ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

Aplicación De La Arquitectura Dirigida Por Modelos A Las Líneas De Producción De Software

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

AUTORES:

Freddy Mauricio Inga Aguilera
freddy.ingaec@gmail.com

Nila Pamela Sarabia Maldonado
npsm_1@yahoo.es

DIRECTOR:

Msc. Ing. Raúl Córdova
raul.cordova@epn.edu.ec

Quito, julio 2013

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Freddy Mauricio Inga Aguilera y Nila Pamela Sarabia Maldonado, bajo mi supervisión.

_____

Msc. Ing. Raúl Córdova
DIRECTOR DEL PROYECTO

DECLARACIÓN

Nosotros, Freddy Mauricio Inga Aguilera y Nila Pamela Sarabia Maldonado declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Freddy Mauricio Inga Aguilera **Nila Pamela Sarabia Maldonado**

AGRADECIMENTOS

Me siento profundamente agradecido con todas las personas que se han cruzado en mi vida universitaria, compañeros de clases que me permitieron vivir experiencias inolvidables, especialmente a los ingenieros que generosamente impartieron su sabiduría para lograr culminar con éxito esta maravillosa carrera y a mi querida familia que me han inspirado, apoyado y contribuido muchas gracias.

Freddy

AGRADECIMENTOS

Agradezco a Dios por sus bendiciones y ser mi compañía en cada instante de mi vida, por poner en mi camino a personas que me han ayudado a trazar un sendero por el cual caminar en lo personal y lo profesional.

Gracias a mis Padres que estuvieron día a día viendo crecer mis expectativas de vida, apoyándome sin importar las dificultades, a mis hermanos que me alentaron a no desmayar, a mis amados sobrinos que son una indiscutible fortaleza en mi vida.

Agradezco a mis compañeros, profesores y amigos que siempre me alentaron a culminar esta meta, en especial al Msc. Ing. Raúl Córdova director de esta tesis, quien nos ha proporcionado todo su conocimiento y apoyo incondicional.

A ti mi amado esposo un gracias infinito por tu paciencia y amor.

Pamela

DEDICATORIA

Dedico esta tesis a mis padres y hermanos, por el total apoyo incondicional, su muestra de perseverancia, paciencia y amor me alentaron en todo momento de mi carrera universitaria para continuar sin desistir, a mi esposa e hijo que sirvieron de inspiración para lograr tal objetivo deseado, culminar mi carrera.

Freddy

DEDICATORIA

Este trabajo que ha sido merecedor de horas de esfuerzo y dedicación lo dedico a mis padres pilar esencial de mi vida en lo profesional y personal, mi tía hermanos y sobrinos que son compañía certera en momentos de alegría y tristeza.

Dedicado a mi esposo quien es para mí un leal amigo, que con su amor me ha acompañado en esta última etapa de mi carrera.

A mi Aylin mi hija adorada, que este trabajo represente uno de los tantos esfuerzos que como madre quise hacer para ser mejor por mí y por ti, porque te amo.

Pamela

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 PLANTEAMIENTO DEL PROBLEMA

1.1 IDENTIFICACIÓN Y DESCRIPCIÓN DEL PROBLEMA

En la actualidad existen muchos centros en los que el desarrollo de software tiene un alto porcentaje de trabajo artesanal por lo que requiere un gran despliegue de recursos, principalmente de tiempo y costo, convirtiéndose en un problema para las empresas desarrolladoras. Hoy existen técnicas modernas como la Arquitectura Dirigida por Modelos (MDA) y las Líneas de Producción de Software (LPS), que se perfilan como una conjunción de conocimiento y metodología, las mismas que permiten por separado altos porcentajes de reutilización, mejoras en el proceso de desarrollo, reduciendo el esfuerzo y el costo. Sin embargo, estas técnicas se han aplicado de manera aislada, por lo que se hace necesario integrarlas para obtener mayores beneficios.

La industrialización del proceso de software facilita la evaluación, medición y control del proceso, y con ello, su mejora y adaptación al cambio, no sólo en el análisis de los procesos internos, sino en la investigación de nuevas tecnologías, herramientas y métodos.

Debido a las necesidades que se han generado en torno a estas tecnologías, las organizaciones están requiriendo cada vez, herramientas más flexibles y adaptables a su infraestructura operativa, situación que no siempre es sencilla de superar.

La tecnología MDA se ha convertido en una necesidad para muchas de las compañías, ya que se centra primero en la funcionalidad y el comportamiento de una aplicación distribuida o sistema, no distorsionado por la plataforma tecnológica o plataformas en las que se llevarán a cabo el desarrollo. De esta manera, MDA divorcia detalles de la implementación de funciones de negocios.

Por lo tanto, no es necesario repetir el proceso de definición de una aplicación o la funcionalidad del sistema y el comportamiento cada vez que una nueva tecnología se presente, por ejemplo Web Services.

Otras arquitecturas están ligadas generalmente a una tecnología en particular. Con MDA, la funcionalidad y el comportamiento se modelan una vez y sólo una vez.

Además en la actualidad la resolución de problemas puntuales y software a la medida presenta una serie de dificultades, si se considera que muchas veces existen organizaciones que requieren soluciones similares a algunas previamente elaboradas, pero que por la misma concepción del software resulta costoso y complejo realizar cualquier adaptación.

Como una solución, el enfoque de líneas de producción, pretende ofrecer un conjunto de aplicaciones de software que permitan generar de forma automática software con funcionalidad básica a partir de un modelo de negocio previamente definido. Para lograr este objetivo global, es importante identificar, entre otros, las variaciones que soportará la línea de producción y establecer la infraestructura adecuada que permita generar productos a bajo costo conservando alta calidad.

En torno a las LPS existen diversas tecnologías que están siendo utilizadas y que proporcionan los elementos necesarios para dar soporte a los diversos procesos involucrados, convirtiendo a las LPS en una estrategia, que cuando se aplica hábilmente, puede producir muchos beneficios y en última instancia, dar a las organizaciones una ventaja competitiva.

1.2 CARACTERÍSTICAS DE MDA PARA DESARROLLO DE SISTEMAS SOFTWARE

Las metodologías de desarrollo de software se están encaminando al uso de modelos no como mera documentación del proceso, sino como elementos claves a la hora de obtener una implementación definitiva. A fin de poseer un estándar común, el Object Management Group (OMG) definió MDA (Model Driven Architecture) para impulsar el desarrollo de software guiado de modelos conceptuales [1].

En esencia MDA es una nueva metodología para la definición de especificaciones y el desarrollo de aplicaciones, que se basa en la transformación de diferentes modelos que van refinando la funcionalidad de una aplicación hasta la obtención del código final.

Es preciso distinguir MDA de MDD (Model Driven Development), que es una aproximación al desarrollo de software basado en el modelado conceptual y la generación de la aplicación a partir del mismo. Al ser una aproximación, sólo recomienda una estrategia general a seguir pero sin indicar plataformas, estándares o procesos que ayuden a cumplirla [2].

MDA se fundamenta en tres ideas principales:

1. **Representación directa:** se debe tratar de apartar el desarrollo de software del dominio de la solución o representación tecnológica del mismo, para centrarse en las ideas y conceptos del dominio del problema, reduciendo la distancia semántica con el dominio del problema y haciendo que su representación permita obtener soluciones más precisas a los problemas planteados.
2. **Automatización:** Utilizar herramientas para mecanizar todas aquellas tareas del desarrollo del software que no necesiten de la creatividad humana. Para ello se debe enfatizar la transformación entre modelos y la generación automática de código a partir de éstos.
3. **Estándares abiertos:** El hecho que los estándares sean abiertos facilita que sean adaptados por la industria, que sea posible la interoperabilidad y que el usuario final tenga posibilidades de elección.

MDA resuelve los retos de los sistemas actuales altamente conectados y constantemente cambiantes, tanto en reglas de negocio como en tecnología, proponiendo un marco de trabajo para una arquitectura que asegura [3]:

- Portabilidad, aumentando el re-uso de las aplicaciones y reduciendo el costo y complejidad del desarrollo y administración de las aplicaciones.

- Interoperabilidad entre plataformas, usando métodos rigurosos para garantizar que los estándares basados en implementaciones de tecnologías múltiples tengan todas idénticas reglas de negocio.
- Independencia de plataforma, reduciendo el tiempo, costo y complejidad asociada con aplicaciones desplegadas en diferentes tecnologías.
- Especificidad del dominio, a través de modelos específicos del dominio, que permiten implementaciones rápidas de aplicaciones nuevas, en una industria específica sobre diversas plataformas.
- Productividad, permitiendo a los desarrolladores, diseñadores y administradores de sistemas usar lenguajes y conceptos con los que se sienten cómodos, facilitando la comunicación e integración transparente entre los equipos de trabajo.
- Puntos de vista del sistema. MDA especifica tres puntos de vista en un sistema: independiente de cómputo, independiente de plataforma y específico de plataforma. El punto de vista independiente de cómputo se enfoca en el contexto y los requisitos del sistema, sin considerar su estructura o procesamiento. El punto de vista independiente de plataforma se enfoca en las capacidades operacionales del sistema fuera del contexto de una plataforma específica, mostrando sólo aquellas partes de la especificación completa que pueden ser abstraídas de la plataforma. El punto de vista dependiente de plataforma agrega al punto de vista independiente los detalles relacionados con la plataforma específica. Una plataforma es un conjunto de subsistemas y tecnologías que proveen un conjunto coherente de funcionalidad por medio de interfaces y patrones de uso. Los clientes de una plataforma hacen uso de ella sin importarles los detalles de implementación. Estas plataformas pueden ser sistemas operativos, lenguajes de programación, bases de datos, interfaces de usuario, soluciones de middleware, etc.
- Independencia de Plataforma. Cualidad que un modelo puede exhibir cuando es expresado independientemente de las características de otra plataforma.
- Modelo de Plataforma. Describe un conjunto de conceptos técnicos de una plataforma, representando sus elementos y los servicios que

provee. También especifica las restricciones en el uso de estos elementos y servicios por otras partes del sistema.

- Transformación de Modelos. Proceso de convertir un modelo a otro dentro del mismo sistema. Una transformación combina el modelo independiente de plataforma con una definición de otra plataforma, para producir un modelo específico a una plataforma correspondiente a un punto de vista del sistema, como interfaz de usuario, información, ingeniería, arquitectura, etc.

1.2.1 CICLO DE VIDA DEL DESARROLLO DE MDA

El ciclo de vida del desarrollo de MDA, no difiere mucho del enfoque tradicional, las mismas fases son identificadas como se ve en la Figura 1.1, una de las principales diferencias radica en la naturaleza de los artefactos que se crean durante el proceso de desarrollo. Los artefactos son los modelos formales, es decir, modelos que pueden ser comprendidos por los ordenadores. Los siguientes tres modelos son el núcleo fundamental de MDA.

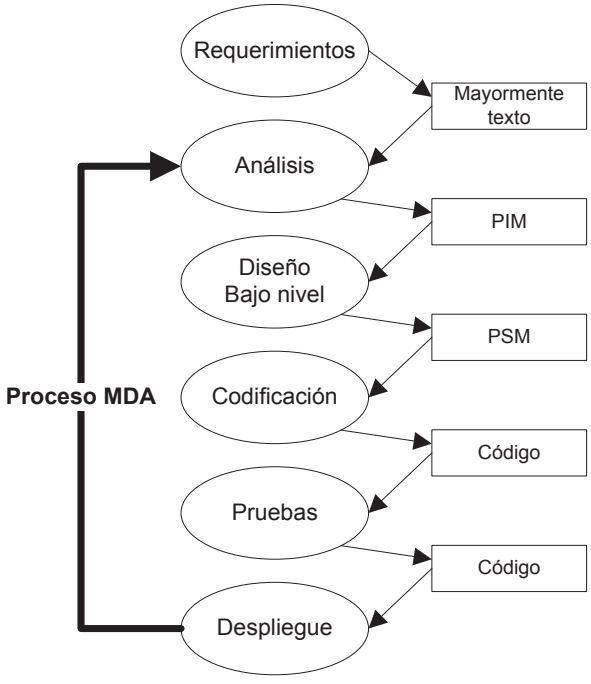


Figura 1.1 Ciclo de vida del desarrollo de software MDA

Fuente: (Guambo, 2006)

MDA especifica tres modelos básicos de un sistema, correspondientes a los puntos de vista expresados anteriormente. Estos modelos pueden verse como niveles de abstracción donde en cada nivel pueden construirse varios modelos

Modelo Independiente de Cómputo. Al modelo independiente de cómputo (Computation Independent Model o CIM) se le conoce como el modelo del dominio o del negocio, porque se modela en términos familiares a los expertos del negocio, representa exactamente lo que se espera del sistema, sin contemplar toda la información relacionada con la tecnología a fin de mantenerse independiente de cómo será implementado el sistema. Este modelo salva el abismo existente entre los expertos del negocio y los responsables de las tecnologías de información. En una especificación MDA el modelo CIM debe ser rastreable a las construcciones que lo implementan ya sean independientes o específicas a una plataforma.

Modelo Independiente de Plataforma. El modelo independiente de plataforma (Platform Independent Model o PIM) exhibe un grado de independencia tal que permite mapearlo a una o varias plataformas, Esto se logra definiendo una serie de servicios abstrayéndolos de los detalles técnicos para que otros modelos especifiquen cómo será la implementación.

Modelo Específico de Plataforma. El modelo específico de plataforma combina la especificación de un PIM con los detalles para indicar como el sistema usa una plataforma en particular. Para llegar a los objetivos señalados es importante especificar que, se debe partir de los modelos que permiten la abstracción de un software, estos modelos son especificaciones pertenecientes a la OMG.

Código. El paso final en el desarrollo es la transformación de cada PSM a código. Debido a que un PSM se ajusta a su tecnología, esta transformación es relativamente sencilla.

En la aproximación MDA, en primer lugar se modela un CIM en términos familiares a los expertos del negocio, luego un PIM, que debe capturar la funcionalidad de nuestra aplicación de forma totalmente independiente de la

plataforma tecnológica en la cual será desarrollada. Dicho PIM deberá ser transformado definiendo las correspondientes reglas, en un PSM que estará basado en las características concretas de una plataforma de implementación. A partir de dicho PSM será de donde se aplicará una transformación definitiva al código de la aplicación. Por lo tanto básicamente una aplicación MDA consiste en un PIM y uno o varios PSM's para cada una de las plataformas.

1.2.2 HERRAMIENTAS DE TRANSFORMACIÓN

El proceso MDA es muy parecido al desarrollo tradicional. Sin embargo, hay una diferencia crucial. Tradicionalmente, las transformaciones de modelo a modelo, o de modelo a código, se hacen principalmente a mano. Muchas herramientas pueden generar algo de código de un modelo, pero que generalmente no va más allá de la generación de algún código de la plantilla, donde la mayor parte de la labor todavía tiene que ser llenado a mano.

En cambio, las transformaciones MDA se ejecutan siempre por herramientas como se muestra en la Figura 1.2. Muchas herramientas son capaces de transformar un PSM en el código. Teniendo en cuenta el hecho de que el PSM está ya muy cerca del código.

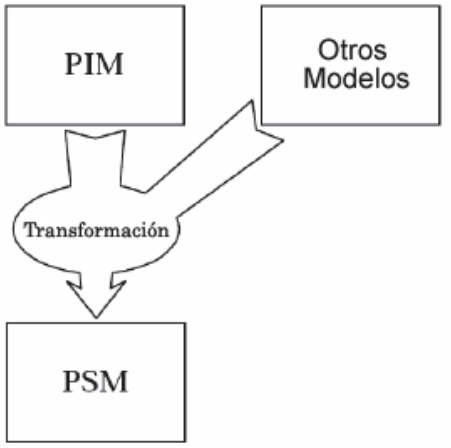


Figura 1.2 Transformación de modelos

Fuente: (Vallecillo)

Para poder cumplir el ciclo de transformación de modelos se han propuesto estándares que permiten una buena aplicación de MDA. La OMG ha propuesto y desarrollado los siguientes estándares: MOF (Meta Object Facility), UML, OCL (Object Constraint Language), XML (XML Metadata Interchange), QVT (Query/Views/Transformations). Estos estándares son un conjunto de lenguajes que permiten el diseño y modelado, lo cual ayuda a desarrollar los modelos iniciales para la transformación de modelos a modelos y modelos a código.

1.2.3 META OBJECT FACILITY (MOF)

El Meta modelado utilizado en MOF básicamente consiste en usar modelos para describir otros modelos.

El meta modelado de un lenguaje es una definición de sus elementos mediante conceptos y reglas de un lenguaje para crear modelos en dicho lenguaje de modelado. De acuerdo a esta definición y tomando al Lenguaje Unificado de Modelado, se dice que el meta modelado de UML define los conceptos y reglas para crear modelos UML Figura 1.3.

Niveles o capas de modelado que maneja OMG.

M0 Instancia: En el nivel M0 se encuentran las instancias reales del sistema, o los objetos de la aplicación provenientes de clases. Dentro de este nivel no se mencionan clases ni atributos únicamente los datos instanciados de las mismas.

M1 Modelo del Sistema: Este nivel se sitúa por encima de la capa M0, y representa el modelo de un Software. Dentro de esta capa se representan categorías de las instancias del nivel M0, es decir, los elementos de la capa M0 son instancias de los elementos de la capa M1.

M2 Metamodelo: De la misma manera como ocurre con los anteriores niveles, los elementos del nivel M1 son instancias del nivel M2; dentro de esta capa aparecen conceptos como Clase, Atributo o Relación.

M3 Meta metamodelo: Siguiendo lo mencionado anteriormente, los elementos del nivel M2 son instancias de este nivel M3 Meta metamodelo.

El lenguaje que define a la capa M3 es MOF (Meta Object Facility), lo que resulta que todos los metamodelos del nivel M2 son instancias de MOF, o de otra manera, UML se define usando MOF.

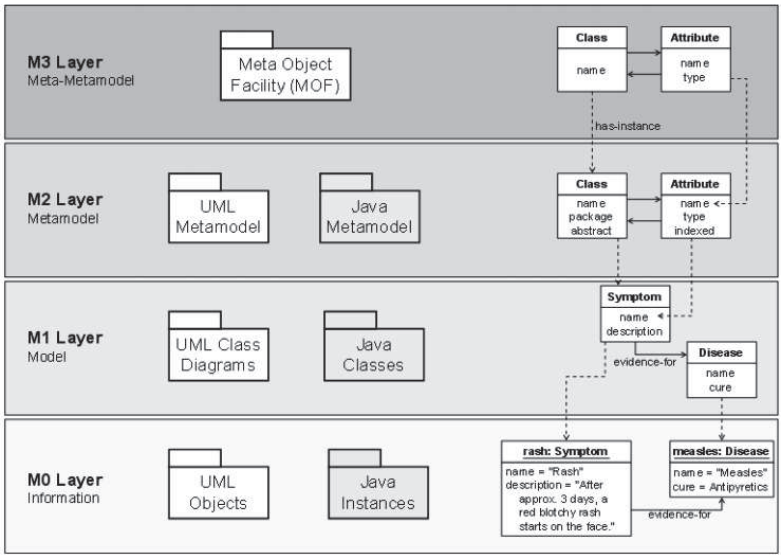


Figura 1.3 Estructura de 4 capas (MDA)

Fuente: (Protege, 2006)

1.2.4 UML LENGUAJE UNIFICADO DE MODELADO

UML es un lenguaje que permite especificar, construir, visualizar y documentar los artefactos o componentes de un sistema. Posee niveles claramente definidos para el proceso de desarrollo como se muestra en la Figura 1.4.

La meta de UML es convertirse en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones. En

UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo.

UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

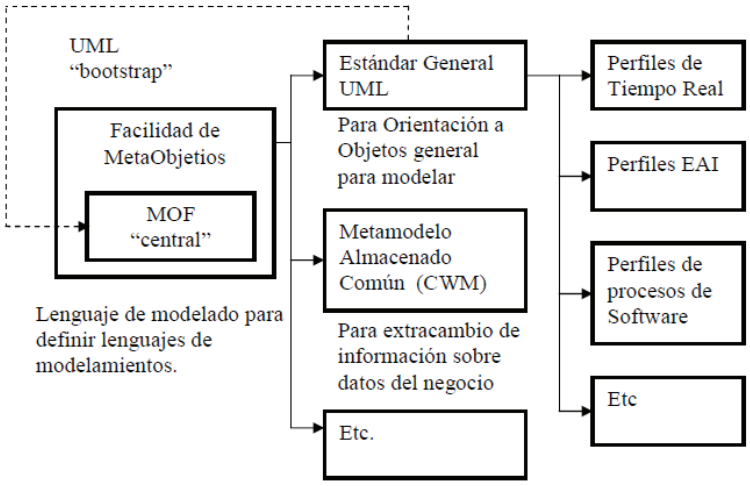


Figura 1.4 Niveles de UML

Fuente: (Alberto C. R., 2009)

1.2.5 OCL

Permite expresar instrucciones semánticas a los metamodelos. Es un complemento ideal para MOF, ya que MOF no puede añadir estas restricciones.

El OCL de la Figura 1.5 asegura que una tabla al heredar de otra herede también las columnas de sus superclases.

```
-- Una Tabla hereda las columnas de su superClases.

context Tabla inv:
   superclase.columna->forAll
      (superClassColumna | self.columna->incluyes
                           (superClassColumna) )
```

Figura 1.5 Redefinición de subclase

Fuente: (Alberto C. R., 2009)

1.2.6 XMI XML METADATA INTERCHANGE (XML DE INTERCAMBIO DE METADATOS)

Es una especificación para el Intercambio de Diagramas; la especificación para el intercambio de diagramas fue escrita para proveer una manera de compartir modelos UML entre diferentes herramientas de modelado. En versiones anteriores de UML se utilizaba un Esquema XML para capturar los elementos utilizados en el diagrama; pero este esquema no decía nada acerca de cómo el modelo debía graficarse.

1.2.7 QVT

El principal problema sobre MDA es la transformación entre modelos porque no existe un lenguaje estándar que realice las transformaciones. Sin embargo en la creación de lenguajes, la OMG definió que las propuestas de los lenguajes estándares tienen que ser compatibles con: UML, MOF, CWM, SPEM. Además el lenguaje debe cumplir con lo siguiente:

- Realizar transformaciones entre modelos de metamodelos definidos bajo MOF.
- Permitir la creación de la vista de un metamodelo.
- Permitir la flexibilidad en su implementación.

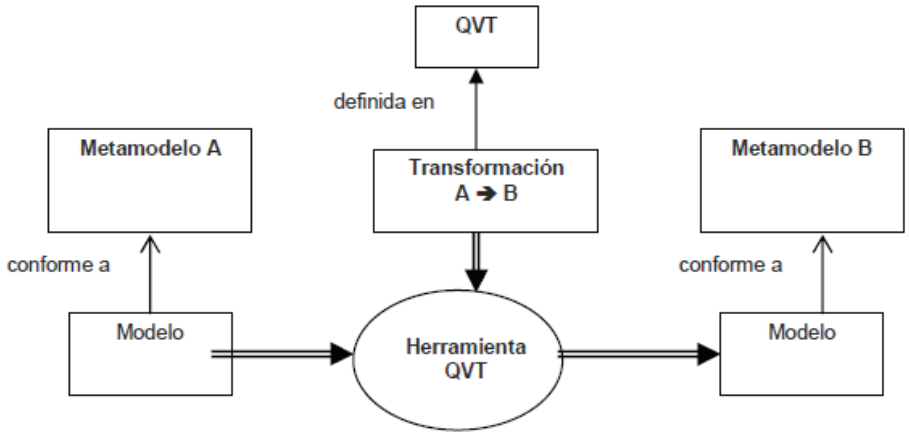


Figura 1.6 Proceso de transformación con QVT

Fuente: (Alberto C. R., 2009)

1.3 CARACTERÍSTICAS DE LAS LÍNEAS DE PRODUCCIÓN DE SOFTWARE PARA EL DESARROLLO DE SISTEMAS SOFTWARE

Un nuevo desarrollo puede implicar una configuración específica de componentes, implementación de componentes adicionales y adaptación de algunos componentes para satisfacer las nuevas demandas. Esta característica es lo que hace a las LPS una de las aproximaciones más efectivas para la reutilización de software.

Las LPS parten de la idea de la producción en serie, dando la capacidad a una organización de producir o generar sistemas de software en forma masiva, satisfaciendo las demandas del mercado, entre otros beneficios significativos. La adaptación masiva consiste en la capacidad para generar productos que comparten en gran parte características comunes, pero también mantienen características propias de cada uno de ellos.

