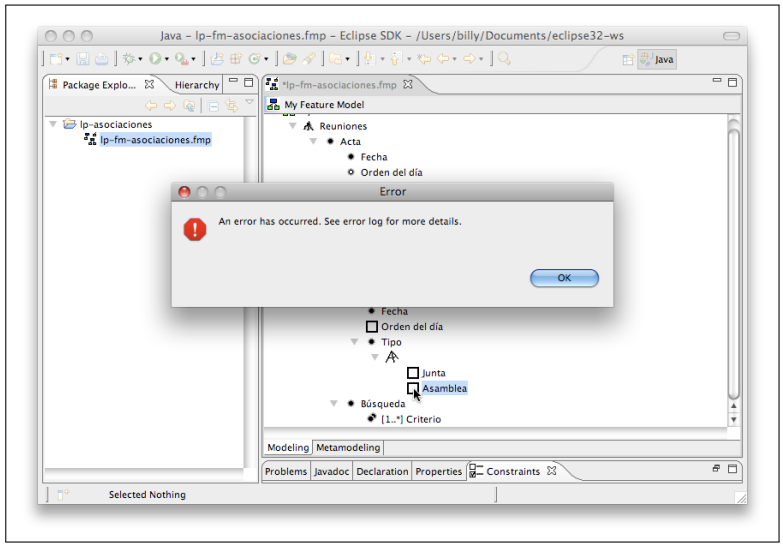


Figura C.19: Mensaje de error de *Feature Model Plug-In*

C.2.2. Exportación a XML

Para obtener la representación en XML del modelo de características basta con que obtenga el menú desplegable (pulsando con el botón derecho sobre la característica de la que desee obtener el esquema (generalmente se suele hacer sobre la característica raíz para que englobe a todas las subcaracterísticas).

En la ilustración de la figura C.15 se mostraba el proceso de obtener este menú, seleccionando `XML Export` obtendrá lo que se ilustra en la figura C.20, es decir, el resultado de la exportación, que habrá que copiar y pegar (el texto de la ventana emergente es seleccionable) en un fichero de texto.

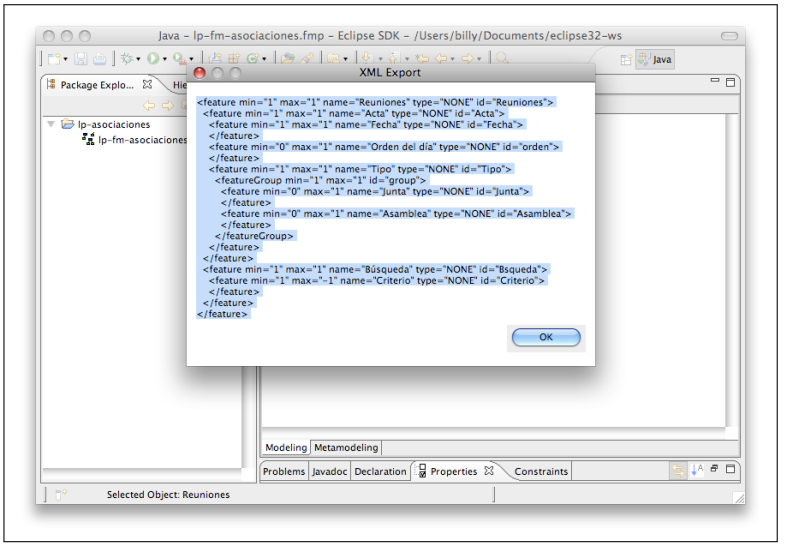


Figura C.20: Representación en XML del modelo de características

Apéndice D
Contenido del disco

"Cada generación se ríe de las viejas modas, pero sigue rigurosamente las nuevas"

Henry David Thoreau

Los contenidos del disco que acompaña a este trabajo son los que se muestran en la ilustración de la figura D.1.