1.3.1 LÍNEAS DE PRODUCTOS DE SOFTWARE

Una LPS es: “un conjunto de sistemas de software compartiendo características comunes y administradas que satisface las necesidades específicas de un segmento de mercado particular o misión y que son desarrolladas de forma pre-escrita a partir de un conjunto común de elementos clave” [4]. Como lo indica la definición, una LPS consiste en un conjunto de elementos clave para producir sistemas de software que comparten características comunes (llamadas similitudes), pero al mismo tiempo mantienen características propias (llamadas variabilidad). Para proporcionar estas características individuales a cada miembro de la familia, la adaptación masiva debe ser lo suficientemente flexible para satisfacer tal necesidad, pues además el número distinto de miembros está limitado a una serie de restricciones las cuales también deben satisfacerse; a ésto se le llama administración de la variabilidad.

1.3.2 FASES O ETAPAS PARA IMPLEMENTAR UNA LPS

En una LPS se propicia una reutilización a alto nivel, desarrollando los elementos comunes que servirán como base para la generación de cada miembro de la familia. Las etapas esenciales a ejecutar son tres:

- **Ingeniería de dominios** en la cual se desarrollan los elementos comunes.
- **Ingeniería de aplicaciones** donde se generan los sistemas que son miembros de la familia. Dichos miembros se generan a partir de los elementos comunes, pero tomando en cuenta el modelo de variabilidad definido.
- **Conjunto de actividades** que orquestarán las dos etapas anteriores. Es decir, se necesita de un proceso para ejecutar de manera ordenada la ingeniería de dominios y la ingeniería de aplicaciones.

Las dos primeras etapas están compuestas por las fases tradicionales del modelo de cascada (Requerimientos, Diseño, Implementación, Pruebas) pero con objetivos distintos como ya se mencionó.

1.3.3 ENFOQUES DE ADOPCIÓN DE UNA LPS

Se han planteado varios enfoques como estrategia para realizar la adopción de una LPS, entre ellos algunos tienen nombre distinto de acuerdo al “framework” o metodología en la que están contenidos, pero consisten en la misma estrategia. Básicamente, existen tres enfoques de adopción los cuales han sido reportados bajo otros nombres, sin embargo, en esencia son lo mismo [5]. Dichos enfoques son:

Proactivo: funciona cuando la implantación de la LPS se hace desde cero, es decir que aún no se cuenta con sistemas de software que pertenecerán a la familia.

Extractivo: inicia con la selección de uno o más sistemas ya existentes, que serán parte de la familia de productos; efectuando un tipo de ingeniería inversa para extraer los artefactos de software comunes para establecerlos como elementos comunes y modelar la variabilidad que existe entre ellos.

Reactivo: toma la esencia del proceso de espiral o iterativo para efectuar la transición poco a poco. Se realizan los pasos como en el enfoque proactivo, pero para cada ciclo o espiral, de esta manera se van eliminando riesgos y se va aclarando la visión de las similitudes y variabilidades de los productos que serán miembros de la familia.

1.3.4 BENEFICIOS DE UNA LPS

La implantación de una LPS no es una tarea sencilla, sino que requiere un gran esfuerzo y sobre todo una gran inversión para su éxito. Sin embargo, los beneficios obtenidos motivan a realizarlo.

Existen muchos beneficios que una LPS puede proporcionar, destacando los siguientes entre los más relevantes:

Obtención de las ganancias de una producción a gran escala. Como se mencionó en la adaptación masiva, una LPS habilita a producir sistemas de software a partir de una plataforma de artefactos comunes de manera que los productos sólo necesiten ensamblarse en vez de ser desarrollados individualmente.

Mejora la capacidad para el "Time-to-market". Muchas veces las demandas del mercado exigen tener estrategias de mercadotecnia que implican tener productos o implementar ideas innovadoras antes de cierta fecha. Bajo un esquema tradicional de desarrollo de sistemas individualmente, dichas estrategias son demasiado difíciles de ejecutar haciendo perder a una organización la oportunidad de crecer en un mercado cambiante, pues como se dice por ahí: "el que pega primero pega dos veces". Con una LPS, una organización puede tomar el reto de establecer estrategias de mercadotecnia agresivas, ya que cuentan con la capacidad de generar productos de forma acelerada. En la Figura 1.7 se muestra una gráfica que representa el tiempo de desarrollo de sistemas de software con y sin una LPS, donde el eje horizontal representa el número de sistemas distintos desarrollados, y el eje vertical el tiempo de liberación en el mercado. Como se puede ver al inicio, el tiempo de desarrollo de la LPS hace que éste sea mucho mayor que el desarrollo de los primeros sistemas individuales ,debido a que primero se deben desarrollar los

elementos comunes, sin embargo, conforme se empiezan a desarrollar más sistemas, el tiempo de liberación en una LPS se reduce radicalmente en comparación al desarrollo individual, lo cual demuestra que a un mediano plazo los beneficios serán mucho mayores; pues como se mencionó, una organización se puede comprometer a liberar cierto producto de su LPS en menos de la mitad del tiempo que le llevaría desarrollarlo individualmente.

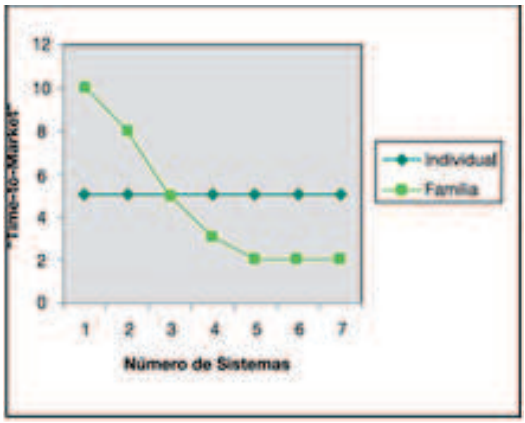


Figura 1.7 Time-To-Market con y sin una

Fuente: (Grajales, 2009)

Reducción de costos de desarrollo. Con una LPS los costos de desarrollo de los sistemas miembros de la familia, están muy por debajo de los costos de desarrollo de sistemas individuales. Al igual que en el “time-to-market”, al inicio el desarrollo de los elementos comunes tiene mayor costo que desarrollar un sistema individualmente. No obstante, una vez obtenidos, el costo disminuye. Se puede decir que la tasa de incremento en el costo acumulado es mucho menor en una LPS que en sistemas individuales.

Mejora de la calidad de los productos. Desde que el desarrollo de los elementos comunes es similar al desarrollo de los artefactos para un sistema individual, dichos elementos son probados constantemente por cada producto generado, y por formar parte de todos los productos de la familia se puede decir que la calidad es más fácil y fielmente asegurada en todos ellos.

Logro de las metas de reutilización de la organización. Dado que las LPS se basan en la reutilización de artefactos para un conjunto de sistemas, la

reutilización planeada se obtiene al desarrollar la plataforma de elementos comunes.

Reducción de la necesidad de contratar nuevo personal. Desde que el proceso de generación de los productos de software se automatiza cada vez más, la necesidad de contratar gente disminuye debido a que dichos productos son generados cada vez con menos ingenieros de software, lo cual de alguna manera ayuda a disminuir los costos a la organización.

Incremento de la satisfacción del cliente. Por varios de los beneficios mencionados anteriormente, los clientes y usuarios de los sistemas generados con la LPS obtienen productos de software de mucha mayor calidad a precios mucho más bajos que los que obtendrían con un proveedor que no cuenta con una LPS.

2 DEFINICIÓN DEL PROCESO DE DESARROLLO BASADO EN LPS Y MDA

2.1 ARQUITECTURA DIRIGIDA POR MODELOS CON UML PARA LÍNEAS DE PRODUCCIÓN DE SOFTWARE

OMG mantiene a UML como estándar. En la visión de OMG, “Modelado es el diseño de aplicaciones de software antes de la codificación.” OMG promueve la arquitectura dirigida por modelos como el enfoque en que los modelos UML de la arquitectura de software se desarrollan antes de la implementación. De acuerdo con OMG, UML es independiente de la metodología; UML es una notación para describir los resultados de un análisis orientado a objetos y diseño desarrollado a través de la metodología de elección.

Un modelo UML puede ser un modelo independiente de la plataforma (PIM) o un modelo específico de la plataforma (PSM). El modelo independiente de la plataforma es un modelo preciso de la arquitectura de software antes de comprometerse con una plataforma específica. El desarrollo de la primera PIM es particularmente útil debido a que el mismo PIM se pueda designar a diferentes plataformas middleware, tales como: COM, CORBA, .NET, J2EE, Web Services u otra plataforma [6].

Así utilizando el concepto de MDA se desarrollará una arquitectura de software para una LPS basada en componentes, expresándolo como un modelo UML independiente de la plataforma.

El método de diseño de LPS tiene que extender los conceptos de un solo sistema de análisis y diseño del modelo de la línea de productos, en particular para modelar el carácter común y variabilidad de la línea de productos y para extender la notación UML para describir esta comunidad y variabilidad.

2.1.1 MODELAMIENTO DE CASOS DE USO

Para una LPS, que consta de una familia de sistemas, sólo algunos requisitos son comunes a todos los miembros de la familia. Para especificar los requisitos funcionales de una LPS, es importante capturar los requisitos comunes de la línea de productos, es decir, los requisitos que son comunes a todos los miembros de la familia, así como los requisitos opcionales y alternativos.

Cuando una línea de producción está siendo modelada, por lo general sólo algunos de los casos de uso y actores son requeridos por un determinado miembro de la línea de productos. Los casos de uso que son requeridos por todos los miembros de la línea de productos se conocen como **casos de uso núcleo**. Los **casos de uso opcionales** son requeridos por algunos pero no todos los miembros de la línea de productos. Algunos pueden ser **casos de uso alternativos**, es decir, diferentes versiones del caso de uso son requeridos por los diferentes miembros de la línea de productos. Estos casos de uso alternativos suelen ser mutuamente excluyentes.

2.1.1.1 Documentación de casos de uso para líneas de producción de software

Los casos de uso para una LPS se documentan como se lo hace normalmente en UML con pequeñas variaciones propias de las LPS que se deben agregar y que son la Reutilización de categoría y los Puntos de variación. A continuación se muestra la manera de describir un caso de uso en este entorno. [6]

Nombre de caso de uso. Texto que identifica al caso de uso.
Reutilización de categoría. En esta sección se especifica si el caso de uso es el núcleo, opcional o alternativo.
Resumen. Describe brevemente el caso de uso.
Actores. Nombra a los actores en el caso de uso. Siempre hay un actor principal que inicia el caso de uso y actores secundarios que pueden participar en el caso de uso.
Dependencia. Esta sección opcional describe si el caso de uso depende de otros casos de uso, es decir, si se incluye o se extiende a otro caso de uso.
Condiciones previas. En esta sección se especifican una o más condiciones que deben cumplirse antes de que el caso de uso comience.

Descripción. La mayor parte de la descripción en los casos de uso es una narrativa de la secuencia principal del caso de uso.
Alternativas. Proporciona una descripción narrativa de ramas alternativas de la secuencia principal. Puede haber varias ramas alternativas.
Puntos de variación. Define en la descripción de casos de uso los lugares donde se introduce una funcionalidad diferente para los diferentes miembros de la línea de producción.
Pos condición. En esta sección se detalla la condición en la cual queda el sistema después de que el caso de uso se ha ejecutado.
Cuestiones pendientes. En esta sección se documentan preguntas sobre el caso de uso para las discusiones con los usuarios.

2.1.1.2 Modelamiento de variabilidad en casos de uso

Una forma de manejar la variabilidad de casos de uso en LPS es a través de puntos de variación. Un punto de variación es una ubicación en un caso de uso donde un cambio puede tener lugar [7]. El término variación en este contexto se entiende como la situación que se maneja de manera diferente por los distintos miembros de la línea de producción.

2.1.1.3 Modelamiento de variaciones pequeñas

En algunas situaciones una variación pequeña afecta solo a una o dos líneas de producción en las descripciones de casos de uso.
Para documentar una variación pequeña se debe facilitar:

Nombre del punto de variación.
Tipo de funcionalidad que se insertará en el punto de variación: alternativa, opcional, obligatoria y alternativa opcional.
Línea en la descripción del caso de uso donde se encuentra la variabilidad introducida.
Descripción de funcionalidad insertada en el punto de variación.

2.1.1.4 Modelamiento de la variabilidad con relación de extensión

En ciertas situaciones, un caso de uso puede llegar a ser muy complejo, con muchas ramas alternativas. La relación de extensión se utiliza para modelar caminos alternativos que un caso de uso base puede tomar [6].

2.1.1.5 Puntos de extensión en líneas de producción de software

Los puntos de extensión se utilizan para especificar la ubicación exacta en el caso de uso base en la que las extensiones se pueden agregar [8].

A cada punto de extensión se le da un nombre. La extensión de un caso de uso puede tener uno o más segmentos de inserción. Cada segmento se inserta en el lugar de su punto de extensión en el caso de uso base. La relación de extensión con su estereotipo <<extend>> puede ser condicional, lo que significa que una condición que se define debe ser cierta para el caso de uso de extensión que se invoque. Así, es posible tener más de un caso de uso de extensión para el punto de misma extensión, pero con cada caso de uso la extensión satisface una condición diferente.

Considérese el ejemplo de una línea de productos de supermercado (Figura 2.1). Un punto de extensión llamado pago se declara en un caso de uso base llamado Pagar cliente. El caso de uso base trata de comprobar las compras de los clientes. Tres casos de uso de extensión tratan el tipo de pago efectuado: Pagar en efectivo, Pagar con tarjeta de crédito, y Pagar con tarjeta de débito. Consideremos dos situaciones diferentes en las líneas de productos. Si los tres casos de uso son extensión del núcleo, es decir, si cada miembro de la línea de productos proporciona estos tres tipos de pago, entonces sólo uno es necesario para proporcionar una condición de selección para cada caso de uso de extensión. Las condiciones de selección son mutuamente excluyentes [pago en efectivo], [pago con tarjeta de crédito], y [pago con tarjeta de débito], respectivamente. Durante la ejecución del caso de uso, dependiendo de cómo el cliente decide pagar, la condición de selección adecuada se establece en verdadera.

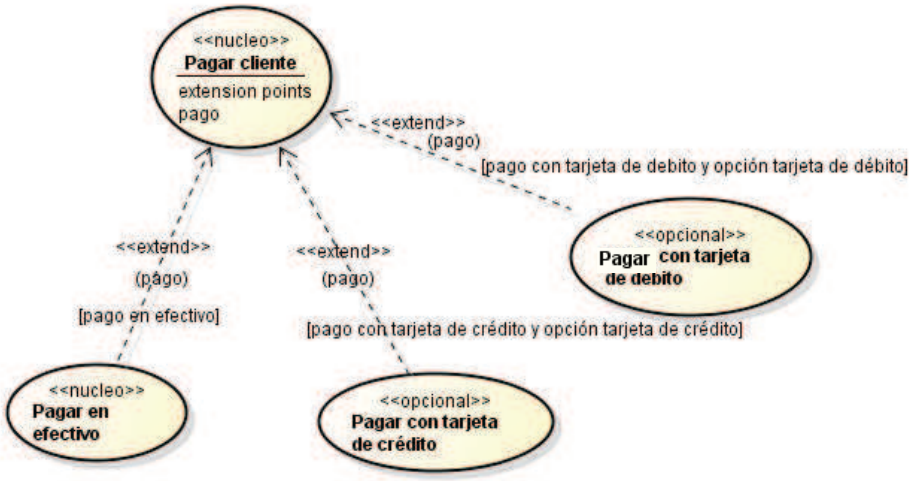


Figura 2.1 Ejemplo de una relación y casos de uso de extensión

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

Por otro lado, se debe tener en cuenta que *Pagar en efectivo* es el núcleo, es decir, cada miembro de la línea de producción ofrece esta característica y los casos de uso *Pagar con tarjeta de crédito* y *Pagar con tarjeta de débito* son opcionales, es decir, solo algunos miembros de la línea de producción ofrecen estas características. Las condiciones de selección: [pago en efectivo], [pago con tarjeta de crédito], y [pago con tarjeta de débito] son mutuamente excluyentes. Estas condiciones opcionales alternativas se llaman [opción de tarjeta de crédito] y [opción de tarjeta de débito], respectivamente. En el lugar del caso de uso base donde se requiere el pago, la descripción de casos de uso es la siguiente:

Caso de Uso base: Pagar cliente
Nombre del caso de uso: Pagar cliente.
Reutilización de la categoría: Núcleo.
Resumen: El sistema registra la compra del cliente.
Actor: Cliente.
Precondición: la estación de salida es inactiva, mostrando mensaje "Bienvenido".
Descripción:

El Cliente escanea el elemento seleccionado.

1. El sistema muestra el nombre del elemento, el precio, y el total acumulado.
2. Se repite los pasos 1 y 2 por cada artículo comprado.
3. El cliente selecciona el pago.
4. El sistema muestra instrucciones para el pago en efectivo, tarjeta de débito o tarjeta de crédito.
5. <pago>
6. El sistema muestra la pantalla de agradecimiento.

En la descripción de este caso de uso base, el paso 6 <pago> es un marcador que identifica el lugar en que se ejecuta el caso de uso de extensión apropiado. Para el caso de uso de extensión *Pagar en efectivo*, la condición de extensión es una condición de selección llamada [pago en efectivo]. Este caso de uso de extensión se ejecuta si la condición [pago en efectivo] es verdadero.

Caso de uso de extensión Pagar en efectivo
Nombre del caso de uso: pagar en efectivo.
Reutilización de la categoría: Núcleo.
Resumen: El cliente paga en efectivo por los artículos comprados.
Actor: Cliente.
Dependencia: Extensión Pagar cliente
Pre condición: El cliente ha comprobado los elementos, pero aún no ha pagado por ellos.
Descripción:

1. El cliente selecciona el pago en efectivo.
2. El sistema le pide al cliente que deposite dinero en billetes y / o monedas.
3. El cajero ingresa el monto de dinero en efectivo.
4. El sistema calcula el cambio.
5. El sistema muestra el monto total a pagar, pago en efectivo, y el cambio.
6. El sistema imprime monto total a pagar, pago en efectivo, y el cambio en el recibo.

Para el caso de uso de extensión *Pagar con tarjeta de crédito*, la condición de selección se llama [pago con tarjeta de crédito]. También hay una condición de la línea de productos, que se llama [opción de tarjeta de crédito]. La condición de extensión es el AND lógico de la condición de selección y la condición de la línea de productos, es decir, [pago con tarjeta de crédito y opción de tarjeta de crédito] (ver Figura 2.1). Este caso de uso de extensión se ejecuta si la condición de extensión es verdadera, lo que significa que [opción de tarjeta de crédito] es proporcionada por este miembro de la línea de productos y el usuario decide pagar con tarjeta de crédito. Por supuesto, si el usuario prefirió pagar en efectivo, entonces el caso de uso *Pagar en efectivo* se ejecutará en su lugar.

Caso de uso de extensión Pagar con tarjeta de crédito:

Nombre del caso de uso: Pagar con tarjeta de crédito.
Reutilización de la categoría: Opcional.
Resumen: El cliente paga con tarjeta de crédito los artículos comprados.
Actor: Cliente.
Dependencia: Extiende el caso de uso Pagar Cliente.
Pre condición: El cliente ha seleccionado los elementos, pero aún no pagado por ellos.
Descripción:

1. El cliente selecciona el pago con tarjeta de crédito.
2. El sistema pide al cliente que pase la tarjeta magnética.
3. El cliente pasa la tarjeta.
4. El sistema lee ID de la tarjeta y la fecha de vencimiento.
5. El sistema envía la transacción al centro de autorización que contiene la ID de la tarjeta, fecha de vencimiento y el monto del pago.
6. Si la transacción es aprobada, el centro de autorización devuelve una confirmación positiva.
7. El sistema muestra el monto del pago y la confirmación.
8. El Sistema imprime el monto del pago y la confirmación en el recibo.

La descripción del caso de uso de extensión *Pagar con tarjeta de débito* es similar, excepto que el cliente además debe introducir un código PIN. *Pagar con tarjeta de débito* tiene una condición de selección llamada [pago con tarjeta de débito] y una condición de la línea de productos llamada [opción de tarjeta de débito].

2.1.1.6 Modelamiento de variabilidad de la línea de producción de software con puntos de extensión

Los puntos de extensión pueden ser usados para modelar la variabilidad de líneas de producción de software de las siguientes maneras:

Variabilidad Alternativa. Los casos de uso de extensión alternativos son mutuamente excluyentes. Cada caso de uso de extensión tiene una condición de la línea de producción en la relación de extensión, que debe ser verdadera para el caso de uso alternativo que debe facilitarse.

Variabilidad opcional. Un caso de uso opcional es proporcionado por el sistema de línea de producto sólo si la condición de la línea de producción es verdadera.

Evolución futura de la línea de producto. Un punto de extensión también puede permitir la evolución futura de la LPS, con este enfoque, el punto de extensión se utiliza como un marcador de posición para futuras ampliaciones a la línea de producción.

2.1.1.7 Modelamiento de la variabilidad de la línea de producción de software con la relación de inclusión en líneas de producción de software

La relación de inclusión (Include) de casos de uso puede ser utilizada para apoyar los casos de uso opcionales. Los casos de uso opcionales están protegidos por una condición de inclusión. Para un determinado miembro de la línea de productos, si la condición de la inclusión es verdadera, entonces el caso de uso opcional estará disponible para ese miembro.

Considere el ejemplo de la banca, que tiene un caso de uso núcleo abstracto *(Validar PIN)* y tres casos de uso núcleo concreto *(Retirar fondos, Consultar cuentas y Transferir fondos*). Otros casos de uso concretos opcionales, tales como el D*epositar fondos* e I*mprimir declaración,* incluidos también en el caso de uso núcleo *Validar PIN*. El diagrama de casos de uso se muestra en la Figura 2.2. Los casos de uso opcionales tienen que ser custodiados por las condiciones de inclusión. Por lo tanto estos casos de uso opcionales estarán disponibles sólo si la condición de inclusión es verdadera. Así, si la condición de inclusión [opción de depósito] es verdadera, el caso de uso Depositar fondos es proporcionado por ese miembro de la línea de productos.

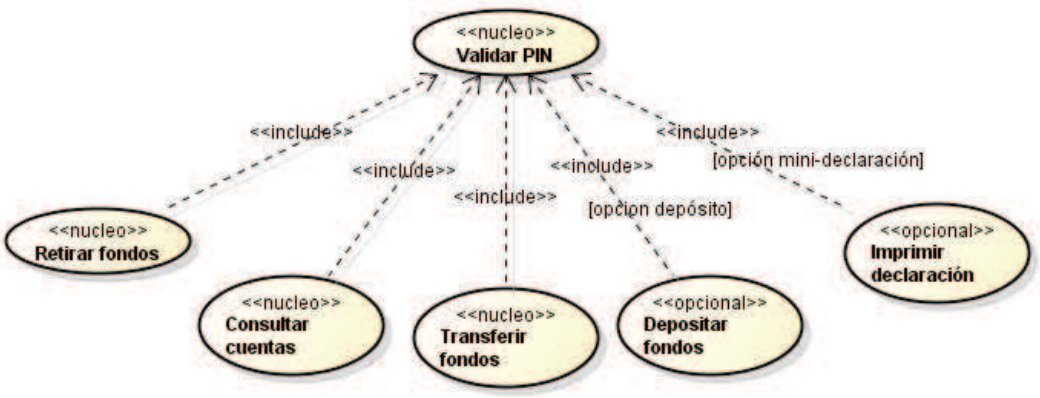


Figura 2.2 Ejemplo de casos de uso núcleo y opcionales con relaciones

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

Las partes principales de las descripciones de casos de uso están dadas por el caso de uso núcleo abstracto, un caso de uso núcleo concreto, y un caso de uso concreto opcional.

2.1.2 MODELAMIENTO DE CARACTERISTICAS

En particular, las características son los requerimientos funcionales que se usan para diferenciar a los miembros de la línea de producción y por lo tanto, para determinar y definir la funcionalidad común y variable de una LPS.

2.1.2.1 Modelamiento de la variabilidad de LPS con características

En el análisis y la categorización de características de una LPS, una importante distinción se hace entre las características comunes, características opcionales y características alternativas. Las **características comunes** son un subconjunto de las características de la línea de producción que deben ser proporcionadas por cada miembro de la línea de producción. Las **características opcionales** son aquellas que son proporcionadas sólo por algunos miembros de la línea de producción. Características alternativas son necesarias cuando hay una elección de características para seleccionar un determinado miembro de la línea de producción. Las características opcionales y alternativas describen la variabilidad en la línea de producción y determinan las características de un determinado miembro de la línea de producción.

2.1.2.2 Análisis de características comunes/variables

Durante el análisis de la variabilidad de características, las características de las líneas de producción, se analizan y se clasifican como: *opcional común* u *opcional alternativa*. Las características comunes determinan el grado de uniformidad en la línea de producción. Las características opcionales y alternativas sirven para determinar el grado de variabilidad en la línea de producción.

Las características se representan en UML con el estereotipo «tipo de característica» seguido por el nombre de la característica.

2.1.2.2.1 Características comunes

Las características comunes deben ser proporcionadas por cada miembro de la LPS. Las características comunes se las referencia a veces como elementos obligatorios, necesarios, o el *núcleo* de la línea de producción.
Estas características se escriben de la siguiente manera:

«Característica común» Nombre de la característica

2.1.2.2.2 Características opcionales

Las características opcionales son proporcionadas sólo por algunos miembros

de la LPS. Las características opcionales pueden asumir que las características comunes están dadas, por lo que pueden depender de la presencia de las características comunes. Estas se describen de la siguiente manera:

«Característica opcional» Nombre de la característica

2.1.2.2.3 Características alternativas

Dos o más características pueden ser alternativas a otra, donde sólo una de ellas se puede proporcionar en cualquier miembro dado de la LPS. Por lo tanto, las características son mutuamente excluyentes. La relación entre estas características alternativas se especifica como un grupo de características. Se las describe de la siguiente manera:

«Característica alternativa» Nombre de la característica

2.1.2.2.4 Características parametrizables

Una característica parametrizable es una característica que define un parámetro de la línea de productos cuyo valor debe ser definido en el tiempo de configuración del sistema.

Una característica parametrizable se representa de la siguiente manera:

«Característica parametrizable» Nombre de la característica {tipo = tipo de parámetro, valor permitido = rango de valores del parámetro o valores enumerados del parámetro, valor por defecto = valor del parámetro}

2.1.2.2.5 Características de pre-requisitos

Una característica puede depender de otra característica, la característica que es dependiente de otra se denomina **característica de pre-requisito**. Es posible que una característica tenga más de una característica de pre-requisito. Los dos casos se muestran aquí:

{Pre-requisito = Nombre de la característica de pre-requisito}

{Pre-requisito = Nombre de la primera característica de pre-requisito,..., Nombre de la enésima característica de Pre-requisito}

Si una característica opcional, característica alternativa o una característica parametrizable tiene un pre-requisito, entonces el valor de la etiqueta de la característica del **pre-requisito** es representado luego del nombre de la característica, como se indica a continuación:

«Característica opcional» Nombre de la característica {prerrequisito = Nombre de la característica de pre-requisito}

«Característica alternativa» Nombre de la característica {prerrequisito = Nombre de la característica de pre-requisito}

2.1.2.2.6 Características mutuamente incluyentes

Si dos características se requieren siempre juntas, entonces las características se consideran características mutuamente incluyentes.

Se las representa como un valor UML etiquetado con la palabra reservada **mutuamente incluyente**.

{Mutuamente Incluyente = Nombre de la característica Mutuamente Incluyente}

Una característica opcional con una característica mutuamente incluyente se expresa como sigue:

«Característica opcional» Nombre de la característica {Mutuamente Incluyente = Nombre de la característica Mutuamente Incluyente}

2.1.2.3 Características y casos de uso

Los casos de uso y características pueden ser utilizados como complementarios. En particular, los casos de uso se pueden asignar a las características sobre la base de sus propiedades de reutilización.

Las características se pueden determinar a partir de los casos de uso de varias maneras diferentes. Una característica puede corresponder a un solo caso de uso, un grupo de casos de uso, o a un punto de variación dentro de un caso de uso.

2.1.2.4 Modelamiento de características con UML

Hay varias maneras en que las características pueden ser modeladas en UML, como se describe a continuación. A menudo es útil usar más de un enfoque para proporcionar una mayor profundidad en el modelamiento de características.

2.1.2.4.1 Modelamiento de características como un paquete de casos de uso

Cuando a un grupo de casos de uso se los reutiliza en conjunto, se los puede asignar a una característica y mostrarlos como un paquete de casos de uso.

2.1.2.4.2 Modelamiento de características como metaclases

Con el modelamiento de metaclases, una clase se utiliza para representar un elemento modelado. En el modelamiento de características y grupos de características, estas se representan como metaclases, de modo que toda la potencia del modelamiento estático puede ser utilizada para representar características, grupos de características y relaciones.

La clasificación de los diferentes tipos de características de acuerdo a los estereotipos, se representa en la Figura 2.3 como un diagrama de metaclases.

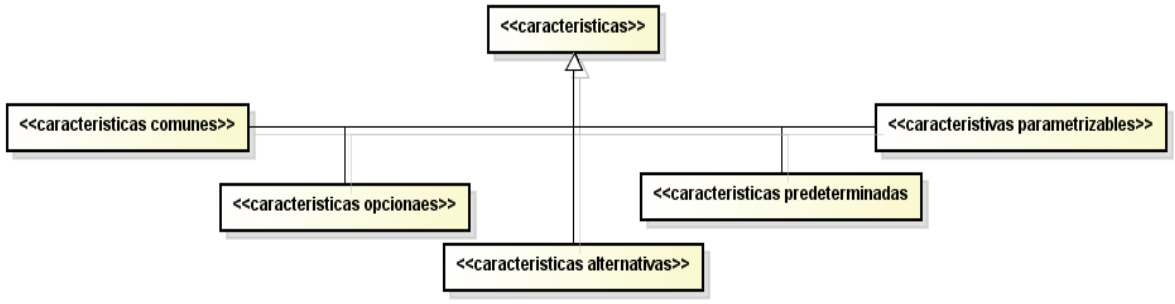


Figura 2.3 Clasificación de características de LPS usando estereotipos de UML

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

Las dependencias entre características se describen como asociaciones, tales como requeridas y mutuamente incluyentes; se muestran también las restricciones mutuamente excluyentes.

La Figura 2.4 muestra ejemplos de dependencias de característica de una línea de producción de horno microondas. *Receta* es una característica opcional que requiere la función de *Pantalla multilínea* y mutuamente incluye la función de *Peso analógico*. Los nombres de asociaciones diferentes usadas para modelar estas relaciones se utilizan para mostrar que *Pantalla multilínea* es una característica explícita que puede ser seleccionada de forma independiente por un miembro de la línea de productos, mientras que *Peso analógico* es una característica implícita de inclusión mutua que sólo se puede seleccionar en conjunto con la característica *Receta*.

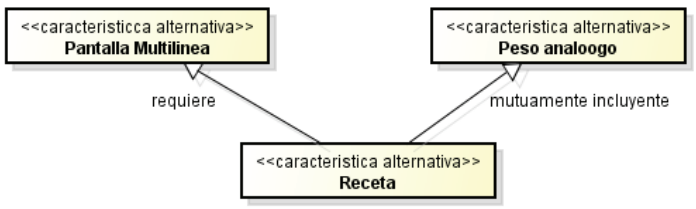


Figura 2.4 Ejemplo de dependencia de características en una LPS

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

2.1.2.4.3 Representación de características en tablas

Las características pueden describirse de forma concisa en una representación tabular. Cada fila de la tabla identifica una característica y su descripción se muestra en las columnas. Este enfoque se utiliza en la línea de productos del horno de microondas para describir de forma concisa la relación características/casos de uso, como se muestra en la Tabla 2.1

Nombre de la característica	Tipo/ categoría de la característica	Nombre del caso de uso	Categoría del caso de uso /Punto de variación (vp)	Nombre del punto de Variación
Núcleo del Horno Microondas	Común	Cocinar alimentos	Núcleo	
Luz	Opcional	Cocinar alimentos	vp	Iluminar
Plato giratorio	Opcional	Cocinar alimentos	vp	plato giratorio
Alerta	Opcional	Cocinar alimentos	vp	Alerta
Minuto adicional	Opcional	Cocinar alimentos	vp	Minuto adicional
Pantalla de una línea	Predeterminado	Cocinar alimentos	vp	Unidad de Pantalla
Pantalla de múltiple línea	Alternativa	Cocinar alimentos	vp	Unidad de Pantalla
Inglés	Predeterminado	Cocinar alimentos	vp	Idioma de pantalla
Francés	Alternativa	Cocinar alimentos	vp	Idioma de pantalla
Español	Alternativa	Cocinar alimentos	vp	Idioma de pantalla

Alemán	Alternativa	Cocinar alimentos	vp	Idioma de pantalla
Italiano	Alternativa	Cocinar alimentos	vp	Idioma de pantalla
Peso booleano	Predeterminado	Cocinar alimentos	vp	Sensor de peso
Peso analógico	Alternativa	Cocinar alimentos	vp	Sensor de peso
Calentamiento de un nivel	Predeterminado	Cocinar alimentos	vp	Elemento de calentamiento
Calentamiento de múltiple nivel	Alternativa	Cocinar alimentos	vp	Elemento de calentamiento
Nivel de potencia	Opcional	Cocinar alimentos	vp	Elemento de calentamiento
Reloj	Opcional	Setear hora del día	Opcional	
		Mostrar hora del día	Opcional	
Hora de reloj 12/24	Parametrizable	Setear hora del día	Vp	12/24 Hora de reloj
		Mostrar hora del día		
Receta	Opcional	Cocinar con receta	Opcional	

Tabla 2.1 Representación tabular de la relación características/casos de uso: LPS de horno de microondas

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

La característica común *Núcleo del Horno Microondas* consiste en el caso de uso núcleo *Cocinar alimentos*. Sin embargo, este caso de uso tiene varios

puntos de variación, en los que se inserta la funcionalidad especificada por las características opcionales, por defecto o alternativas. Por ejemplo, Luz es una característica opcional determinada a partir del caso de uso *Cocinar alimentos;* representa un punto de variación en el caso de uso *Iluminar*. Reloj es una característica opcional que agrupa a los dos casos de uso opcionales sobre la hora del día, y *Receta* es una característica opcional que corresponde a un caso de uso*: Cocinar con receta.*

2.1.2.5 Grupos de características

Las características relacionadas pueden agruparse en grupos de características, proporcionando así una restricción sobre cómo deben ser utilizadas por un determinado miembro de la línea de productos de software.

La notación de modelamiento estático utilizado originalmente para las características, se ilustra en la Figura 2.5.

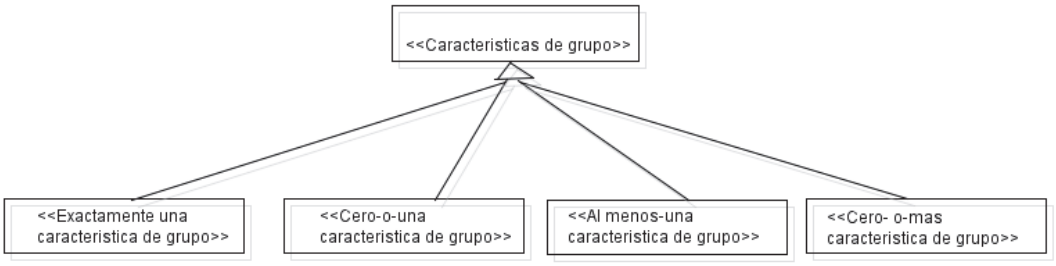


Figura 2.5 Clasificación de grupos de características de LPS utilizando estereotipos UML

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

El formato general es:
«tipo del grupo de características» Nombre del grupo de características {alternativa = Nombre de la primera característica alternativa,…, Nombre de la enésima característica alternativa, prerrequisito = Nombre de la característica de prerrequisito}

2.1.2.5.1 Características mutuamente excluyentes

Una característica puede ser una alternativa, es decir, una de un grupo de características mutuamente excluyentes. En algunas familias de producción, una de las alternativas siempre debe ser elegida para un miembro de la familia de producción dada. En otros casos, sin embargo, la selección de una de las alternativas es opcional para un miembro de la familia. Una *cero-o-una característica del grupo* maneja el caso en el que la selección de una de las alternativas es opcional. En las Figuras 2.6 y 2.7 se muestran algunos ejemplos de grupos de características para las líneas de producción de software.

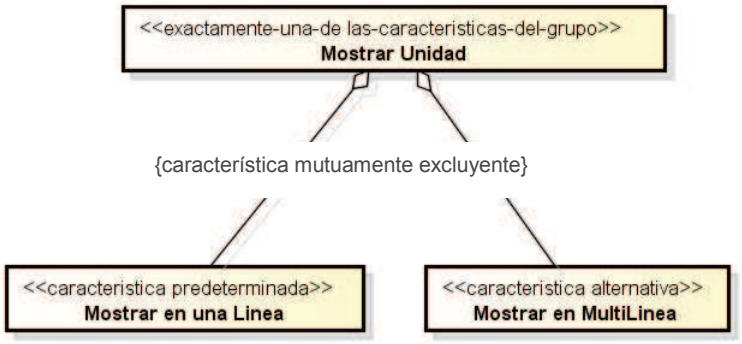


Figura 2.6 Línea de producción-Horno Microondas

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

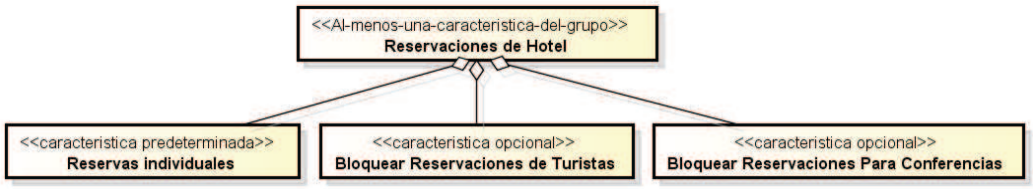


Figura 2.7 Línea de producción-Reservación de Hotel

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

2.1.2.5.2 Exactamente-una-característica del grupo

En la exactamente-una-característica del grupo, una característica siempre debe estar seleccionada entre el grupo. A este grupo de características se le da el nombre y el estereotipo «exactamente-una característica del grupo».

2.1.2.5.3 Al menos-una-característica del grupo

En al menos-una-característica del grupo, una o más características opcionales se pueden seleccionar entre el grupo, pero al menos una característica debe ser seleccionada. También es posible para tal grupo de características tener una característica predeterminada y un prerrequisito.

2.1.2.5.4 Cero-o-más-característica del grupo

A primera vista un grupo de características cero-o-más-característica del grupo podría parecer innecesaria, ya que *cero o más* significa que todas las características del grupo son opcionales y no hay restricciones en la selección de características dentro del grupo. Sin embargo, a veces es útil agrupar características opcionales debido a una dependencia.

2.1.2.5.5 Representación de características del grupo mediante tablas

La información del grupo de características también se puede resumir en una tabla. Una representación tabular de los dos grupos de características que aparecen en las Figuras 2.6 y 2.7 se presentan en la Tabla 2.2.

Nombre Del Grupo De Características	Categoría del grupo de características	Características en grupo de características	Categoría de característica
Mostrar Unidad	Exactamente-una-de	Mostrar Una-línea	Predeterminada
		Mostrar Multi-linea	Alternativa
Reservación de Hotel	Al-menos-una-de	Reservaciones Individuales	Predeterminada

		Bloquear reservaciones de Turistas	Opcional
		Bloquear Reservaciones para Conferencia	Opcional

Tabla 2.2 Ejemplo de una representación tabular de grupos de características

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

2.1.3 MODELAMIENTO ESTÁTICO EN LINEAS DE PRODUCCIÓN DE SOFTWARE

El modelo estático es una anotación de gran alcance para la captura de lo común y de lo variable en una familia de producción. Un modelo estático describe la estructura estática de la línea de producción que se esté modelando.

El enfoque comienza por modelar las clases del mundo real, que se determinan a partir del dominio del problema, antes de modelar las clases de software dentro de la LPS. Y describe la clasificación de las clases de LPS, desde dos perspectivas: el papel que la clase desempeña en la aplicación, y la característica reutilizable de la clase.

2.1.3.1 Modelamiento de lo común y la variabilidad en la LPS

Un tema crucial en el modelamiento de las LPS es la forma de analizar y modelar el carácter común y la variabilidad en la familia de producción. En un modelo estático de un sistema único, todas las clases son obligatorias. En LPS, sin embargo, una clase es requerida por al menos un miembro de la línea de producción. Una clase que es requerida por todos los miembros de la línea de producción se conoce como una clase núcleo. A veces, los diferentes

miembros de la familia de productos requieren versiones alternativas de una clase. Estas clases se denominan variantes de clases. Entre un grupo de variantes de clases, una clase puede ser designada como la clase predeterminada.

2.1.3.1.1 Categorización y estereotipos UML

En el modelado de líneas de productos de software, cada clase se puede clasificar de acuerdo a su característica de reutilización. Los estereotipos «núcleo», «opcional» y «variante» se utilizan para distinguir entre las características de reutilización de las clases de líneas de productos.

2.1.3.1.2 Modelamiento estático del dominio del problema para líneas de producción de software

En el modelo estático del dominio del problema, el énfasis inicial es en el modelado de clases Físicas y clases de Entidad. *Las **clases Físicas*** son las clases que tienen características físicas, es decir, que pueden ser vistas y tocadas. *Las **clases Entidad*** son las clases conceptuales de datos intensivos que suelen ser persistentes, es decir, perduran en el tiempo.

En una LPS es posible que algunas de las clases del mundo real sean núcleo, mientras que otras sean opcionales y otras sean alternativas. La Figura 6.3 muestra el modelo conceptual estático del dominio del problema para una línea de producción de horno microondas. Muestra las clases del núcleo, tales como reloj y sensor de puerta, y muestra tres clases opcionales: Alarma, PlatoGiratorio, y Luz. Las clases del núcleo, como el Teclado y SensordePuerta, tienen una asociación uno-a-uno con la clase compuesta HornoMicroondas. Las clases opcionales, tales como Alarma y Luz, sin embargo, tienen una asociación opcional (cero-o-uno) con la clase compuesta HornoMicroondas.

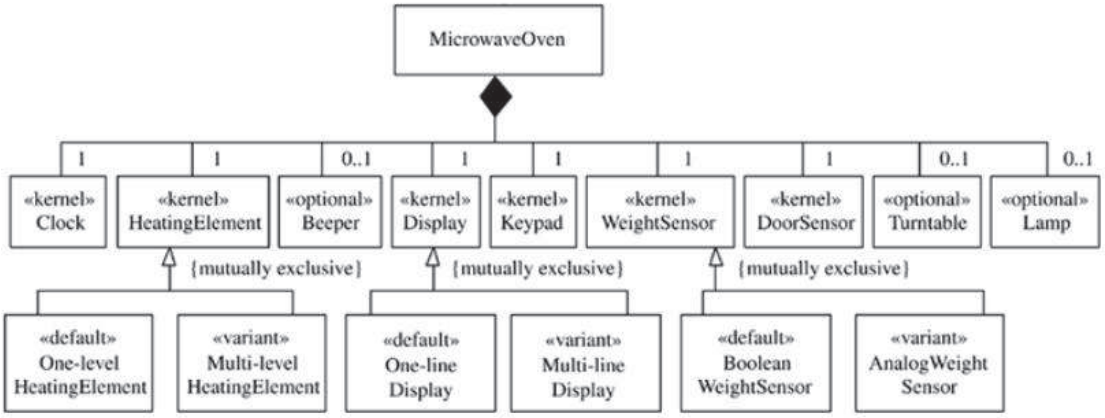


Figura 2.8 Modelo estático conceptual de la línea de producción de horno microondas

Fuente: (Gomma, 2007)

El modelo estático del dominio del problema también representa a las clases de variantes, tales como Sensor de Peso de booleano y Sensor de Peso analógico. Estas clases en el mundo real se modelan como una jerarquía de generalización / especialización. Las variantes se representan como subclases de una clase, de esta forma Sensor de Peso es una clase núcleo porque captura las propiedades comunes de un sensor de peso. Si una de las clases de la variante es la clase predeterminada, como Sensor de Peso Booleano, se le da el estereotipo «predeterminada>>. La otra clase o clases variantes tienen el estereotipo de «variante», como Sensor de Peso Analógico. Una restricción mutuamente excluyente que se representa entre llaves, también se representa para indicar que las subclases nunca pueden coexistir en el mismo horno microondas.

2.1.3.2 Modelamiento estático del alcance de aplicación de LPS

Es muy importante entender el alcance de una LPS. El límite entre la línea de producción de software y el ambiente externo se conoce como el contexto de líneas de producción de software. La notación UML no apoya explícitamente un diagrama de contexto del sistema, sino que el contexto del sistema es representado por un modelo estático [9] [10]. En el contexto de la línea de

producción, un diagrama de clases tiene que capturar la variabilidad en la frontera.

2.1.3.2.1 Clases externas de la línea de producción

El modelo de contexto de la línea de producción utiliza estereotipos UML para representar el límite de la línea de producción con un diagrama de clases UML. El contexto de la línea de producción se representa mostrando un sistema de línea de producción (miembro de la línea de producción) como una clase agregada con el estereotipo de «línea de producción del sistema», y el ambiente externo se representa como clases externas para las cuales el sistema debe tener interfaz.

Las clases externas se clasifican por el estereotipo <<externa>>.

Dispositivo de entrada externa. Es un dispositivo que sólo proporciona la entrada al sistema, por ejemplo, un sensor; es representado con el estereotipo <<externa input device>>

Dispositivo de salida externa. Es un dispositivo que sólo recibe la salida del sistema, por ejemplo, un actuador; el estereotipo es <<externa output device>>

Entrada externa / dispositivo de salida. Es un dispositivo que proporciona la entrada al sistema y recibe la salida del sistema, por ejemplo, un lector de tarjeta de cajero automático. Se representa con el estereotipo <<externa input/output device>>

2.1.3.3 Modelos estáticos de clases entidad

Las clases entidad conceptuales son clases intensivas de datos que almacenan los datos persistentes (es decir, de larga vida). El modelamiento de las clases entidad es similar a modelar un esquema de base de datos lógica [11]. Las clases entidad a menudo se asignan a una base de datos en la fase de diseño. A pesar de que ambos enfoques, el modelo estático de UML y el modelo entidad-relación (que se utiliza con frecuencia para el diseño de base de datos lógica), permiten modelar las clases, los atributos de cada clase y las

relaciones entre las clases, UML también permite operaciones de la clase a ser especificada.

Un ejemplo de modelo de clase entidad viene de la línea de producción de comercio electrónico, en el que las cuentas bancarias, clientes, facturas, y los catálogos son mencionados en la descripción del problema. Cada una de estas entidades conceptuales del mundo real se modela como una clase entidad y se representa con el estereotipo de «entidad». Los atributos de cada clase entidad se determinan, y las relaciones entre las clases entidad se definen.

2.1.3.3.1 Desarrollo de modelos de clases entidad en líneas de producción de software

La forma en que se desarrolla el modelo de clase entidad para la línea de producción depende de la estrategia utilizada. El modelo de clase entidad para la línea de producción se produce por la integración de los modelos de clase entidad individuales. El modelo de clase entidad para el núcleo del sistema se desarrolla primero y luego evoluciona a medida que las variaciones se consideran para su inclusión en el modelo de clase entidad.

Cuando los modelos de la clase entidad para los diferentes miembros de la línea de producción se integran, las clases que son comunes a todos los miembros de la línea pasan a ser las clases núcleo del modelo clase entidad de la línea de producción integrada. Las clases que se encuentran en un punto de vista, pero no otros se convierten en clases opcionales en el modelo clase entidad de línea de producción. Las clases que tienen algunos atributos y operaciones comunes, pero también algunas diferencias, son generalizadas. Los atributos y operaciones comunes son capturados en la superclase, las diferencias se reflejan en las subclases de la variante de la superclase.

2.1.4 MODELAMIENTO DE INTERACCIÓN DINÁMICA PARA LÍNEAS DE PRODUCTOS SOFTWARE

El modelamiento dinámico ofrece una vista de una línea de productos en la que se considera el control y la secuencia, ya sea dentro de un objeto (por medio de una máquina de estado finito) o entre objetos (por análisis de interacciones de objeto).

El modelamiento dinámico se basa en los casos de uso desarrollados durante el modelado de casos de uso. Para cada caso de uso, los objetos que participan en el caso de uso están determinados, y las formas en que los objetos interactúan se muestran con el fin de satisfacer los requisitos descritos en el caso de uso. Para cada caso de uso, un diagrama de interacción está desarrollado para mostrar los objetos que participan en el caso de uso y la secuencia de mensajes que se pasan entre ellos. La interacción se representa bien en un diagrama de comunicaciones (que se llama diagrama de colaboración en UML 1.x) o un diagrama de secuencia. Una descripción narrativa de la interacción de objetos se proporciona en una descripción de la secuencia de mensajes.