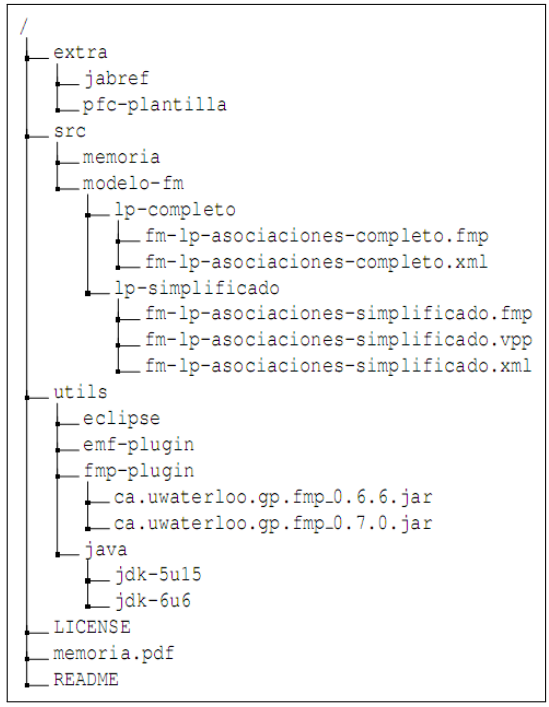


Figura D.1: Árbol de ficheros del disco

A continuación se describen brevemente, de mayor a menor importancia (es decir, el orden de descripción no se corresponde con el de la ilustración), los contenidos del disco (algunos directorios no han sido desglosados, y por tanto tampoco descritos aquí con detalle, en los subdirectorios o ficheros que puedan contener ya que no se consideran relevantes).

- **README**: fichero de texto con una versión reducida de los contenidos de este apéndice.
- **LICENSE**: fichero de texto con los términos de uso de la licencia en castellano del trabajo realizado.
- **memoria.pdf**: versión electrónica en formato PDF de esta memoria.
- **src**: directorio con los contenidos de los ficheros fuente creados para las diferentes partes de este Proyecto Fin de Carrera.
  - **memoria**: ficheros fuente en LaTeX e imágenes utilizados para la generación de esta memoria.
  - **modelo-fm**: ficheros de los modelos de características de la línea de productos para asociaciones (sólo se podrán abrir con el *plug-in* de *Eclipse*: *fmp*).
    - **lp-completo**: versión completa del modelo de características de la línea de productos para asociaciones.
      - **fm-lp-asociaciones-completo.fmp**: fichero con el modelo de características completo de la línea de productos.
      - **fm-lp-asociaciones-completo.xml**: representación en XML del fichero anterior.
    - **lp-simplificado**: versión simplificada del modelo de características de la línea de productos para asociaciones.
      - **fm-lp-asociaciones-simplificado.fmp**: fichero con el modelo de características simplificado de la línea de productos.
      - **fm-lp-asociaciones-simplificado.vpp**: versión electrónica de los diagramas de casos de uso y de clases del dominio mostrados en el capítulo 4.

        Se han realizado con la herramienta *CASE*, *Visual Paradigm for UML 6.2 Community Edition*.
      - **fm-lp-asociaciones-simplificado.xml**: representación en XML del fichero con el modelo de características simplificado.
- **utils**: directorio con los programas utilizados y los complementos necesarios para poder ejecutarlos.
  - **eclipse**: entorno de desarrollo utilizado como soporte tecnológico por el *plug-in fmp*.

    Se entrega la versión 3.2.2 ya que es la más actual manteniendo la compatibilidad con el *plug-in fmp*.
  - **emf-plugin**: *framework* de modelado de uso imprescindible por el *plug-in fmp* para la construcción gráfica de los modelos de características.
  - **fmp-plugin**: *plug-in* utilizado para la realización de los modelos de características.
    - **ca.uwaterloo.gp.fmp_0.6.6.jar**: versión 0.6.6 del *plug-in fmp*.

- **ca.uwaterloo.gp.fmp_0.7.0.jar**: versión 0.7.0 del *plug-in fmp*.

  Se recomienda su uso, aunque también se advierte que es relativamente inestable, con lo que se aconseja grabar cualquier cambio cada poco tiempo).

- **java**: plataforma de soporte al lenguaje de programación *Java* imprescindible para poder ejecutar *Eclipse*.
  - **jdk-5u15**: versión 1.5.0u15 de la plataforma *Java*.

    Se entrega tanto la maquina virtual (JRE) como el kit de desarrollador (SDK).

    Con esta versión el *plug-in* ha sido bastante más estable en la versión 10.5.3 del sistema operativo del fabricante *Apple*: *MacOS X*, que en otros sistemas operativos de otros fabricantes.
  - **jdk-6u6**: versión 1.6.0u6 de la plataforma *Java*.

    Se entrega tanto la maquina virtual (JRE) como el kit de desarrollador (SDK).

- **extra**: directorio con contenidos de interés para el lector, aunque no relacionados directamente con el Proyecto Fin de Carrera.
  - **jabref**: gestor de referencias bibliográficas *open source* en formato BIBTEX. Requiere de la plataforma de soporte a *Java*.
  - **pfc-plantilla**: ficheros de la plantilla, en LATEX, de escritura de Proyecto Fin de Carrera adaptada a la normativa de la E. T. S. de Ingeniería Informática (Universidad de Valladolid).