2.1.4.1 Modelo dinámico evolutivo en líneas de producción de software

En los sistemas individuales, cada caso de uso es necesario. En las LPS, sin embargo, los casos de uso se clasifican como núcleo, opcional o alternativo. Un miembro determinado de la línea de productos necesita de todos los casos de uso del Núcleo, pero sólo algunos de los casos de uso opcional y / o alternativas. Para las LPS, es necesario realizar el modelo dinámico en todos los casos de uso cualquiera que sea su categoría de reutilización con el fin de determinar qué objetos son necesarios para cada caso de uso y cómo interactúan entre sí.

Los diagramas de comunicación se clasifican en núcleo, opcional o alternativo, correspondiente a la categoría de reutilización de los casos de uso para el que se desarrollan. Las variantes de diagramas de comunicación pueden ser desarrolladas para mostrar el impacto del punto de variación en los casos de uso.

Además, es posible desarrollar características basadas en diagramas de comunicación, es decir, un diagrama de comunicación para cada característica funcional. Este enfoque es útil cuando los casos de uso se vuelven a utilizar juntos y por lo tanto se combinan en una característica, por la que cada diagrama de comunicación basada en una característica es desarrollado.

2.1.4.1.1 Análisis evolutivo dinámico para líneas de productos software

El análisis dinámico evolutivo es una estrategia iterativa para ayudar a determinar cómo los objetos del modelo de análisis deben interactuar unos con otros para apoyar los casos de uso. El análisis dinámico se lleva a cabo para cada caso de uso. Un primer intento se hace para determinar los objetos que participan en un caso de uso, utilizando los criterios de estructuración de objetos. Luego, se analiza la forma en que estos objetos colaboran para ejecutar el caso de uso. Este análisis podría mostrar una necesidad de que se definan objetos adicionales y / o interacciones adicionales [6].

Dado que los casos de uso se han clasificado como núcleo, opcional o alternativa, la estrategia se inicia con los casos de uso del núcleo seguido por los casos de uso opcional y alternativo. Además, el orden de desarrollo de los diagramas de comunicación de objetos, sigue la jerarquía de características dependientes. Esto significa que cuando un diagrama de comunicación se desarrolla, se puede suponer la existencia de objetos que apoyan las características de pre-requisitos y casos de uso.

El análisis dinámico puede ser dependiente del estado o no dependiente de estado, dependiendo de si el objeto de comunicación es dependiente del estado.

2.2 PROCESO PARA LA APLICACIÓN DE MDA A LÍNEAS DE PRODUCCIÓN DE SOFTWARE

El proceso para la aplicación de MDA a LPS en este trabajo se encuentra establecida bajo los lineamientos estudiados en las secciones anteriores sobre MDA y LPS y se describe en la Tabla 2.3, mostrada a continuación.

ETAPAS DE DESARROLLO	Tareas en LPS	Modelos MDA
Análisis de Requerimientos	Análisis de requerimientos Descripción del Problema Selección de requerimientos principales y secundarios. Refinamiento.	**CIM**
Diseño	Modelamiento de casos de uso Diagrama de CU Documentación Variabilidad	**PIM**
	Modelamiento de características Análisis de características. Representación de características. Modelamiento en grupo de características. Diagrama de grupo de características	
	Modelamiento estático. Diagrama estático conceptual.	

	Diagrama de clases entidad.	
	Modelamiento Dinámico. Diagrama de comunicación.	
Código	Generación de código	**PSM**
Pruebas	Plan de Pruebas	
Producto	LPS desarrollada.	

Tabla 2.3 Proceso de desarrollo con LPS y modelos MDA

Elaborado por: (Inga F & Sarabia P, 2013)

Para lograr cumplir con la aplicación de MDA a LPS, se requiere la construcción de modelos, lo que permite introducir una serie de herramientas que complementan el proceso de desarrollo con el fin de dar soporte al mismo.

Lo que se busca es incluir varios modelos relacionados entre sí, organizados a través de varios niveles de abstracción, con mapeos definidos de un conjunto de modelos hacia otro y que pueden incluir lo descrito en MDA.

2.2.1 DESCRIPCIÓN DEL PROCESO DE DESARROLLO PARA LA APLICACIÓN DE MDA A LÍNEAS DE PRODUCCIÓN DE SOFTWARE

a) Análisis de requerimientos

a.1) Análisis de requerimientos: Esta etapa permite la obtención de especificaciones a partir del cliente.

a.1.1) Descripción del problema: Permite conocer la temática y el problema a resolver.

a.1.2) Selección de requerimientos funcionales y no funcionales: La especificación de requerimientos, es una parte crucial en el proceso de desarrollo de software, ya que de esta etapa depende el logro de los objetivos finales previstos; por lo que se debe determinar los requerimientos principales como punto de partida.

a.1.3) Refinamiento: Dada la realimentación que existe con el cliente, permite refinar la información y corregirla si es necesario.

El modelo MDA a implementarse es un modelo CIM el mismo que contenga todas las reglas del negocio.

b) Diseño

b.1) Modelamiento de casos de uso: En esta etapa se extiende el enfoque del modelo de casos de uso a LPS.

b.1.1) *Diagrama de casos de uso:* Se selecciona y modela los requisitos funcionales.

b.1.2) Documentación: Se describe cada caso de uso considerado en el modelo de casos de uso.

b.1.3) *Variabilidad:* Se establecen los puntos de variación de los casos de uso.

b.2) Modelamiento de Características:

b.2.1) *Análisis de Características*: Se analizan y se clasifican las características.

b.2.2) *Representación de características*: Se modelan las características usando los estereotipos definidos por su clasificación.

b.2.3) Modelamiento en grupo de características: Se seleccionan y agrupan las características que se relacionan entre sí.

b.2.4) Diagrama de grupo de características: Se modela el grupo de características asignando a cada característica su estereotipo.

b.3) Modelamiento Estático:

b.3.1) Diagrama estático conceptual: Se modela el carácter común y variable en la familia de producción, utilizando los estereotipos designados para cada tipo de clase.

b.3.2) Modelamiento de clases entidad: Se modelan las clases entidad que se asignan a una base de datos.

b.4) Modelamiento Dinámico:

b.4.1) Diagrama de comunicación: En esta etapa se clasifican a los diagramas según la categoría de reutilización de los casos de uso.

En la etapa de diseño se aplica el modelo PIM de MDA, ya que se posee la información necesaria para el modelo conceptual completo con todas sus relaciones.

c) Código:

c.1) Generación de código: Se genera el código en base al modelo de clases.

En esta etapa se utilizan herramientas que permitan transformar un modelo PIM a un modelo PSM.

d) Pruebas:

d.1) Pruebas: Se deben realizar las pruebas con lo que se recomienda el planteamiento de un plan de pruebas de validación para comprobar la funcionalidad de la aplicación y certificar que cumple con los requerimientos indicados en la etapa de análisis.

3 PROTOTIPO DE APLICACIÓN

3.1 DEFINICION DEL PROTOTIPO

Para el desarrollo del prototipo utilizaremos los principales componentes de comercio electrónico conocidos como Negocio-a-Negocio (Business-to-Business B2B) y Negocio-a-cliente (Business-to-Customer B2C).

Cabe mencionar que como se trata de un prototipo no se llevará a cabo la totalidad de etapas de desarrollo de software, más bien, se ha seleccionado etapas de desarrollo que permitan identificar el proceso propuesto para la aplicación de MDA a Líneas de Producción de Software planteado en el Capítulo 2 , que es objeto de este trabajo.

3.1.1 COMPONENTE BUSINESS-TO-BUSINESS B2B

En los sistemas de comercio electrónico B2B, hay clientes y proveedores. Cada cliente tiene un contrato con un proveedor para realizar una compra, así como una o más cuentas bancarias a través del cual se pueden realizar los pagos a los proveedores. Cada proveedor ofrece un catálogo de productos, acepta pedidos de clientes, y mantiene cuentas con cada cliente para recibir el pago.

Un cliente es capaz de navegar a través de varios catálogos disponibles en la Web proporcionados por los proveedores y seleccionar artículos para comprar. El pedido del cliente se debe cotejarse con los contratos disponibles para determinar si existe un contrato de cliente válido con el proveedor, que luego se utilizara para cargar la compra. Cada contrato tiene fondos de funcionamiento comprometidos. Es necesario determinar si hay suficientes fondos disponibles para la orden del cliente. Suponiendo que el contrato y los fondos están en su lugar, se crea una orden de entrega y se envía al proveedor del catálogo. El proveedor confirma la orden y entra en una fecha de entrega prevista. A medida que pasa el tiempo, la orden de envío se controla, y el proveedor y el cliente son notificado si hay un retraso de envío. Cuando se envía la orden, se notifica al cliente. El cliente reconoce cuando se recibe el envío y la orden de

entrega se actualiza. Después de la recepción del envío, el pago de la factura es autorizada. La factura se verifica por primera vez contra el contrato, los fondos disponibles y el estado de la orden de entrega, y luego es enviado a cuentas por pagar, que autoriza el pago de los fondos. El pago se realiza a través de transferencia electrónica de fondos desde el banco del cliente al banco proveedor.

Opcionalmente, un proveedor puede crear una orden de compra (OC) solicitar nuevos suministros de inventario de un mayorista. La OC se envía directamente al mayorista. Cuando el pedido es entregado al proveedor, los nuevos materiales se introducen en el inventario del proveedor, y el pago se realiza mediante transferencia electrónica de fondos del banco proveedor al banco mayorista.

3.1.2 COMPONENTE BUSINESS-TO-CONSUMER B2C

En los sistemas de comercio electrónico B2C, un cliente solicita la compra de uno o más elementos de esta última. El cliente proporciona información personal, como la dirección y la información de su tarjeta de crédito. Esta información se almacena en una cuenta de cliente. Si la tarjeta de crédito es válida, entonces se crea una orden de entrega y es enviada al proveedor. El proveedor confirma la orden y entra en una fecha de entrega prevista. Cuando se envía el pedido, el cliente es notificado y se carga a la cuenta de la tarjeta de crédito del cliente.

3.2 DESARROLLO DEL PROTOTIPO

3.2.1 ANÁLISIS DE REQUERIMIENTOS

Por tratarse de un caso de estudio basado en los requerimientos generales establecidos para sistemas de comercio electrónico B2B y B2C no se realiza una descripción del problema ni análisis de requerimientos, ya que las especificaciones necesarias básicas para la creación del prototipo se detallan de forma general en la sección 3.1.1 y 3.1.2.

3.2.1.1 Requerimientos funcionales

Los requerimientos funcionales para los componentes B2B y B2C se indican en las Tablas 3.1 y 3.2 respectivamente, que se muestran a continuación:

Componente B2B
1. Búsqueda de artículos en el catálogo de proveedor. **2.** El cliente puede realizar una solicitud de pedido. **3.** El cliente puede solicitar al sistema enviar una solicitud de compra al proveedor. **4.** El proveedor debe poder solicitar una orden de entrega, determinar que el inventario está disponible para cumplir la orden, y mostrar la orden. **5.** El proveedor debe poder confirmar el envío realizado. **6.** El cliente debe poder confirmar la llegada de su entrega. **7.** El proveedor debe enviar la factura, a la organización del cliente.

Tabla 3.1 Requerimientos funcionales Componente B2B

Elaborado por: (Inga F & Sarabia P, 2013)

Componente B2C
1. Búsqueda de artículos en el catálogo de proveedor. **2.** Debe poder validarse los datos de la tarjeta de crédito del cliente. **3.** El proveedor debe poder solicitar una orden de entrega, determinar que el inventario está disponible para cumplir la orden, y mostrar la orden. **4.** El proveedor debe poder confirmar el envío realizado. **5.** El cliente debe poder confirmar la llegada de su entrega. **6.** El sistema debe recuperar la información de la tarjeta de crédito del cliente y enviar una solicitud de pago al centro de autorización de la tarjeta de crédito.

Tabla 3.2 Requerimientos funcionales Componente B2C

3.2.1.2 Requerimientos no funcionales

Los requerimientos no funcionales para los componentes B2B y B2C se indican en la Tabla 3.3, que se muestran a continuación:

Componente B2B y B2C
Usuario: 1. Protección de usuario 2. Flexibilidad para adicionar requerimientos 3. Integridad 4. Confiabilidad 5. Usabilidad Desarrollo: 1. Mantenibilidad Documentos de Diseño Estándares de documentación 2. Facilitar pruebas 3. Portabilidad Seguridad: 1. Protección de la información

Tabla 3.3 Requerimientos no funcionales Componentes B2B y B2C

Elaborado por: (Inga F & Sarabia P, 2013)

3.2.1.3 Refinamiento

Es nuestro caso por tratarse de un prototipo nos basaremos en los requerimientos funcionales indicados anteriormente.

3.2.2 DISEÑO

3.2.2.1 Modelamiento de Casos de Uso

Debido a que hay dos principales sistemas B2B y B2C en la línea de productos de comercio electrónico, se aplicara primero un enfoque de ingeniería inversa para cada tipo de sistema, de la cual se determina la similitud y la variabilidad de la línea de productos.

MODELOS DE CASOS DE USO DE COMERCIO ELECTRÓNICO BUSINESSTO-BUSINESS

El modelo de casos de uso para los sistemas B2B en la línea de productos de comercio electrónico se muestra en la Figura 3.1. Hay tres actores: cliente, proveedor, y Banco. El cliente inicia cuatro casos de uso: **Ver catálogo**, **Crear**

solicitud, **Realizar solicitud de compra** y **Confirma entrega**. El proveedor inicia tres casos de uso: **Procesar orden de entrega**, **Confirmar envío**, y **Enviar factura**. Estos son los principales casos de uso del B2B en la línea de productos.

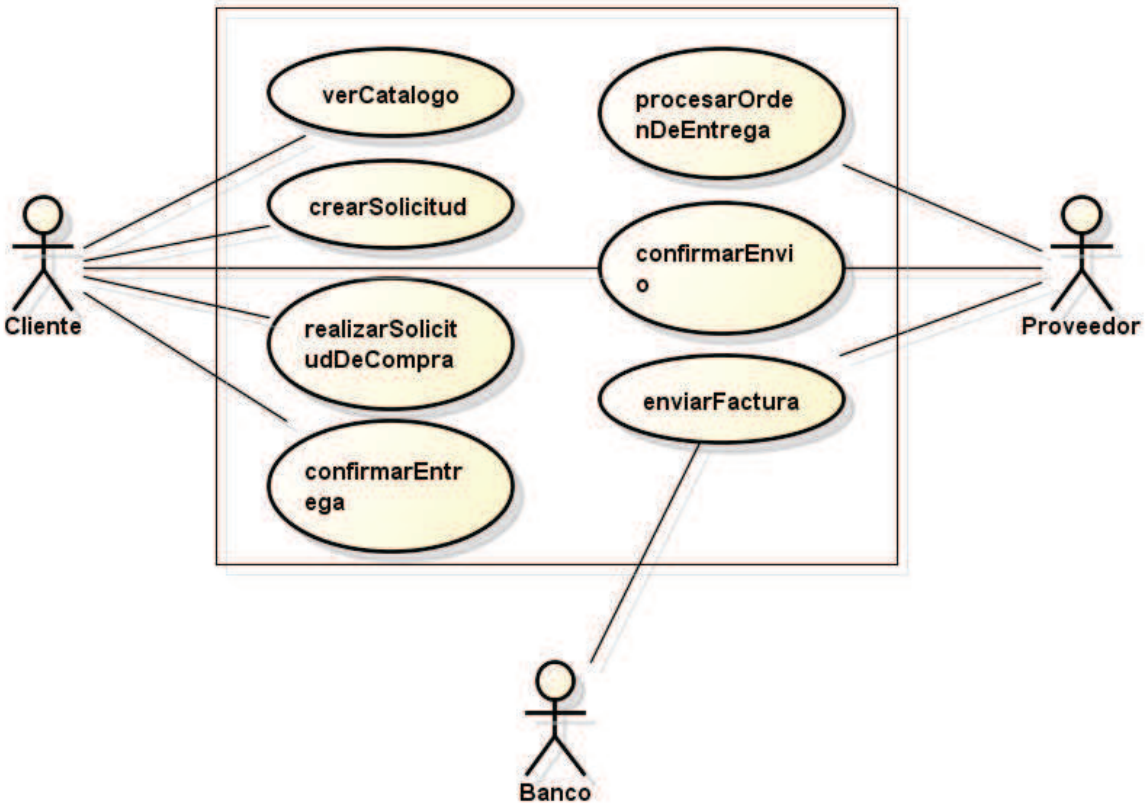


Figura 3.1 Sistema de Comercio Electrónico B2B: Casos de Uso

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

Descripción de los casos de uso del sistema de comercio electrónico B2B.

En el caso de uso **verCatálogo**, el cliente navega por los diferentes catálogos de la internet, considera varios artículos del catálogo proporcionado por el proveedor, y los selecciona.

En el caso de uso **crearSolicitud**, el cliente realiza una solicitud de pedido. El sistema tiene que encontrar el contrato del cliente en el catálogo del proveedor y determinar que existan suficientes fondos de operación. Si un contrato válido se encuentra, y existen suficientes fondos de operación para la solicitud realizada, el sistema autoriza la solicitud e informa al cliente.

En el caso de uso **realizarSolicitudDeCompra**, el cliente solicita al sistema enviar una solicitud de compra al proveedor. Aunque un análisis de caso de uso inicial podría sugerir que **crearSolicitud** y **realizarSolicitudDeCompra** deben ser combinados en un solo caso de uso (por ejemplo, ubicarSolicitud), la razón de la división de los dos casos de uso es que crearSolicitud sólo se aplica a los sistemas B2B, mientras **realizarSolicitudDeCompra** se aplica a todos los sistemas de comercio electrónico.

En el caso de uso **procesarOrdenDeEntrega**, el proveedor solicita una orden de entrega, determina que el inventario está disponible para cumplir la orden, y muestra la orden.

En el caso de uso **confirmarEnvío**, el proveedor prepara el envío de forma manual y luego confirma el envío.

En el caso de uso **confirmarEntrega**, cuando el envío llega al cliente, el cliente confirma la entrega. Los fondos de operación son comprometidos para el pago.

En el caso de uso de **envíoDeFactura**, el proveedor envía una factura a la organización del cliente. Después de que el cliente confirma la entrega, la factura está aprobada por el departamento de Cuentas por Pagar de la organización del cliente, y una solicitud de pago electrónico es enviada al banco del cliente.

MODELO DE CASOS DE USO PARA COMERCIO ELECTRÓNICO BUSINESS-TO-CONSUMER

El modelo de casos de uso para los sistemas de B2C en la línea de producción de comercio electrónico se muestra en la Figura 3.2. Al igual que en los sistemas B2B, hay tres actores, esta vez, sin embargo, son: Cliente, Proveedor, y Centro de Autorización. El cliente inicia tres casos de uso: verCatálogo, verificarCuentaDeCliente, y realizarSolicitudDeCompra. El proveedor inicia tres casos de uso: procesarOrdenDeEntrega, confirmarEnvío, y facturarCliente.

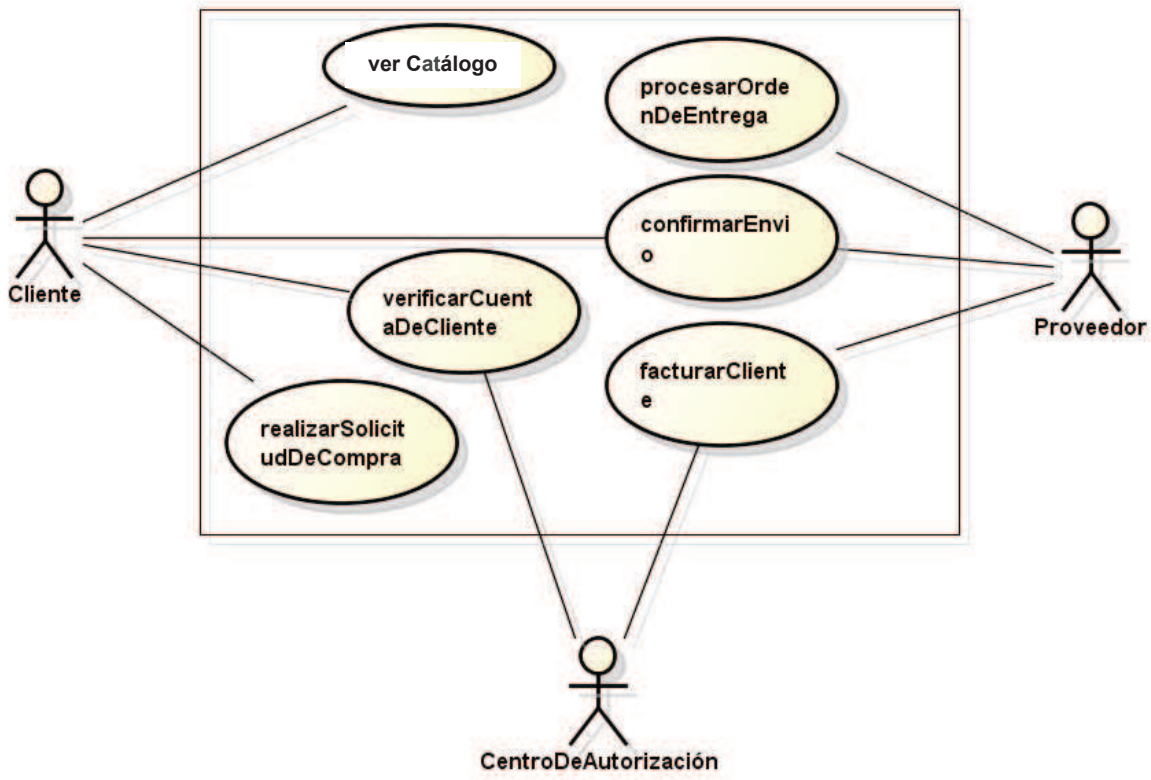


Figura 3.2 Sistema de Comercio Electrónico B2C: Casos de Uso

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

Descripción de los casos de uso del sistema de comercio electrónico B2C.

Dos de los casos de uso iniciados por el cliente en el sistema B2C **verCatálogo** y **realizarSolicitudDeCompra** son idénticos a los casos de uso B2B; **verificarCuentaDeCliente**, sin embargo, es diferente a cualquiera de los casos de uso en el B2B. En el caso de uso **verificarCuentaDeCliente**, el cliente introduce los datos personales. El sistema crea una cuenta de cliente si no existe todavía. En la tarjeta de crédito del cliente se comprueba su validez y el crédito suficiente para pagar los artículos del catálogo solicitado. Si la verificación de tarjeta de crédito muestra que la tarjeta de crédito es válida y tiene el crédito suficiente, entonces la compra del cliente ha sido aprobada.

Dos de los casos de uso iniciados por el proveedor en el sistema B2C: **procesarOrdenDeEntrega** y **confirmarEnvío** son idénticos a los casos de uso B2B; por otro lado, **facturarCliente**, es diferente. En el caso de uso **facturarCliente**, el sistema recupera información de la tarjeta de crédito del

cliente desde la cuenta del cliente y envía una solicitud de pago al centro de autorización de la tarjeta de crédito.

Casos de uso opcionales orden de compra

El proveedor inicia dos casos de uso opcionales que pueden ser utilizados tanto en los sistemas de B2B o B2C: **prepararOrdenDeCompra** y **entregarOrdenDeCompra** (ver Figura 3.3). En el caso de uso **prepararOrdenDeCompra**, el proveedor controla el inventario y solicita la creación de una orden de compra para los artículos del inventario que deben ser repuestos. En el caso de uso **entregarOrdenDeCompra**, el sistema envía la orden de compra al mayorista. Cuando llega la orden de entrega al mayorista, el proveedor introduce la información en la base de datos de inventario y aprueba el pago electrónico del banco proveedor para el banco mayorista.

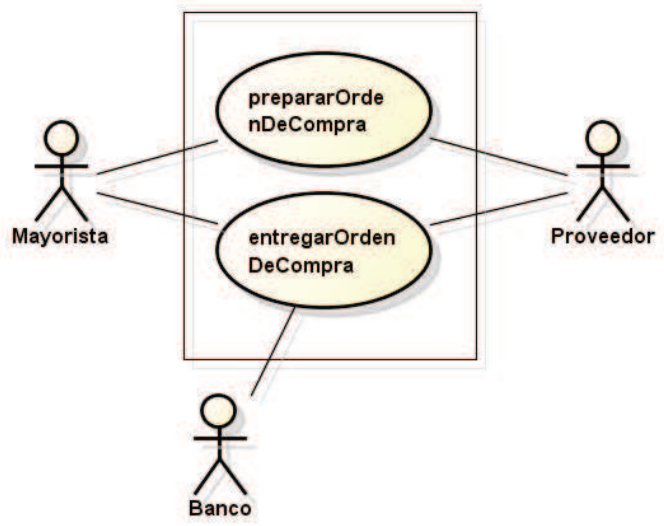


Figura 3.3 Orden de Compra Casos de Uso Opcional

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

MODELO DE CASOS DE USO PARA LA LÍNEA DE PRODUCCIÓN SOFTWARE DE COMERCIO ELECTRÓNICO

En el enfoque de ingeniería inversa, varios modelos de casos de uso (ver Figuras 3.1-3.3) se integran para producir el modelo de casos uso de la línea

de producción. Dos de los casos de uso iniciados en el cliente (VerCatálogo y realizarSolicitudDeCompra) son comunes a todos los sistemas de comercio electrónico, por lo tanto, llegarán a convertirse así en los casos de uso núcleo de la línea de producción.

Del mismo modo, dos de los casos de uso iniciados por proveedores (ProcesarOrdenDeEntrega y confirmarEnvío) son comunes a todos los sistemas de comercio electrónico, por lo que se convierten en casos de uso núcleo.

Por otro lado, dos de los casos de uso de cliente como son (crearSolicitud y confirmarEntrega) sólo se utilizan en los sistemas B2B, y un tercer de casos de uso (verificarCuentaDeCliente) sólo se utiliza en los sistemas B2C. En el lado del proveedor, un caso de uso (enviarFactura) sólo se utiliza en sistemas B2B, y otro caso de uso (facturarCliente) sólo se utiliza en los sistemas B2C. Por último, los dos casos de uso de orden compra (prepararOrdenDeCompra y entregarOrdenCompra) son opcionales y pueden ser utilizados en cualquiera de los sistemas de B2B o B2C.

Los casos de uso que pueden ser usados solo en sistemas B2B o B2C, pero no en ambos, son categorizados como casos de uso alternativos. Los casos de uso opcionales ordenDeCompra, no se sujetan a esta restricción, y son categorizados como casos de uso opcionales. El modelo de casos de uso para líneas de producción se muestra en la Figura 3.4.

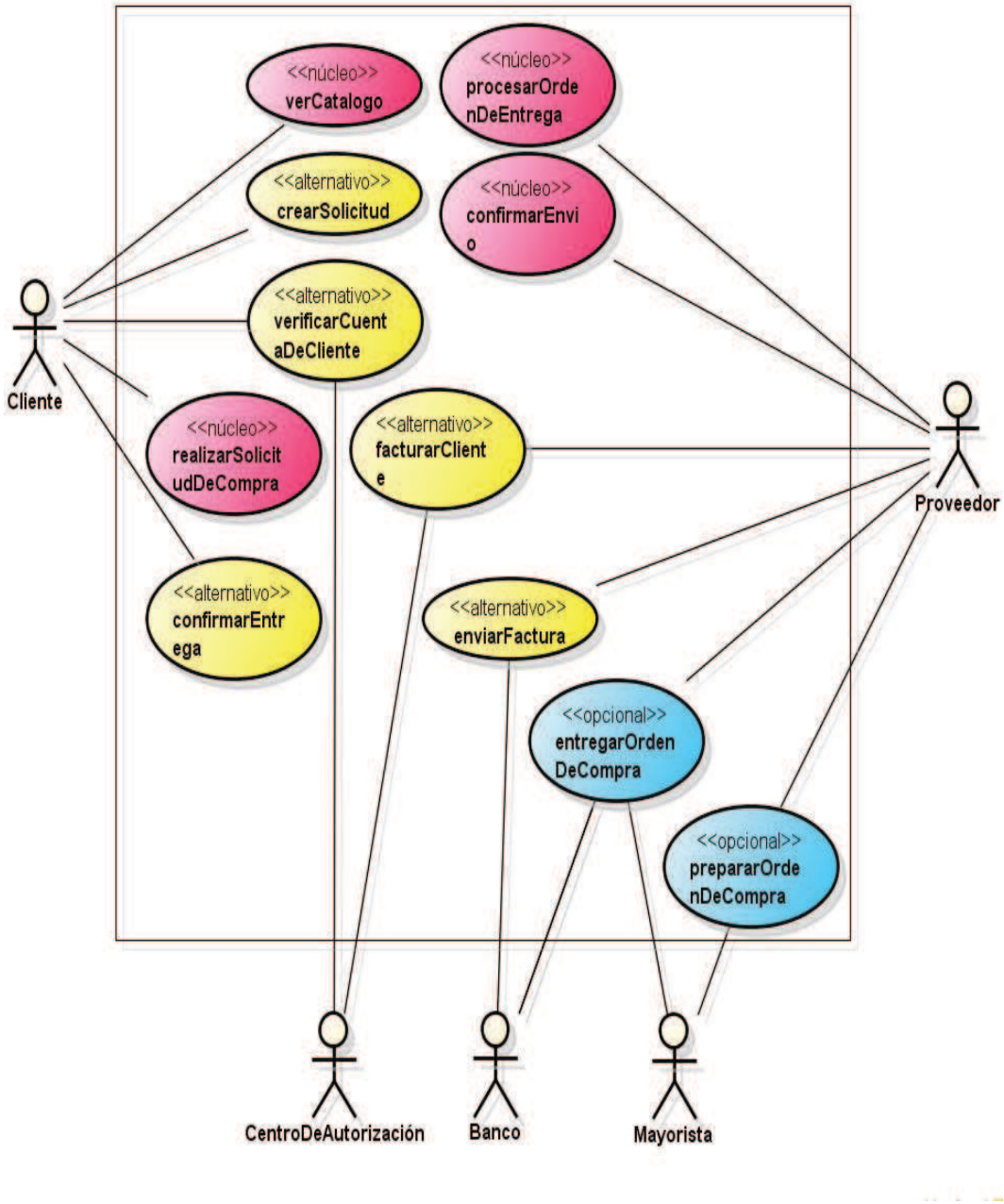


Figura 3.4 Casos de Uso para LPS de Comercio Electrónico

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

3.2.2.1.1 Documentación de casos de uso

Nombre del Caso de Uso:	verCatálogo
Categoría:	Núcleo
Resumen:	El cliente navega por los diferentes catálogos del internet, considera varios artículos del catálogo proporcionado por el proveedor, y los selecciona.
Actores:	Cliente (principal)
Precondiciones:	El sistema esté en funcionamiento
Descripción:	El cliente realiza una solicitud de catálogo. El sistema muestra al cliente varios catálogos para que el cliente busque. El cliente realiza una selección en los catálogos. El sistema confirma la disponibilidad de los artículos seleccionados del catálogo.
Alternativas:	A1. El sistema no se encuentra disponible, la solicitud no puede ser procesada. A4. El sistema indica que no existen artículos disponibles para dicha selección, y sugiere una nueva búsqueda.
Punto de variación:	Selección del catálogo: 1. El cliente selecciona los artículos del catálogo. 2. El cliente selecciona el proveedor, el cliente selecciona los artículos del catálogo.
Post condición:	El cliente ha realizado su selección de artículos con éxito.
Cuestiones pendientes:	Desplegar lista de proveedores, o mostrar catálogo de proveedores

Tabla 3.4 Caso de uso verCatálogo

Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	crearSolicitud
Categoría:	Alternativo
Resumen:	El cliente verifica cantidad de artículo seleccionados del catálogo y crea solicitud..
Actores:	Cliente (principal)
Precondiciones:	La selección de artículos este realizada con éxito.
Descripción:	El cliente solicita al sistema procesar el pedido con la selección realizada. El sistema verifica los fondos de operación del cliente. El sistema autoriza la solicitud e informa al cliente.
Alternativas:	A1. El cliente sale del sistema y termina la transacción. A3. El sistema no autoriza la solicitud.
Punto de variación	No existe punto de variación
Post condición:	La solicitud se ha creado con éxito.
Cuestiones pendientes	No existen cuestiones pendientes

Tabla 3.5 Caso de uso crearSolicitud

Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	realizarSolicitudDeCompra
Categoría:	Núcleo
Resumen:	El cliente solicita al sistema enviar una solicitud de compra al proveedor.
Actores:	Cliente (principal)
Precondiciones:	La solicitud debe estar creada con éxito
Descripción:	El cliente realiza una solicitud de compra. El sistema confirma la solicitud de compra.
Alternativas:	A2. El sistema no procesa la confirmación de la solicitud de compra, sugiere volver a realizar la solicitud.

Punto de variación:	Forma de Pago: 1. El cliente ingresa datos de la tarjeta de crédito para verificación de fondos. 2. El sistema verifica contrato y monto asignado a cada proveedor.
Post condición:	Solicitud de compra realizada con éxito.
Cuestiones pendientes:	Ingresar otra forma de pago

Tabla 3.6 Caso de uso realizarSolicitudDeCompra
Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	procesarOrdenDeEntrega
Categoría:	Núcleo
Resumen:	El proveedor solicita una orden de entrega, determina que el inventario está disponible para cumplir la orden, y muestra la orden.
Actores:	Proveedor (principal)
Precondiciones:	Solicitud de compra del cliente realizada con éxito.
Descripción:	El proveedor solicita al sistema una orden de entrega. El proveedor revisa que los artículos solicitados en la orden de entrega estén disponibles en el inventario. El proveedor indica al sistema el estado de la orden de entrega. El sistema indica al proveedor la orden de entrega y la información del inventario.
Alternativas:	A1. El sistema no cuenta con órdenes de entrega. A2. No existen suficientes artículos en el inventario para procesar la orden de entrega.
Punto de variación:	No existe punto de variación
Post condición:	La orden de entrega se procesa con éxito.
Cuestiones pendientes:	El proveedor da por terminado la venta y devuelve al cliente monto pagado.

Tabla 3.7 Caso de uso procesarOrdenDeEntrega
Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	confirmarEnvío
Categoría:	Núcleo
Resumen:	El proveedor prepara el envío de forma manual y luego confirma el envío.
Actores:	Proveedor
Precondiciones:	La orden de entrega se haya procesado con éxito.
Descripción:	El proveedor ingresa en el sistema la información del envió. El sistema actualiza el inventario. El sistema actualiza el estado de la orden de entrega. El sistema muestra al cliente el estado de la orden de entrega.
Alternativas:	No existen alternativas.
Punto de variación:	No existe punto de variación
Post condición:	Confirmación de envío exitosa.
Cuestiones pendientes:	Mostrar pantalla de servicio contratado para un rastreo del envío.

Tabla 3.8 Caso de uso confirmarEnvío

Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	confirmarEntrega
Categoría:	Alternativa
Resumen:	El cliente confirma la entrega.
Actores:	Cliente (principal)
Precondiciones:	La confirmación de entrega se ha llevado a cabo con éxito por parte del proveedor.
Descripción:	El cliente envía la confirmación de llegada de la orden de entrega. El sistema actualiza el estado de la orden de entrega.

	El sistema actualiza la solicitud realizada por el cliente.
Alternativas:	No existen alternativas
Punto de variación:	No existe punto de variación
Post condición:	La confirmación de entrega por parte del cliente se realiza con éxito.
Cuestiones Pendientes:	Enlace de sistema con el agente de entrega, para confirmar la recepción del cliente.

Tabla 3.9 Caso de uso confirmarEntrega

Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	enviarFactura
Categoría:	Alternativa
Resumen:	El proveedor envía una factura al cliente. Después de que el cliente confirma la entrega.
Actores:	Proveedor (principal), Banco
Precondiciones:	El cliente confirma entrega.
Descripción:	El proveedor consulta el estado de la orden de entrega. El sistema notifica que la orden de entrega ha sido recibida. El sistema consulta el contrato. El sistema notifica que el contrato existe, y solicita monto de compra. Los fondos son comprometidos, se crea una factura y se modifica el estado del pago. El sistema genera factura.
Alternativas:	No existen Alternativas.
Punto de variación:	No existe punto de variación
Post condición:	El cliente recibe factura.
Cuestiones Pendientes:	Envío de factura vía e-mail automáticamente.

Tabla 3.10 Caso de uso enviarFactura

Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	facturarCliente
Categoría:	Alternativa
Resumen:	El sistema recupera información de la tarjeta de crédito del cliente desde la cuenta del cliente y envía una solicitud de pago al centro de autorización.
Actores:	Proveedor (principal), Centro de Autorización
Precondiciones:	Cuenta de cliente activa.
Descripción:	El proveedor solicita al sistema información sobre el estado de la entrega de pedido de un cliente. El sistema notifica el estado de pedido del cliente. El sistema solicita la información de pago del cliente. El sistema envía la información al Centro de Autorización externo. El centro de autorización reconoce que ha facturado al cliente, y devuelve un número de referencia de facturación. El sistema envía el un número de referencia de facturación a la cuenta del cliente. El estado del pago es indicado en el sistema.
Alternativas:	A5. El centro de autorización no autoriza el pago, se reinicia el proceso.
Punto de variación:	No existe punto de variación.
Post condición:	El estado de pago es el correcto.
Cuestiones Pendientes:	No existen cuestiones pendientes.

Tabla 3.11 Caso de uso facturarCliente

Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del Caso de Uso:	verificarCuentaDeCliente
Categoría:	Alternativo
Resumen:	El cliente introduce los datos personales. El sistema crea una cuenta de cliente si no existe todavía.

Actores:	Centro de Autorización, Cliente
Precondiciones:	El cliente ingresa al sistema a registrarse.
Descripción:	El cliente ingresa los datos de su cuenta. El sistema valida los datos ingresados. El sistema muestra el estado al cliente indicándole proseguir o reintentar.
Alternativas:	A1, A2. Los datos ingresados no son correctos, el sistema alerta e indica volver a ingresarlos.
Punto de variación:	Registro cuenta: 1. El sistema guarda registro de cuenta con centro de autorización. 2. El sistema guarda registro de cuenta con centro de autorización, y el cliente guarda contrato con monto aprobado por cada proveedor.
Post condición:	La verificación de la cuenta del cliente se realiza con éxito.
Cuestiones pendientes:	No existen cuestiones pendientes

Tabla 3.12 Caso de uso verificarCuentaDeCliente

Elaborado por: (Inga F & Sarabia P, 2013)

3.2.2.2 Modelamiento De Características

En el modelamiento de características, los requerimientos de una LPS de comercio electrónico son vistos desde una perspectiva de reutilización. Los casos de uso núcleo se agrupan en una característica común, llamada Comercio Electrónico Núcleo, y se muestra como un paquete de casos de uso que consiste de los siguientes casos de uso verCatálogo, realizarSolicitudDeCompra, procesarOrdenDeEntrega, y confirmarEnvio. También hay dos características alternativas Cliente de Negocio y Cliente de Hogar que corresponden a los dos principales usos de esta línea de producción en los sistemas B2B y B2C, respectivamente. Los casos de uso que sólo se utilizan en los sistemas B2B (crearSolicitud, confirmarEntrega, y enviarFactura) se combinan en la característica alternativa llamada Cliente de Negocio. Los casos de uso que sólo se utilizan en los sistemas B2C

(verificarCuentaDelCliente y facturarCliente) se combinan en la función alternativa llamada del Cliente de Hogar. Existe una restricción de característica *exactamente-una-de* entre el Cliente de Negocio y el Cliente de Hogar, lo que significa que una y sólo una de estas características debe ser seleccionada para un sistema de comercio electrónico dado que es un miembro de la línea de productos:

«*exactamente-una del grupo de características*» {alternativa = Cliente de Negocio, Cliente de Hogar}

Los casos de uso opcionales de orden de compra (prepararOrdenDeCompra y entregaOrdenDeCompra) se combinan en una característica opcional llamada Orden De Compra, ya que siempre se reutilizan juntos.

La agrupación de casos de uso en características se representa con la notación de paquetes UML como se muestra en la Figura 3.5, y en forma tabular en la Tabla 3.7.

El grupo de característica está representada en la Tabla 3.13. La Figura 3.6 muestra un diagrama de dependencias entre las características utilizando la notación de estereotipos y Meta modelado.

La Figura 3.6 muestra una característica adicional, Banco, la cual es una característica opcional separada, ya que se necesita tanto para la característica alternativa Cliente de Hogar y la característica opcional Orden de Compra. Las características Cliente de Negocio y Orden de Compra son dos características explícitas que mutuamente incluyen la característica Banco ya que es una característica implícita de que no se puede seleccionar por sí mismo. La característica de grupo de Cliente *exactamente-una-de* contiene dos características mutuamente excluyentes: las características alternativas Cliente de Negocio y Cliente de Hogar.

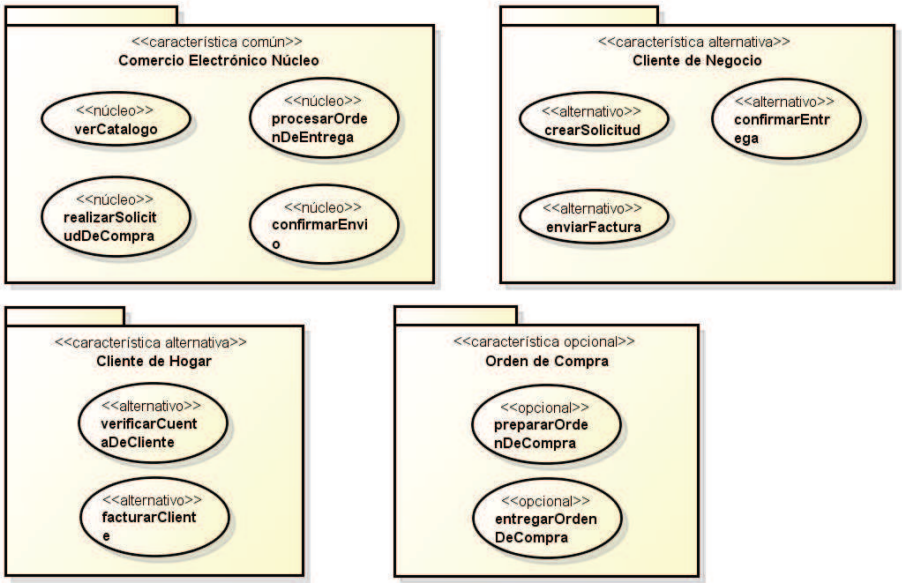


Figura 3.5 Dependencias Características/Casos de uso

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

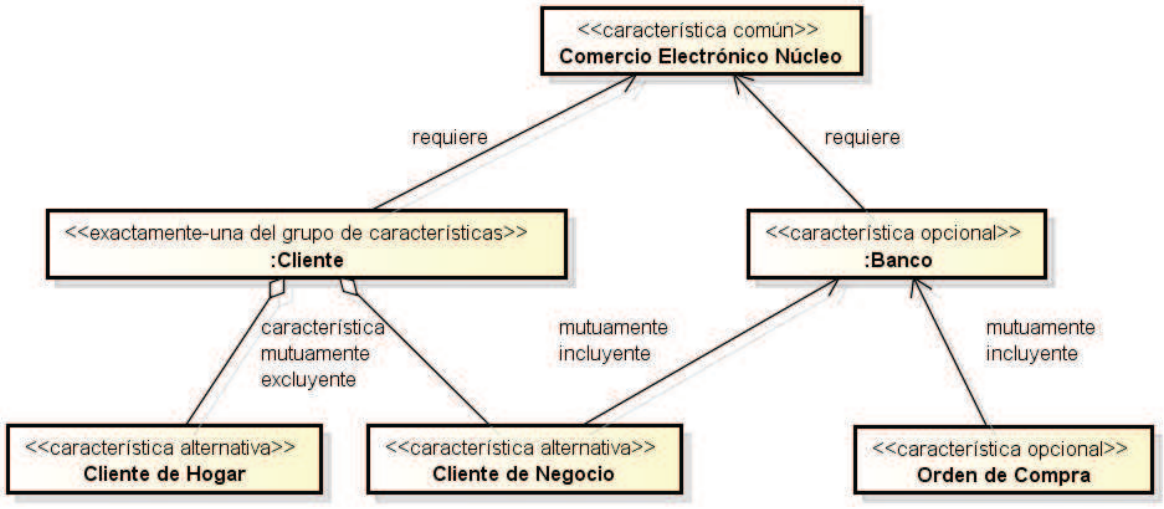


Figura 3.6 Modelo de características de LPS para Comercio Electrónico

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

Nombre de Característica	Categoría de la característica	Nombre del C.U.	Categoría del C.U.
Comercio	común	verCatálogo	núcleo

Electrónico Núcleo		realizarSolicitudDeCompra	núcleo
		procesarOrdenDeEntrega	núcleo
		confirmarEnvío	núcleo
Cliente de Negocio	alternativa	crearSolicitud	alternativo
		confirmaEnvío	alternativo
		enviarFactura	alternativo
Cliente de Hogar	alternativa	verificarCuentaDelCliente	alternativo
		facturarCliente	alternativo
Orden de Compra	opcional	prepararOrdenDeCompra	opcional
		entregaOrdenDeCompra	opcional

Tabla 3.13 Dependencia de Características /Casos de Uso de LPS para Comercio Electrónico

Elaborado por: (Inga F & Sarabia P, 2013)

Nombre del grupo de características	Categoría del grupo de características	Características en grupo de características	Categoría de característica
Cliente	exactamente-una-de	Cliente de Negocio	alternativa
		Cliente de Hogar	alternativa

Tabla 3.14 Grupo de Características en LPS para Comercio Electrónico

Elaborado por: (Inga F & Sarabia P, 2013)

3.2.2.3 Modelamiento Estático

Esta sección describe el modelo estático, el cual consiste de modelo contextual de la línea de producción y el modelo de clases de entidad.

3.2.2.3.1 Modelamiento contextual de la línea de producción de software

El modelo contextual de la línea de producción representa tres clases de usuarios externos, representados como actores: las clases Cliente núcleo, Proveedor, y usuario opcional externo Mayorista. Por otra parte, el actor Cliente se ha especializado en dos variantes: Cliente de Negocio y el Cliente de Hogar.

Hay dos clases opcionales externas del sistema: Centro de Autorización y el Banco. En esta línea de producción el diagrama de contexto (ver Figura 3.7) es muy similar al diagrama de casos de uso debido a que las clases externas corresponden a los actores en el diagrama de casos de uso.[6]

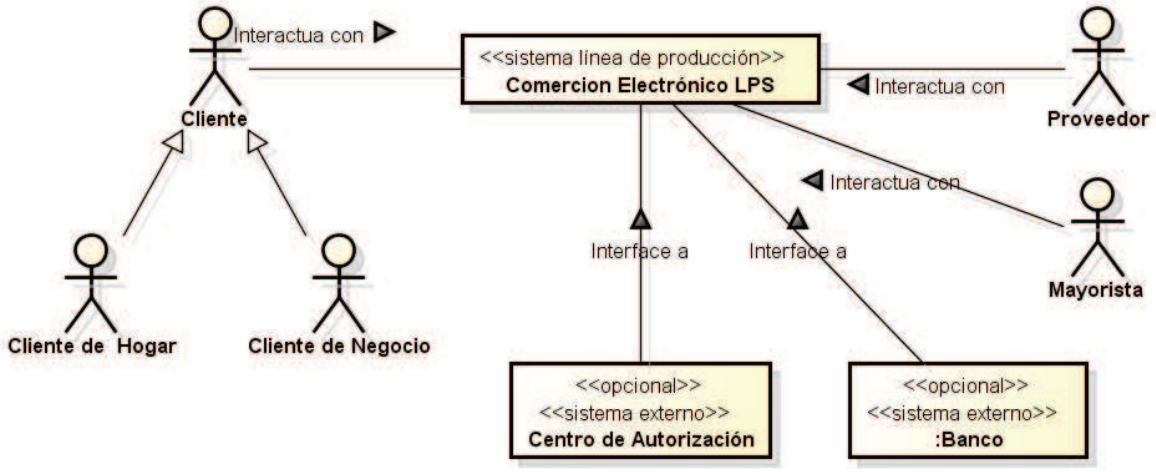


Figura 3.7 Diagrama de clases de contexto de SPL para comercio electrónico

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

Agente de apoyo para el comercio electrónico

En esta sección se describe un enfoque que utiliza agentes de software en la línea de producto de comercio electrónico. En este ejemplo, hay agentes de cliente y los agentes de servidor, donde cada agente define las reglas de negocio para un aspecto particular del dominio del problema de comercio electrónico. En esta aplicación, los agentes de cliente son los agentes de usuario que actúan en nombre de los usuarios y ayudan a los usuarios en el desempeño de sus tareas. Para ello, los agentes de cliente interactuar con los agentes de servidor. Los agentes de servidor reciben las peticiones de los agentes de cliente. Para satisfacer una solicitud de un agente de cliente, un agente de servidor normalmente interactúa con los objetos de servidor y con otros agentes [6].

El uso de agentes de software se ilustra conceptualmente en la figura 3.8 para sistemas B2B de comercio electrónico. En el problema de comercio electrónico, hay dos tipos de agentes de cliente: Agente del Cliente y el Agente del Proveedor. Hay una instancia del Agente del Cliente para cada cliente y una instancia del Agente del Proveedor para cada proveedor. Hay varios agentes de servidor, con muchas instancias de cada uno. Los agentes de servidor son:

Agente Pedido (una instancia por cada solicitud)\
Agente de Orden de Entrega (una instancia para cada orden de entrega)
Agente de factura (una instancia por cada factura)
Agente de facturación (una instancia por cada facturación a un cliente)
Agente de Orden de Compra (una instancia para cada orden de compra)

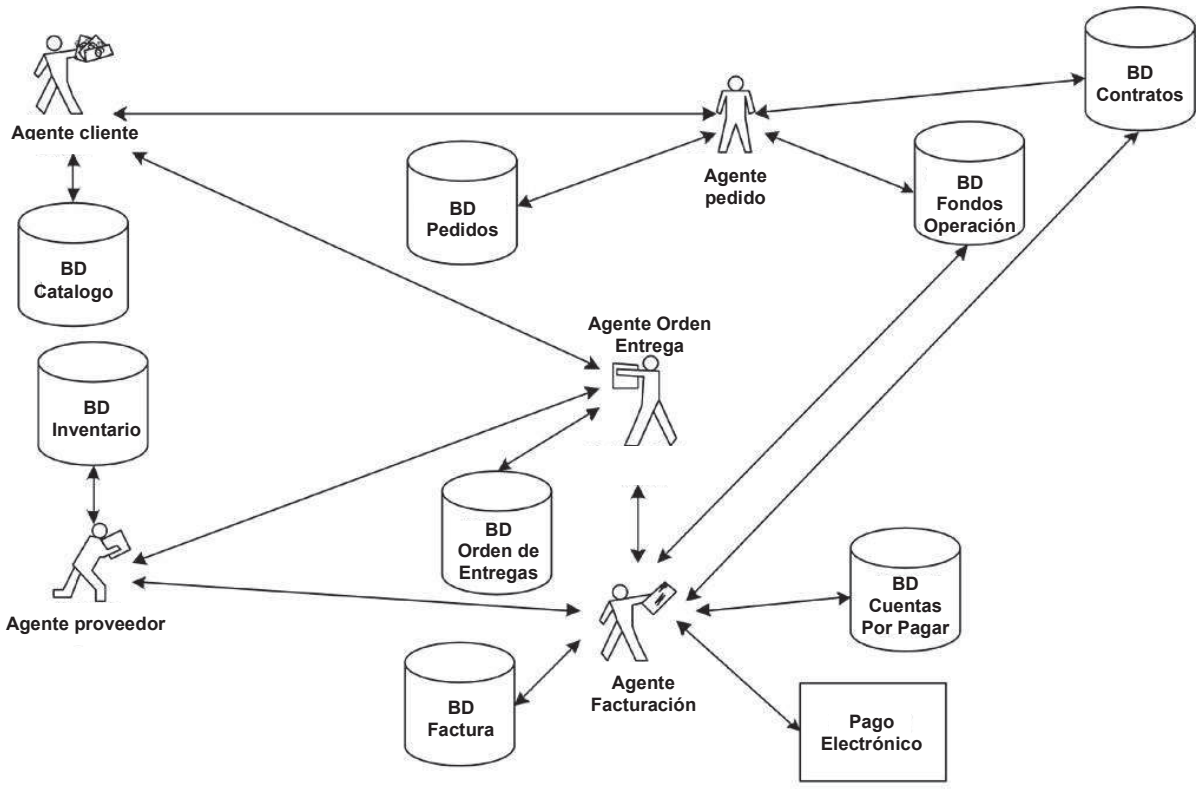


Figura 3.8 Sistema de comercio electrónico B2B basado en agentes: punto de vista conceptual

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2013)

Agente Cliente: (Ver Figura 3.8) ayuda al cliente que desea encargar uno o varios artículos de un catálogo. Este toma las solicitudes del cliente e interactúa con los agentes de servidor para impulsar el proceso de selección del cliente y para realizar un seguimiento de la situación. El cliente puede seleccionar varios artículos de un catálogo. Cuando un cliente completa la selección en el catálogo, el Agente Del Cliente actúa en nombre del cliente e inicia acciones determinadas. En particular, cuando el cliente realiza una selección de catálogo, el Agente Del Cliente envía una petición de solicitud a un agente de Solicitud

Agente Pedido: Es un agente del servidor que consulta varias bases de datos y se comunica con diversos agentes para garantizar la tramitación de una solicitud iniciada por el cliente. Se consulta la base de datos de contratos, base de datos de los contratos, y la base de datos de fondos de operación, así como la comunicación con el Agente del Cliente. Se envía una consulta de un contrato a la base de datos de los contratos para determinar si un contrato está en su lugar. Se envía una consulta de fondos a la base de datos de los fondos de operación para determinar si los fondos están en su lugar. Si la respuesta a ambas preguntas es positiva, el Agente de Solicitud autoriza la solicitud y envía el estado de la solicitud al Agente del Cliente. El Agente Del Cliente envía una solicitud de compra al Agente de Orden de Entrega.

Agente Proveedor: Es un agente de cliente que es instanciado para trabajar con el proveedor. Se recupera una orden de entrega del Agente de Orden de Entrega y ayuda al proveedor a cumplir con la orden. El Agente del Proveedor actualiza la base de datos del inventario y el estado del pedido se envía al Agente de Orden de Entrega y al Agente del Cliente. El cliente finalmente acusa recibo de las mercancías, y la orden de entrega se actualiza para reflejar la fecha de recepción.

El agente proveedor envía la factura al Agente De Factura en la organización del cliente. Cuando se le notifique por el Agente de Orden de Entrega que las mercancías han sido recibidas, el agente consulta la base de datos de facturas y la base de datos de operación de fondos (ver Figura 3.8). Si ambas respuestas son positivas, el Agente De Factura autoriza el pago y envía la

factura a la base de datos de las cuentas por pagar, que actualiza la cuenta. El Agente De Factura a continuación, envía la solicitud de pago electrónico para el banco del cliente.

Agente facturación: Se utiliza en aplicaciones B2C para manejar la validación y facturación de compra electrónica del cliente mediante la interacción con diversos agentes y servidores, así como con el centro de autorización de tarjeta de crédito, como se muestra en la Figura 3.8. El Agente de Orden de Compra (también se muestra en la figura 3.9) es un agente de servidor opcional que se refiere a la preparación y entrega de una orden de compra mediante la interacción con diversos agentes y servidores, así como con el mayorista externo y el banco.

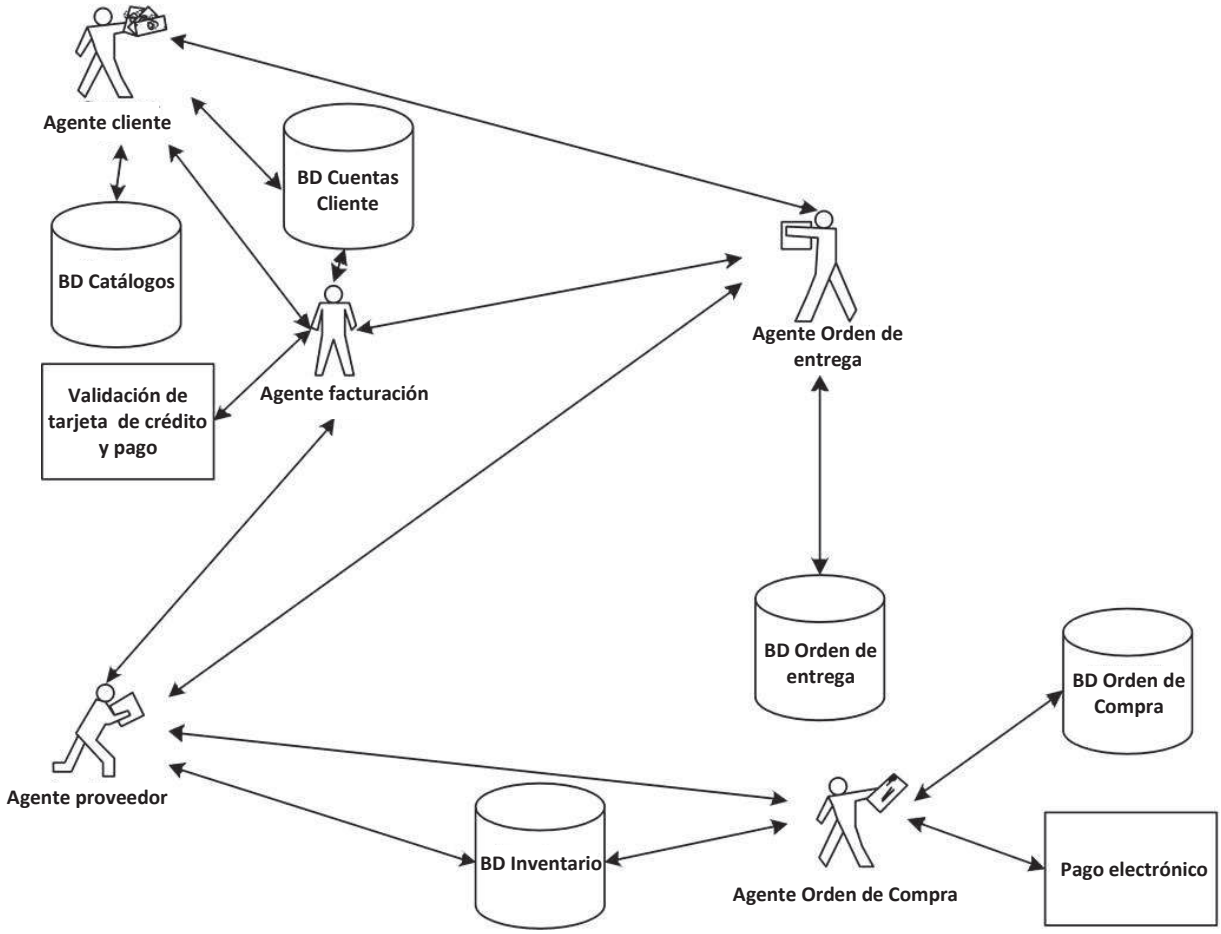


Figura 3.9 Sistema de comercio electrónico B2C basado en agentes: punto de vista conceptual

Fuente: (Gomma, 2007)
Traducido: (Inga F & Sarabia P, 2011)

3.2.2.3.2 Modelamiento estático de clases de entidad para el dominio del problema

Un modelo estático del dominio del problema se desarrolla y se representa en un diagrama de clases (ver Figura 3.10). Debido a que esta es una aplicación de uso intensivo de datos, el énfasis está sobre las clases de entidad, muchas de los cuales representan los datos almacenados en las bases de datos. Los objetos de contenedor de base de datos van a establecer una relación entre los objetos conceptuales y las bases de datos reales.

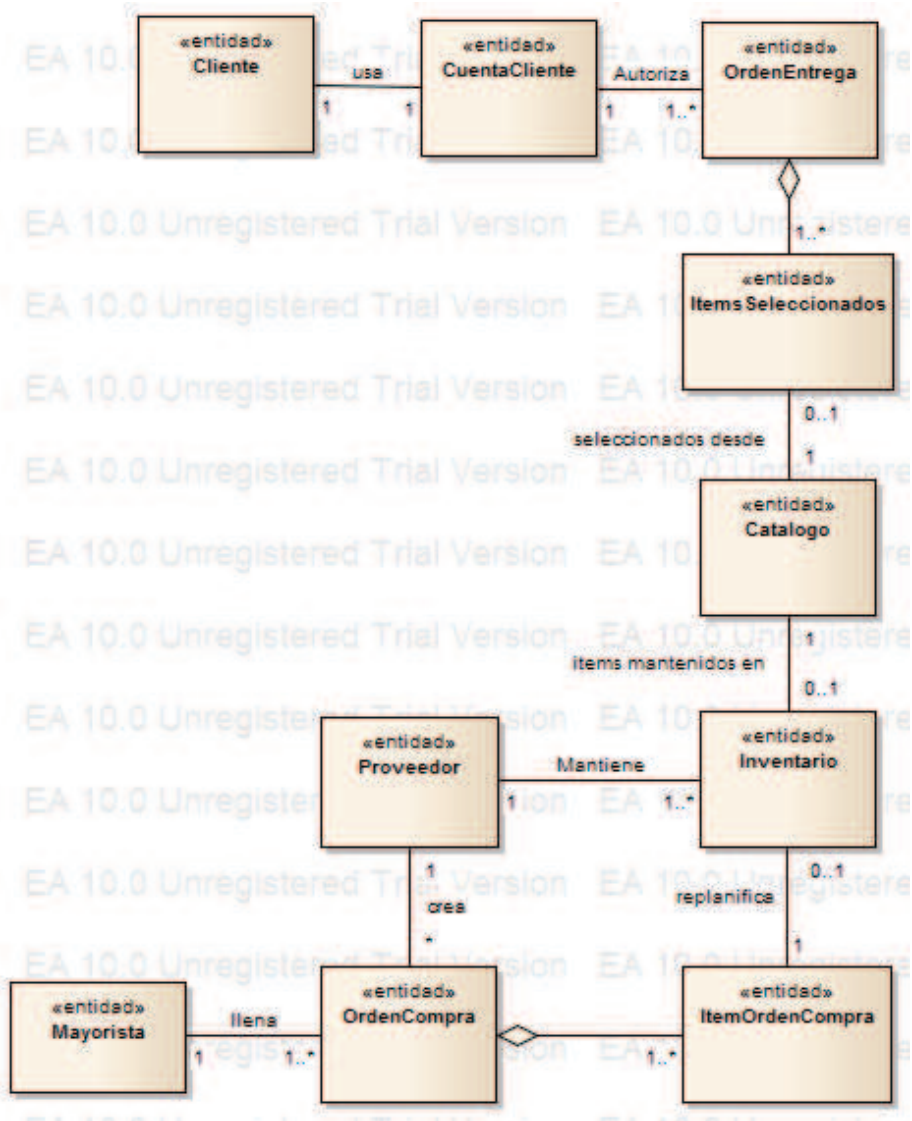


Figura 3.10 Modelo conceptual estático de clases de entidad para el comercio electrónico B2C

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

3.2.2.3.3 Clases Entidad

La LPS de comercio electrónico cuenta con dos grandes grupos de aplicaciones: aplicaciones B2C y aplicaciones B2B. El modelo estático de clases de entidad para aplicaciones B2C se representa en la figura 14.10, que muestra todas las entidades importantes del dominio del problema y las relaciones entre estas clases. Las clases incluyen clases de origen de los clientes (Cliente y Cuenta del Cliente), las clases de proveedor (tales como, Proveedores, Inventario y Catálogo), y las clases que tienen que ver con la orden del cliente (tales como, Orden de Entrega, que es una agregación de artículos seleccionados).

El modelo estático de clases entidad B2B se muestra en la Figura 3.11. Estas clases incluyen clases de negocios de los clientes (tales como, Requisición, Contrato, y Fondos De Operación), las clases del proveedor (tales como, Catálogo e Inventario), y las clases que tienen que ver con la orden del cliente y el pago (tales como, Orden de Entrega, Pago, y Factura).

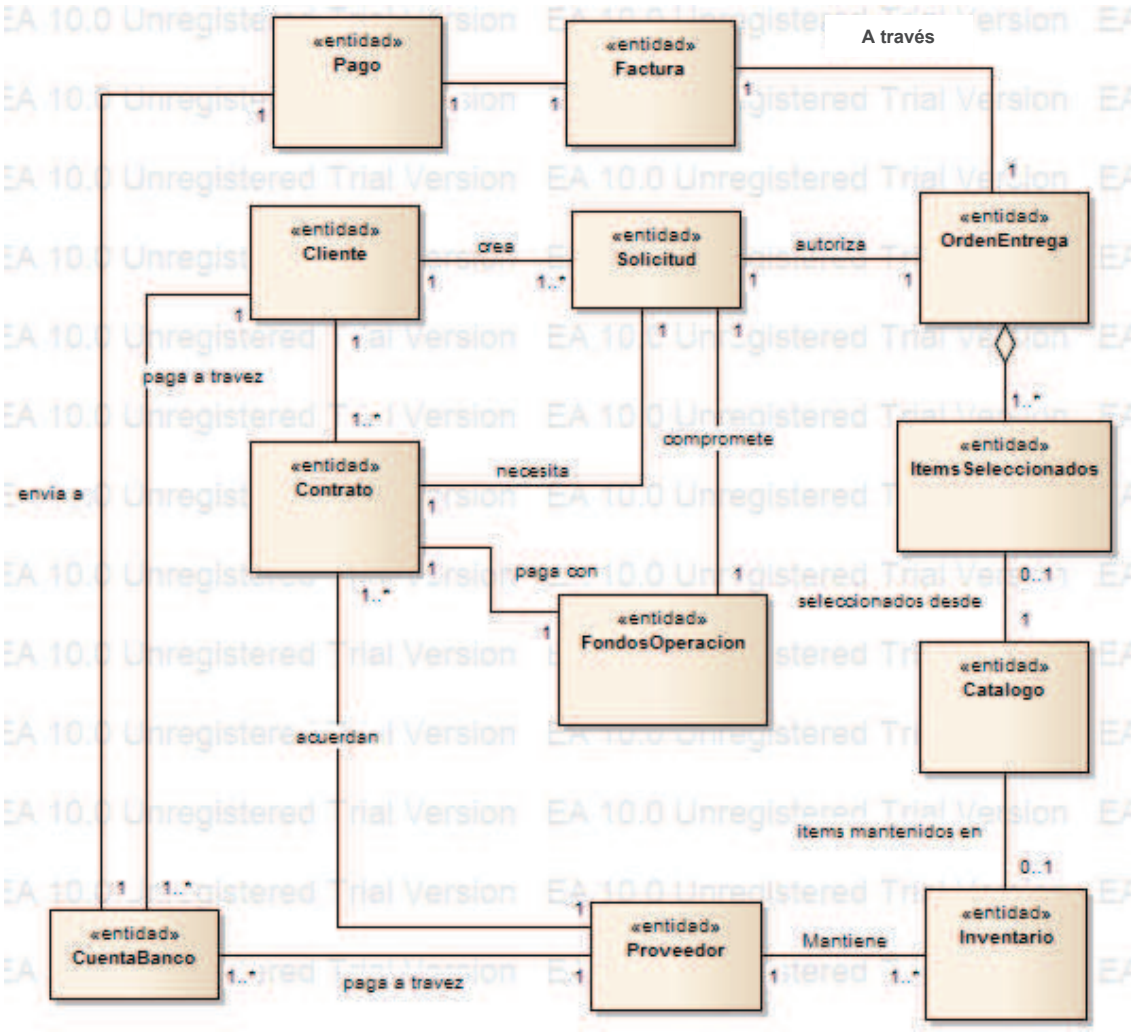


Figura 3.11 Modelo conceptual estático de clases de entidad para comercio electrónico B2B.

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

El modelo estático integrado de clases de entidad para la línea de producción se muestra en la Figura 3.12. En el modelo de línea de producción estático, las clases tienen un segundo estereotipo para describir la categoría de reutilización de cada clase: o «núcleo» u «opcional». Las clases como Cliente y Catálogo, que existen en los modelos estáticos de las aplicaciones B2C y B2B, se convierten en clases núcleo en el modelo de las líneas de producción estático. Las clases que aparecen en sólo uno de los modelos estáticos, tales como Contrato (que sólo aparece en el modelo estático B2B) y Cuenta del Cliente (que sólo aparece en el modelo estático B2C), se convierten en clases opcionales en el modelo de la línea de producción estático. Las clases de

entidad que pudieran existir, ya sea en una aplicación B2B o B2C, como Orden de Compra, también se convierte en clases opcionales en el modelo de la línea de producción estático [6].

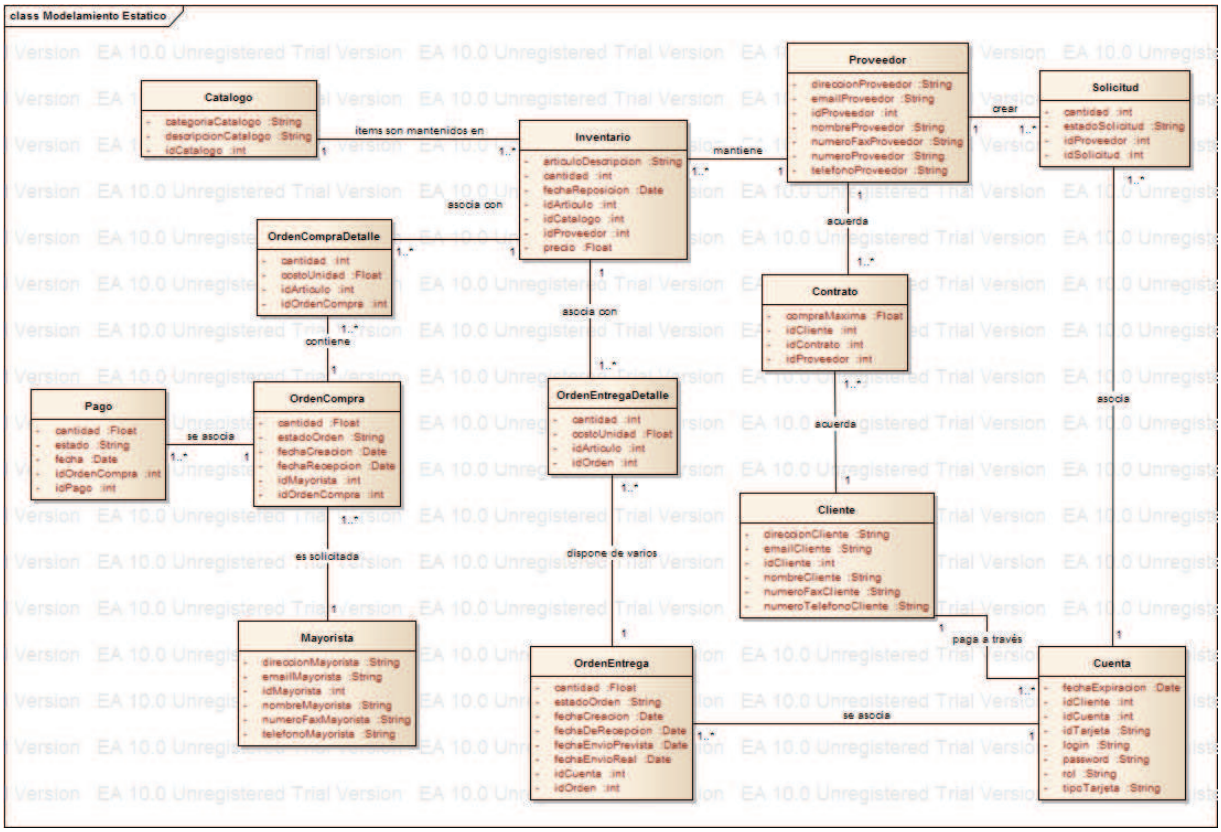


Figura 3.12 Modelo conceptual estático para las clases entidad de LPS de comercio electrónico

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

3.2.2.4 Modelamiento Dinámico

Para cada caso de uso, un diagrama de comunicación es desarrollado representando los objetos que participan en el caso de uso y la secuencia de mensajes transmitidos entre ellos.

3.2.2.4.1 Diagramas de comunicación

Modelamiento dinámico para verCatálogo

En el diagrama de comunicación para el caso de uso verCatálogo (Figura 3.13), la Interfaz del Cliente interactúa con el Agente del Cliente, que a su vez

se comunica con el Servidor de Catálogo. Las descripciones de los mensajes son los siguientes:

A1: El cliente realiza una solicitud de catálogo a través de la Interfaz Del Cliente.
A2: El Agente de Cliente es instanciado para ayudar al cliente. Sobre la base de solicitudes del cliente, el Agente De Cliente selecciona uno o varios catálogos para que el cliente busque.
A3: El Agente de Cliente solicita información desde el Servidor de Catálogo.
A4: El Servidor de Catálogo envía la información del catálogo al Agente de Cliente.
A5: El Agente de Cliente envía la información a la Interfaz del Cliente.
A6: La Interfaz del Cliente muestra el catálogo para el cliente.
A7: El cliente realiza una selección de catálogo a través de la Interfaz del Cliente.
A8: La Interfaz del Cliente pasa la solicitud al Agente de Cliente.
A9: El Agente de Cliente solicita la selección de catálogo del Servidor de Catálogo.
A10: El Servidor de Catálogo confirma la disponibilidad de los artículos del catálogo al Agente de Cliente.
A11: El Agente de Cliente envía la información a la Interfaz del Cliente.
A12: La Interfaz del Cliente muestra la confirmación del catálogo al cliente.

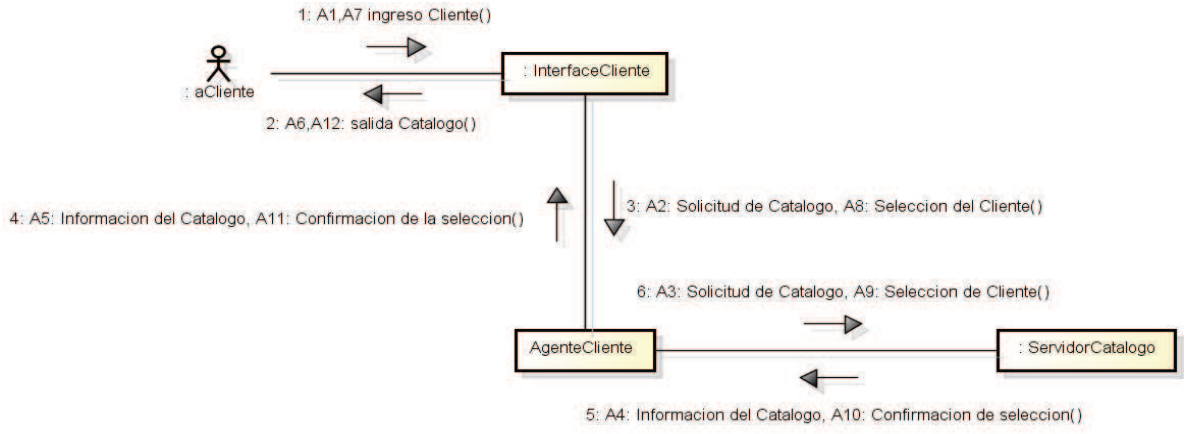


Figura 3.13 Modelamiento dinámico para verCatálogo

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

Modelamiento dinámico para realizarSolicitudDeCompra

En el diagrama de la comunicación para el caso realizarSolicitudDeCompra (Figura 3.14), el Agente de Cliente envía una solicitud al Agente de Orden Entrega y recibe una confirmación. Las descripciones de los mensajes son los siguientes:

P1: El cliente realiza una solicitud de compra.
P2: Interfaz de Cliente envía la solicitud de compra al Agente del Cliente.
P3: Agente de Cliente crea una instancia de Agente de Orden de Entrega y envía la solicitud de compra a la misma.
P4: Agente de Orden de Entrega crea una nueva orden de entrega y la almacenarla en el Servidor de Orden de Entrega.
P5: Agente de Orden de Entrega envía una confirmación de solicitud de compra con el ID de la orden al Agente de Cliente.
P6: El Agente de Cliente envía la confirmación de la solicitud de compra a la interfaz del cliente.
P7: La Interfaz de Cliente muestra el estado de solicitud de compra para al Cliente.

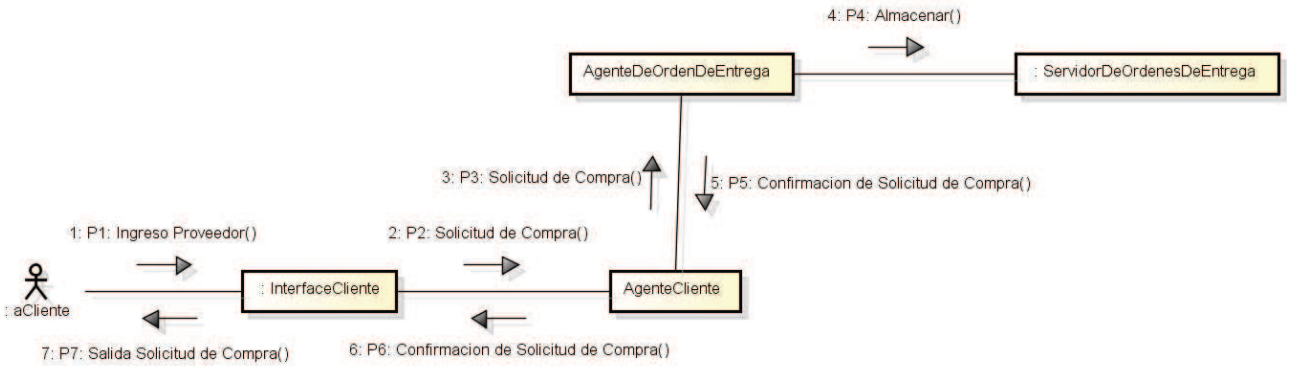


Figura 3.14 Modelamiento dinámico para realizarSolicitudDeCompra

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013))

Modelamiento dinámico para procesarOrdenDeEntrega

En el diagrama de la comunicación para el siguiente caso de uso, procesarOrdenDeEntrega (Figura 3.15), el Agente Proveedor consulta el

Agente de Orden de Entrega, y el Agente de Orden de Entrega selecciona una orden de entrega. El Agente Proveedor revisa el inventario y muestra la información de pedidos e inventario al proveedor a través de la interfaz de usuario. Las descripciones de los mensajes son los siguientes:

C1: El proveedor solicita una orden de entrega.

C2: delante de proveedores de la interfaz de la solicitud al agente proveedor.

C3: Agente Proveedor envía la solicitud a fin de Agente de la entrega de pedidos.

C4, C5: Agente de la entrega de pedidos selecciona una orden de entrega mediante la consulta de servidor de entrega de pedidos.

C6: Agente de la entrega de pedidos envía la orden de entrega al Agente de Proveedores.

C7, C8: Agente Proveedor comprueba que los artículos están disponibles en inventario.

C9: Agente Proveedor envía el estado para la interfaz de Proveedor.

C10: Interfaz de proveedor muestra el orden de entrega y la información del inventario al proveedor.

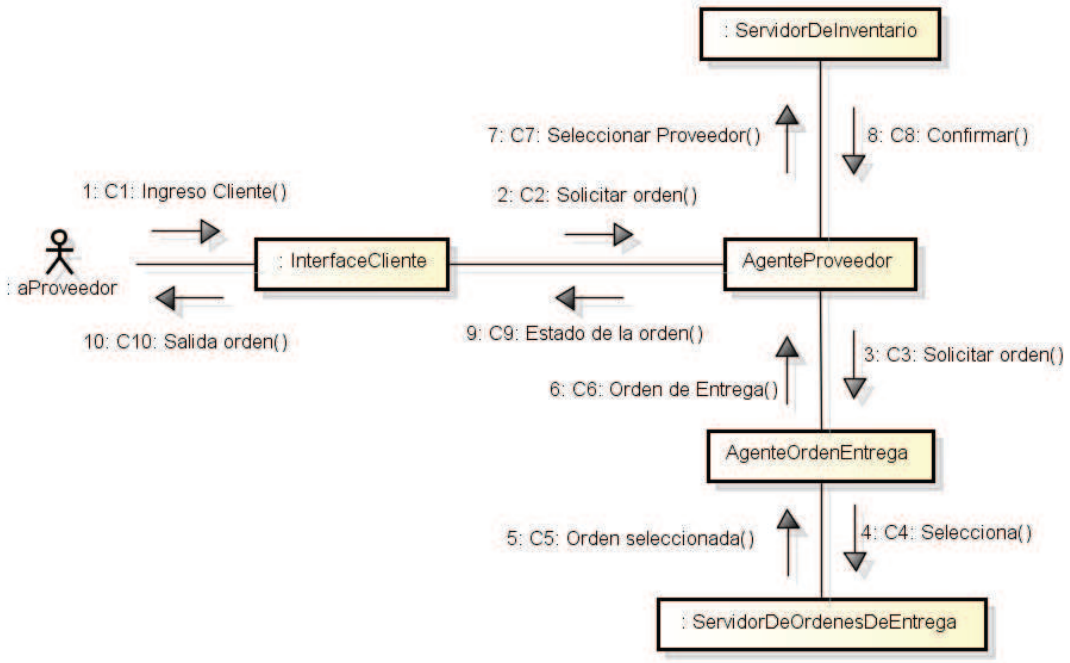


Figura 3.15 Modelamiento dinámico para procesarOrdenDeEntrega

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

Modelamiento dinámico para Confirmar Envío

En el diagrama de la comunicación para el caso de uso de confirmarEnvío (Figura 3.16), el proveedor prepara el envío de forma manual. El proveedor confirma el envío introduciendo la información de envío, además de la fecha de envío. El Agente Proveedor actualiza el inventario, y el estado de su pedido es enviado al Agente de Entrega del Pedido y al Agente del Cliente, el cual muestra el estado del pedido al cliente. Las descripciones de los mensajes son los siguientes [6]:

S1: El proveedor ingresa la información de envío.

S2: La Interfaz de Proveedor envía la solicitud de confirmación del envío con el Agente Proveedor.

S3: El Agente Proveedor actualiza el inventario almacenado en servidor de inventario.

S4: El Agente Proveedor envía el estado de la orden al Agente de Orden de Entrega.

S5: El Agente de Orden de Entrega actualiza el Servidor de Ordenes de Entrega.

S6: El Agente de Ordenes de Entrega envía el estado de la orden al Agente Cliente.

S7: El Agente Cliente envía el estado del pedido a la Interfaz del Cliente.

S8: La Interfaz del Cliente muestra el estado de la orden al cliente.

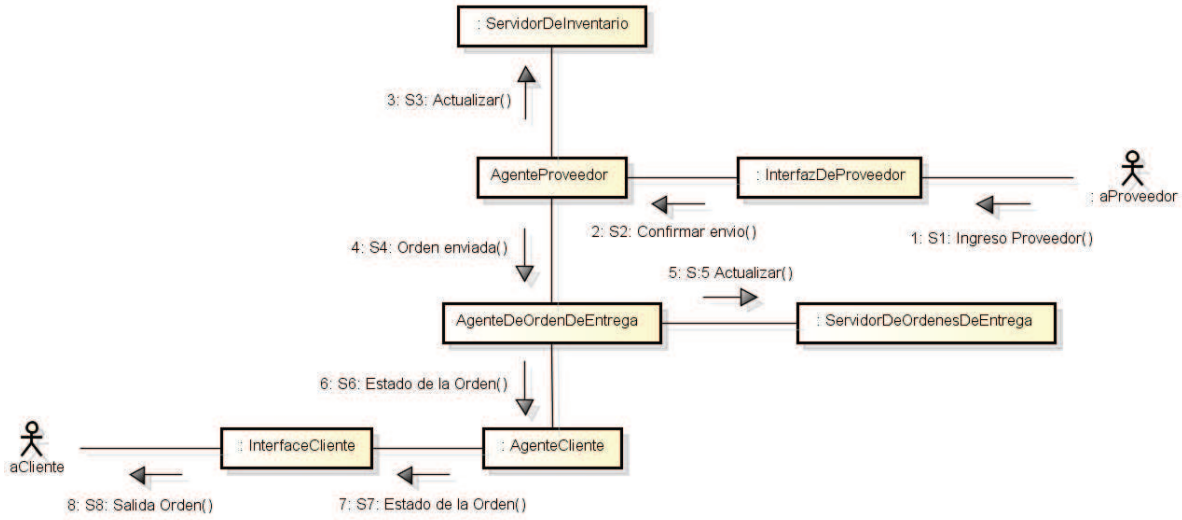


Figura 3.16 Modelamiento dinámico para Confirmar Envío

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

3.2.3 CÓDIGO

Para transformación de modelos utilizaremos la herramienta Enterprise Architect, la cual nos permitirá generar varios modelos PIM y PSM.

El componente central de MDA en Enterprise Architect es un motor de transformación basado en plantillas que genera los elementos del modelo PSM de una fuente de PIM.

Las plantillas permiten la generación de artefactos PSM muy específicos a partir de PIM. Las plantillas de transformación de MDA son similares, y tienen la misma sintaxis que las plantillas de generación de código dentro de EA.

3.2.3.1 El Modelo Independiente de Plataforma PIM

Este diagrama muestra un PIM simple con las clases, que muestra las propiedades abstractas y las relaciones entre estas. Este PIM no tiene propiedades específicas de la plataforma, ya que es una representación abstracta.

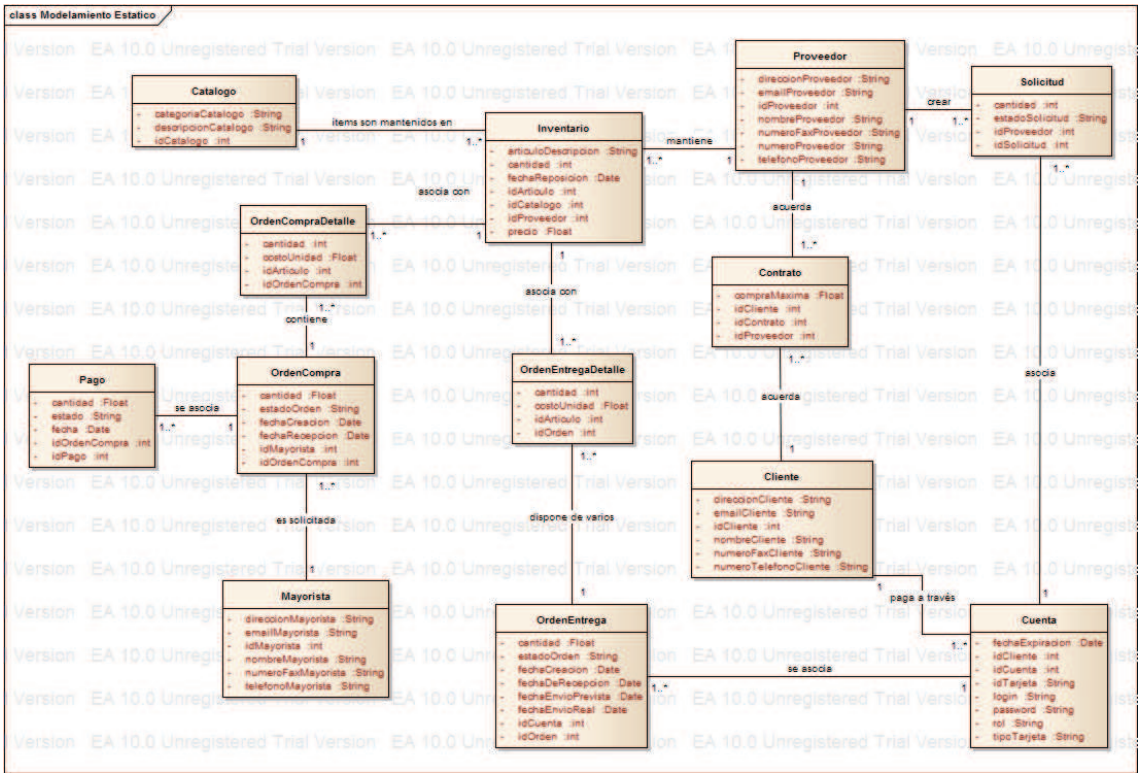


Figura 3.17 Modelo independiente de la plataforma

Fuente: (Gomma, 2007)
Elaborado por: (Inga F & Sarabia P, 2013)

3.2.3.2 Seleccionar e Iniciar la Transformación

Selección de las clases de EA y elegir "Transform" del menú de contexto. Aquí se puede elegir los elementos PIM a transformar, la transformación a ser invocada, y un paquete de destino, si es necesario.

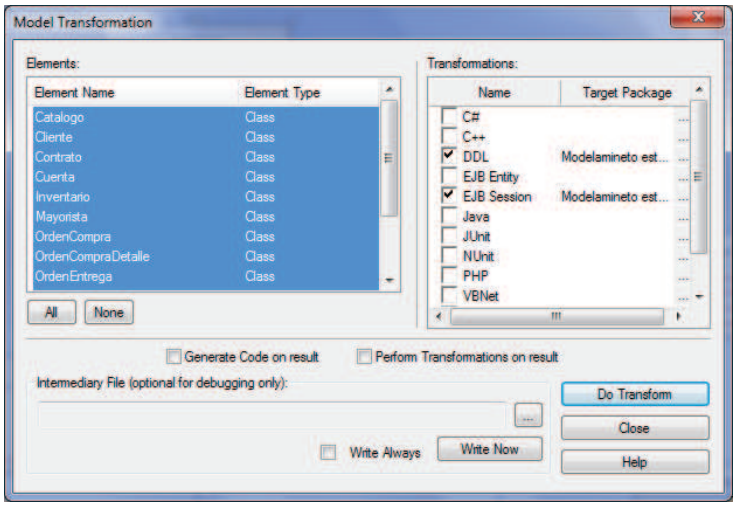


Figura 3.18 Pantalla de opciones de generación de transformaciones

Elaborado por: (Inga F & Sarabia P, 2013)

3.2.3.3 Transformación en progreso

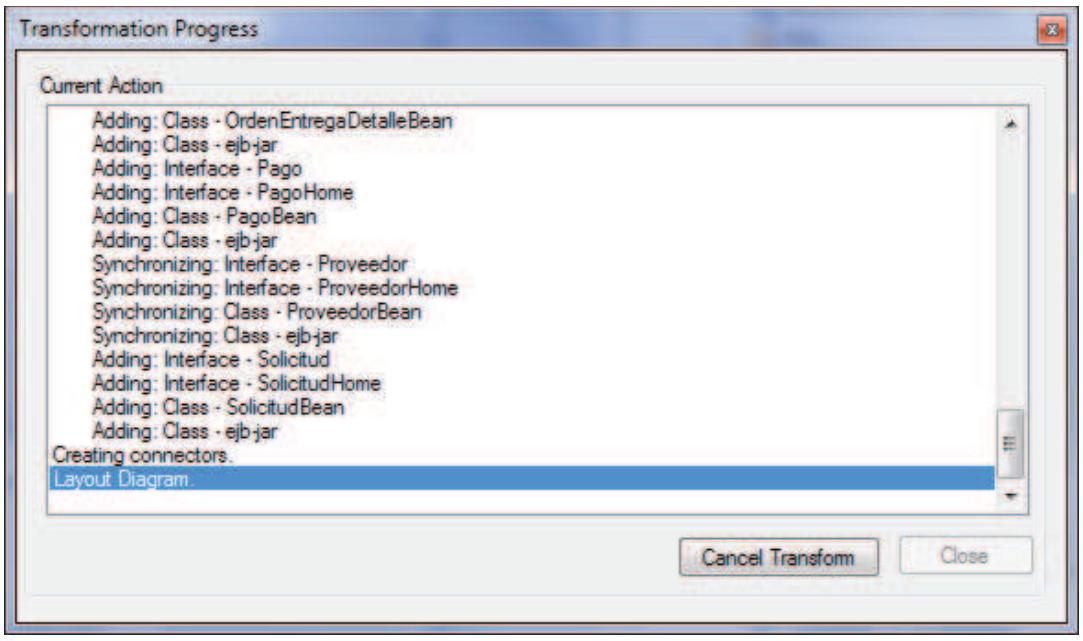


Figura 3.19 Generación de EJBs java

Elaborado por: (Inga F & Sarabia P, 2013)

3.2.3.4 Paquetes creados durante la transformación

Para la transformación de EJB de Sesión, en el siguiente diagrama se muestra los paquetes PSM creados - una para cada Bean de sesión EJB creado a partir de las clases de base de PIM.

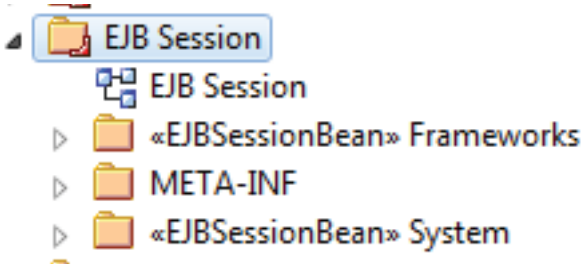


Figura 3.20 Estructura de un EJB generado

Elaborado por: (Inga F & Sarabia P, 2013)

3.2.3.5 Generación de PSM - Ejemplo de un Bean Creado a partir de la Transformación MDA

Este diagrama muestra un ejemplo de un EJB de Sesión PSM. Como PSM es fuertemente vinculado al dominio de destino, Java y EJB propiedades específicas, el código y la configuración son incluidos en el modelo. Estos se derivan lógicamente durante la transformada de acuerdo con las reglas de la plantilla transformada.

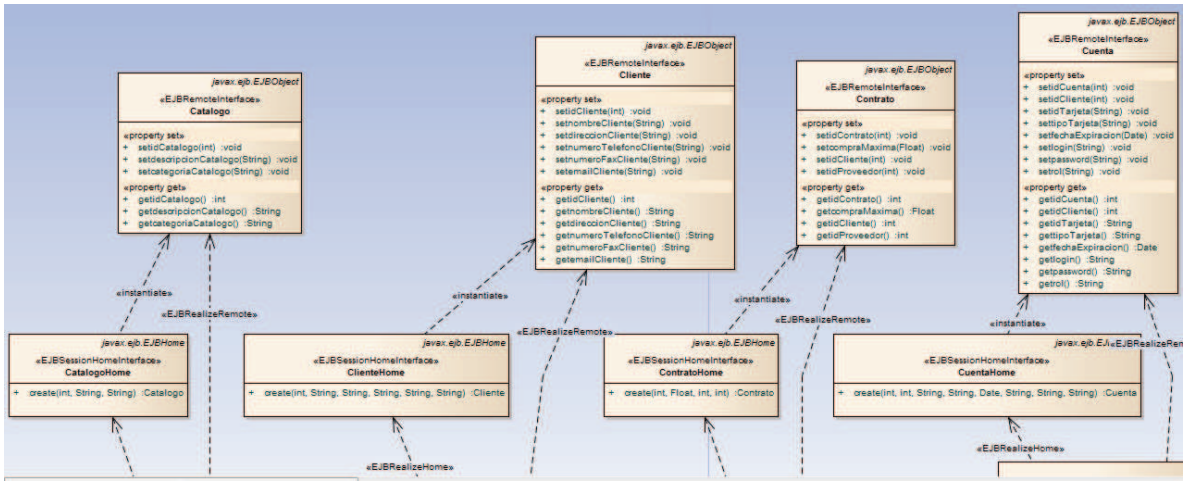


Figura 3.21 Ejemplo de generación de PSM.

Elaborado por: (Inga F & Sarabia P, 2013)

A medida que el PSM contiene propiedades específicas de la plataforma, el código fuente generadas puede ser adaptado a la plataforma de destino bastante bien. Esto se traduce en código de alta calidad que requiere menos trabajo manual.

3.2.4 PRUEBAS

3.2.4.1 Plan de Pruebas del Sistema

De acuerdo a lo establecido en la estrategia de desarrollo para la aplicación de MDA a LPS, se debe definir un plan de pruebas; para este trabajo se lo ha enfocado en el diseño de pruebas de caja negra que se orientan a la búsqueda de errores en las siguientes categorías:

- Funciones incorrectas o faltantes
- Errores de interface
- Errores de acceso a base de datos
- Errores de eficiencia

Para las pruebas de los componentes B2C y B2B, se definen casos de prueba bajo el siguiente esquema [12]:
Caso de prueba n

Esquema de un caso de prueba	
Nombre proceso:	Proceso n
Descripción proceso:	Acción del proceso
Escenario:	Acción del usuario
Entradas:	Datos de entrada del proceso
Pre-condiciones:	Requerimientos para validar el proceso
Procedimientos:	Pasos para validar el proceso
Resultados:	Que se obtiene al validar el proceso
Observaciones:	Observaciones generales de la prueba

Tabla 3.15 Esquema de un caso de prueba

Elaborado por: (Inga F & Sarabia P, 2013)

A continuación se definen los casos de prueba para *los prototipos B2C y B2B*

Ver Catalogo – componente núcleo	
Nombre proceso:	Ver catálogo
Descripción proceso:	Búsqueda y selección de artículos del catálogo.
Escenario:	El usuario ingresa el sistema y debe seleccionar uno o varios artículos mostrados en el catálogo.
Entradas:	Selección de artículos del catálogo.
Pre-condiciones:	Deben existir artículos en el catálogo.
Procedimientos:	1. En la página principal del sistema se muestra el catálogo de artículos. 2. Seleccionar un artículo del catálogo. 3. En la ventana emergente ver la información del artículo y “Aceptar” para agregarlo al carrito de compras.
Resultados:	Todos los artículos seleccionados se reservan para proceder con la creación de una solicitud de compra.
Observaciones:	Es necesario tener definidos los artículos que estarán en el catálogo de artículos.

Tabla 3.16 Caso de prueba Ver Catálogo

Elaborado por: (Inga F & Sarabia P, 2013)

Realizar solicitud de compra – componente núcleo	
Nombre proceso:	Realizar solicitud de compra
Descripción proceso:	El cliente solicita al sistema enviar una solicitud de compra al proveedor.
Escenario:	Existen solicitudes de pedido configuradas, validadas y se desea que el sistema envíe una solicitud de compra al proveedor por cada solicitud.
Entradas:	Solicitudes de pedido.
Pre-condiciones:	Solicitudes de pedido configuradas y validadas

Procedimientos:	1. Con las solicitudes de pedido se procede a enviar una solicitud de compra al proveedor por cada solicitud escogiendo la opción “Crear pedido”.
Resultados:	Solicitud de compra enviada al proveedor.
Observaciones:	Es necesario tener solicitudes de pedido configuradas y validadas.

Tabla 3.17 Caso de prueba Realizar solicitud de compra
Elaborado por: (Inga F & Sarabia P, 2013)

Procesar orden de entrega – componente núcleo	
Nombre proceso:	Procesar orden de entrega
Descripción proceso:	El proveedor solicita una orden de entrega de una lista de órdenes disponibles, determina que el inventario está disponible para cumplir con el detalle de la orden de entrega, y muestra a detalle de la orden al proveedor.
Escenario:	El proveedor ingresa al sistema y selecciona una orden de entrega de una lista de órdenes disponibles.
Entradas:	Orden de entrega.
Pre-condiciones:	Deben existir órdenes de entrega por procesar.
Procedimientos:	1. El proveedor ingresa el sistema. 2. El proveedor selecciona una orden de entrega de una lista de órdenes de entrega. 3. Ve el detalle de una orden de entrega (artículos). 4. Verifica la disponibilidad en el inventario de cada artículo contenido en la orden de entrega.
Resultados:	El proveedor verifica la disponibilidad de los artículos de la orden de entrega contra el inventario.
Observaciones:	La orden de entrega debe contener al menos un artículo.

Tabla 3.18 Caso de prueba Procesar orden de entrega
Elaborado por: (Inga F & Sarabia P, 2013)

Confirmar envío – componente núcleo	
Nombre proceso:	Procesar orden de entrega
Descripción proceso:	El proveedor prepara el envío de forma manual y luego confirma el envío.
Escenario:	El proveedor consulta el detalle de una orden de entrega y verifica la cantidad solicitada de cada artículo, al comprobar la existencia ingresa la fecha de envío y selecciona “Confirmar envío”.
Entradas:	Orden de entrega seleccionada.
Pre-condiciones:	Debe existir un orden de entrega para confirmar el envío.
Procedimientos:	1. El proveedor obtiene la información de una orden de entrega. 2. Ingresa la fecha de envío y selecciona “Confirmar envío”.
Resultados:	La orden de entrega es enviada.
Observaciones:	Los detalles de la orden de entrega deben estar validados.

Tabla 3.19 Caso de prueba Confirmar envío
Elaborado por: (Inga F & Sarabia P, 2013)

Crear solicitud – componente alternativo B2B	
Nombre proceso:	Crear solicitud
Descripción proceso:	El cliente realiza una solicitud de pedido.
Escenario:	El cliente realiza una solicitud de pedido, el sistema tiene que verificar datos del contrato y fondos de operación acordados para proceder con la solicitud realizada.
Entradas:	Solicitud de pedido.
Pre-condiciones:	Debe existir un pedido de orden de compra.
Procedimientos:	1. El proveedor tiene sus artículos agrupados por proveedor. 2. Selecciona uno de la lista.

	3. Selecciona la opción “Crear el pedido”. 4. El sistema busca el contrato establecido entre el cliente y el proveedor. 5. Verifica que el total del pedido no sobrepasen los fondos establecidos en el contrato. 6. Si existen suficientes fondos de operación para la solicitud del pedido, el sistema autoriza la solicitud e informa al cliente.
Resultados:	El pedido es creado e indica la información de dicha acción.
Observaciones:	El pedido debe contener al menos un artículo y el cliente debe tener un contrato con el proveedor.

Tabla 3.20 Caso de prueba Crear solicitud
Elaborado por: (Inga F & Sarabia P, 2013)

Confirmar entrega – componente B2B	
Nombre proceso:	Confirmar entrega
Descripción proceso:	El cliente confirma la recepción del pedido.
Escenario:	Al recibir su pedido el cliente ingresa al sistema y confirma el estado del pedido.
Entradas:	Un pedido creado.
Pre-condiciones:	Debe existir un pedido creado.
Procedimientos:	1. El cliente ingresa al sistema. 2. Selecciona la opción “Envíos” para ver los pedidos creados. 3. Selecciona un pedido de la lista. 4. Ingresa la fecha de recepción de su pedido y “Guarda”.
Resultados:	El pedido cambia de estado y los fondos de operación son comprometidos para el pago.
Observaciones:	El pedido debe contener al menos un artículo y el cliente debe tener un contrato con el proveedor.

Tabla 3.21 Caso de prueba Confirmar entrega
Elaborado por: (Inga F & Sarabia P, 2013)

Verificar cuenta cliente – componente B2C	
Nombre proceso:	Verificar cuenta cliente
Descripción proceso:	El cliente introduce los datos de su tarjeta para proceder con el pago de su pedido.
Escenario:	Al tener el pedido listo para su pago el cliente ingresa los datos de la tarjeta de crédito para que estos sean validados y proceder con la compra.
Entradas:	Datos de la tarjeta de crédito.
Pre-condiciones:	Debe existir el pedido creado.
Procedimientos:	1. El cliente tiene su pedido ya creado. 2. Ingresa los datos de la tarjeta de crédito al momento de pagar. 3. El sistema envía estos datos a un centro de autorización para la verificación del mismo. 4. El centro de autorización confirma los datos y se procede con el pago del pedido.
Resultados:	El pedido es cancelado con la tarjeta de crédito.
Observaciones:	Debe existir un pedido creado.

Tabla 3.22 Caso de prueba Verificar cuenta cliente
Elaborado por: (Inga F & Sarabia P, 2013)

Aplicación del plan de pruebas

Caso de prueba Nº 1			
Nº caso	**Datos entrada**	**Datos obtenidos**	**Datos esperados**
1	Selección de artículos del catálogo.	Error de aplicación, en la ventana emergente no se muestra la información del artículo.	Artículos seleccionados almacenados en sesión (carro de compras).

Tabla 3.23 Caso de Prueba Nº 1
Elaborado por: (Inga F & Sarabia P, 2013)

El caso de prueba número uno resultó con errores, a continuación se presenta la pantalla con la información descrita en el caso de prueba:

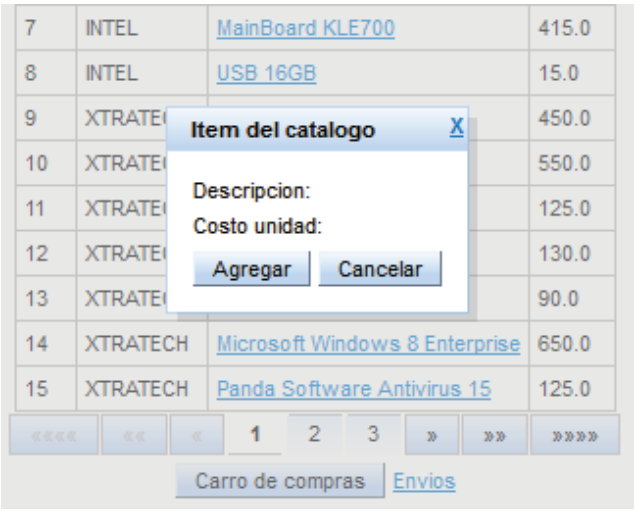


Figura 3.22 Selección de un artículo del catálogo
Elaborado por: (Inga F & Sarabia P, 2013)

Para rectificar el error indicado en el caso de prueba número uno, procederemos a modificar los atributos indicados en el controlador de la pantalla. Así de esa manera se corregirá dicho error.

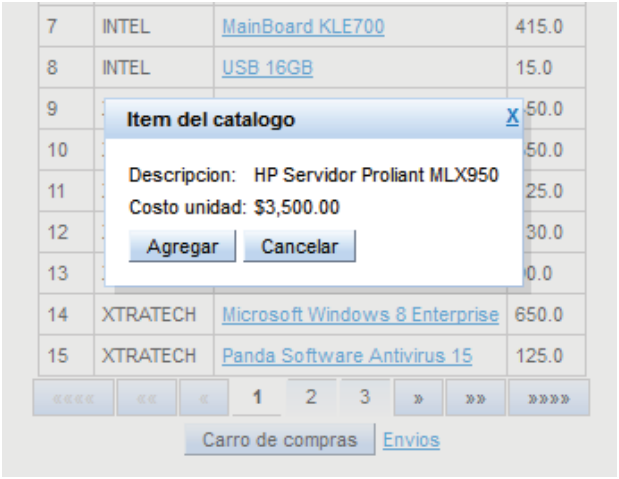


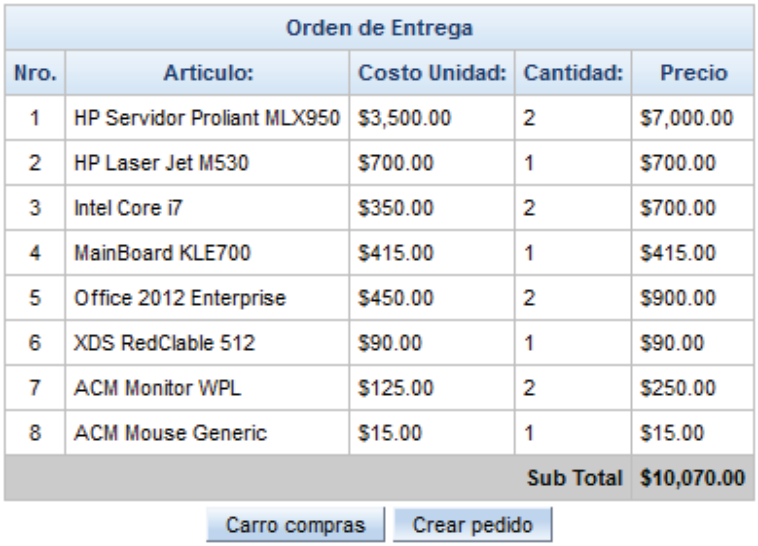
Figura 3.23 Corrección de error en el caso de prueba uno
Elaborado por: (Inga F & Sarabia P, 2013)

Caso de prueba Nº 2			
Nº caso	**Datos entrada**	**Datos obtenidos**	**Datos esperados**
2	Solicitudes de pedido.	Solicitud de compra enviada al proveedor.	Solicitud de compra enviada al proveedor.

Tabla 3.24 Caso de Prueba Nº 2

Elaborado por: (Inga F & Sarabia P, 2013)

El caso de prueba 2 resultó correcto, a continuación se presenta la pantalla con la información descrita en el caso de prueba:



Orden de Entrega

Nro.	Articulo:	Costo Unidad:	Cantidad:	Precio
1	HP Servidor Proliant MLX950	$3,500.00	2	$7,000.00
2	HP Laser Jet M530	$700.00	1	$700.00
3	Intel Core i7	$350.00	2	$700.00
4	MainBoard KLE700	$415.00	1	$415.00
5	Office 2012 Enterprise	$450.00	2	$900.00
6	XDS RedClable 512	$90.00	1	$90.00
7	ACM Monitor WPL	$125.00	2	$250.00
8	ACM Mouse Generic	$15.00	1	$15.00
			Sub Total	**$10,070.00**

Carro compras | Crear pedido

Figura 3.24 Solicitud de compra enviada al proveedor (Crear pedido)
Elaborado por: (Inga F & Sarabia P, 2013)

Caso de prueba Nº 3			
Nº caso	**Datos entrada**	**Datos obtenidos**	**Datos esperados**
3	Orden de entrega.	Detalles de una orden de entrega (artículos).	Detalles de una orden de entrega (artículos).

Tabla 3.25 Caso de Prueba Nº 3

Elaborado por: (Inga F & Sarabia P, 2013)

Este caso de prueba resultó satisfactorio, ya que indica los datos esperados como se podrá observar en las figuras 3,25 y 3.26.

Ordenes de entrega pendientes				
id Orden	Cantidad	Estado	Fecha creacion	
85	963.2	ENV	2013-06-27	Procesar orden
86	1030.4	PEN	2013-06-27	Procesar orden
87	862.4	PEN	2013-06-27	Procesar orden
88	1349.6	ENV	2013-06-27	Procesar orden
89	415.0	PEN	2013-06-27	Procesar orden
90	400.0	PEN	2013-06-27	Procesar orden
91	4330.0	PEN	2013-06-27	Procesar orden
92	3488.8	ENV	2013-06-27	Procesar orden
93	8300.0	PEN	2013-06-27	Procesar orden
94	550.0	PEN	2013-06-27	Procesar orden
95	4330.0	PEN	2013-06-27	Procesar orden
96	11278.4	PEN	2013-07-01	Procesar orden

Figura 3.25 Lista de órdenes de entrega

Elaborado por: (Inga F & Sarabia P, 2013)

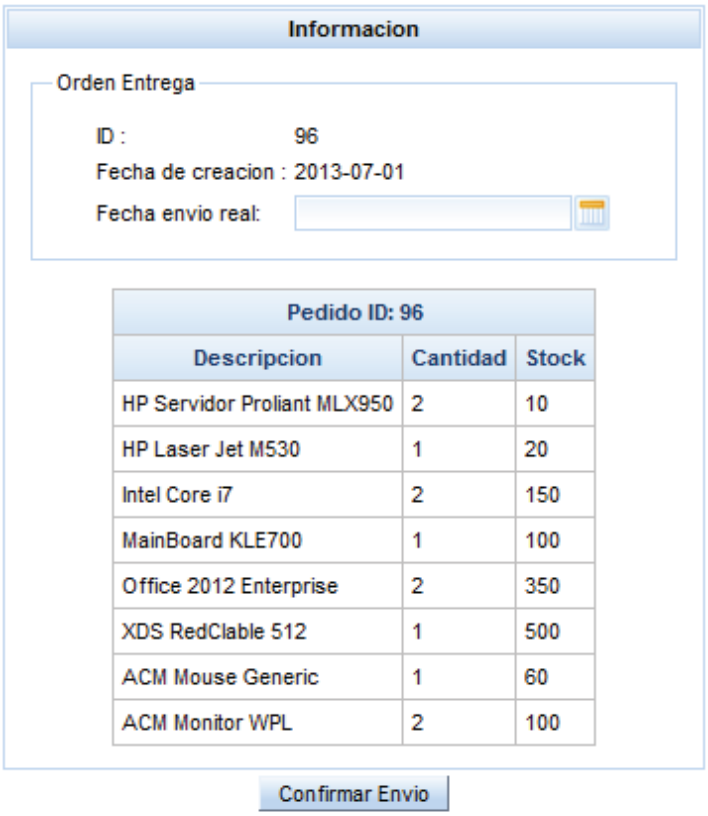


Figura 3.26 Detalle de una orden de entrega

Elaborado por: (Inga F & Sarabia P, 2013)

Caso de prueba Nº 4			
Nº caso	**Datos entrada**	**Datos obtenidos**	**Datos esperados**
4	Orden de entrega.	Error al presentar la orden de entrega cambiada de estado, esta no indica la fecha de envío real ingresada.	La orden de entrega es enviada y muestra la fecha de envío real ingresada.

Tabla 3.26 Caso de Prueba Nº 4

Elaborado por: (Inga F & Sarabia P, 2013)

En la Figura 3.28 se muestra la lista de órdenes de entrega en la que no consta la fecha de envío anteriormente ingresada.

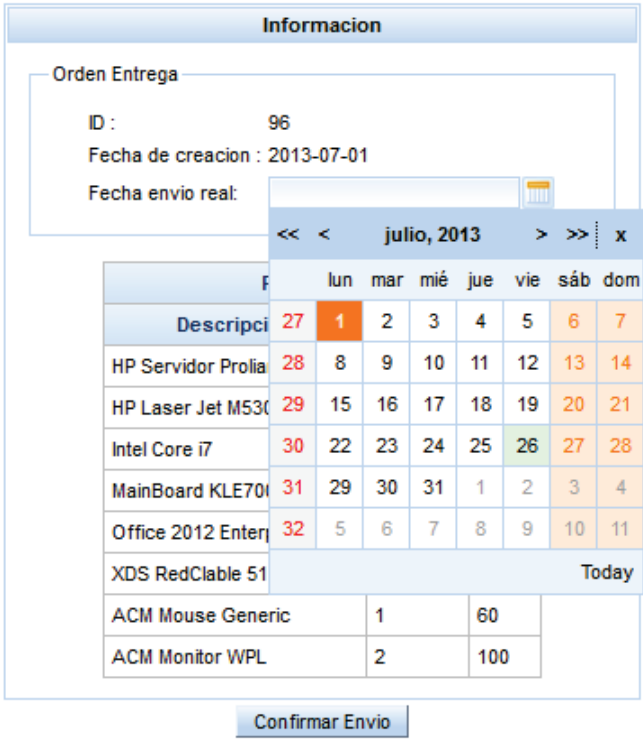


Figura 3.27 Ingreso de la fecha de envío en una orden de entrega
Elaborado por: (Inga F & Sarabia P, 2013)

Figura 3.28 Confirmación de envío de una orden de entrega
Elaborado por: (Inga F & Sarabia P, 2013)

Caso de prueba Nº 5			
Nº caso	**Datos entrada**	**Datos obtenidos**	**Datos esperados**
5	Solicitud de pedido.	El pedido es creado e indica la información de dicha acción.	El pedido es creado e indica la información de dicha acción.

Tabla 3.27 Caso de Prueba Nº 5
Elaborado por: (Inga F & Sarabia P, 2013)

En la Figura 3.29 se muestra el resultado de la prueba número cinco, a continuación se presentan las pantallas con la información descrita en el caso de prueba:

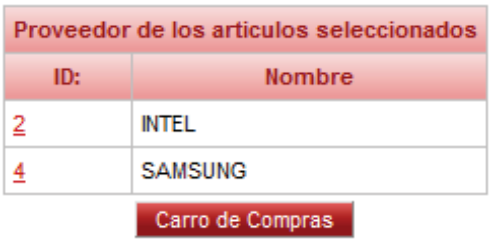


Figura 3.29 Lista de proveedores (artículos agrupados por proveedores)
Elaborado por: (Inga F & Sarabia P, 2013)

Figura 3.30 Detalle de los artículos por proveedor
Elaborado por: (Inga F & Sarabia P, 2013)

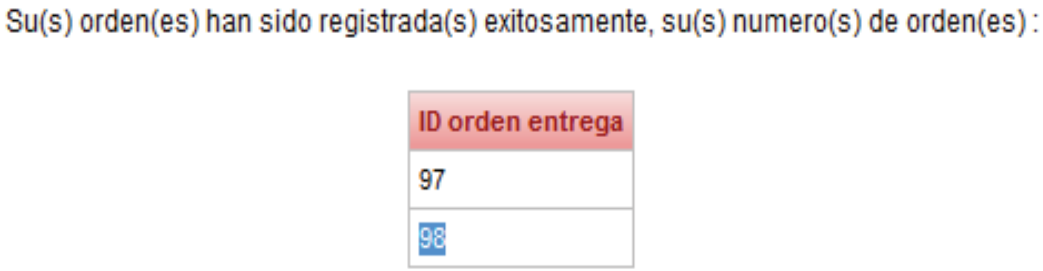


Figura 3.31 Mensaje de confirmación de pedidos creados
Elaborado por: (Inga F & Sarabia P, 2013)

Caso de prueba Nº 6			
Nº caso	**Datos entrada**	**Datos obtenidos**	**Datos esperados**
6	Pedido.	El pedido cambia de estado y los fondos de operación son comprometidos para el pago.	El pedido cambia de estado y los fondos de operación son comprometidos para el pago.

Tabla 3.3 Caso de Prueba Nº 6
Elaborado por: (Inga F & Sarabia P, 2013)

En la figura 3.32 se muestra la opción que dispone el cliente para poder ingresar a ver el estado de sus pedidos.

Figura 3.32 Opción para ingresar a ver el estado de un pedido del cliente
Elaborado por: (Inga F & Sarabia P, 2013)

En la figura 3.33 se muestra las los pedidos de un proveedor asi como el estado en el que se encuentran, recibido (REC) y enviado (ENV).

Figura 3.33 Lista de estados de pedidos del cliente

Elaborado por: (Inga F & Sarabia P, 2013)

Caso de prueba Nº 7			
Nº caso	**Datos entrada**	**Datos obtenidos**	**Datos esperados**
7	Pedido	La verificación de los datos de la tarjeta será siempre valida ya que se trata de un prototipo.	El pedido es cancelado con la tarjeta de crédito.

Tabla 3.29 Caso de Prueba Nº 7

Elaborado por: (Inga F & Sarabia P, 2013)

En la figura 3.34 se muestra el ingreso de los datos de la tarjeta de credito para proceder con el pago del pedido.

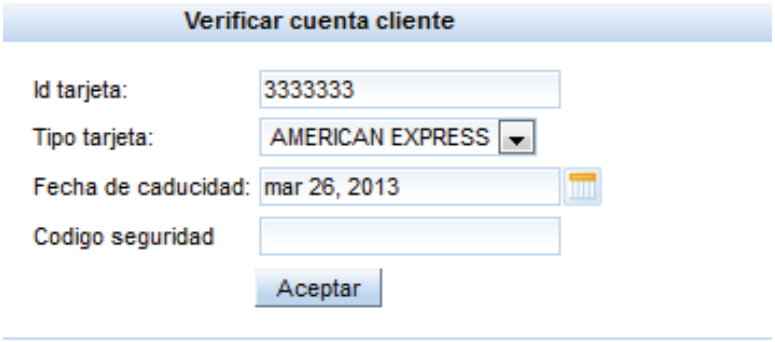


Figura 3.34 Ingreso de datos de la tarjeta de crédito

Elaborado por: (Inga F & Sarabia P, 2013)

3.3 ANÁLISIS DE RESULTADOS

El análisis de resultados expuesto a continuación no contiene un análisis detallado del sistema en sí, debido a que el principal objetivo es poner en uso la metodología planteada de MDA aplicada a LPS definida anteriormente (Capítulo 2).

Como aportación principal se ha propuesto un proceso para la construcción de sistemas software aplicando MDA en base a LPS, es decir, funcionalidades simples o complejas que son ofrecidas por un producto y que permiten satisfacer las necesidades de otro producto o familia de productos.

Este nuevo enfoque metodológico ha permitido aprovechar al máximo los beneficios de la reutilización de componentes, como se lo demostró al reutilizar los componentes del modelo de negocio B2C en el modelo de negocio B2B ya que no se los desarrollo como dos sistemas independientes sino como una familia de productos, favoreciendo positivamente a la reducción en el tiempo de desarrollo y menor uso de recursos, lo que contribuye a la reducción de costos.

Se ha definido y aplicado el conjunto de tareas necesarias planteadas para la generación de cada uno de los modelos propuestos a nivel de MDA y LPS para de esta manera, poder alcanzar el principal objetivo de este trabajo, que es difundir nuevas técnicas para el desarrollo de software a partir de modelos de negocio de alto nivel.

El método propuesto en este proyecto se centra en la reutilización de componentes independientes de plataforma, la definición de modelos de negocio de alto nivel, funcionalidades comunes y variables, que poniéndolos en ejecución como se muestra en el prototipo han facilitado el desarrollo conjunto de dos sistemas.

El método definido propone la utilización de UML para el modelado completo del sistema a nivel PIM y PSM. Esto ha permitido una realización menos compleja de los modelos, ya que es una herramienta comúnmente usada en el entorno de desarrollo, sin embargo, se han presentado dificultades en el momento de hallar herramientas de transformación que permitan generar automáticamente el código.

Realizado el prototipo, si se desea añadir un nuevo campo o atributo dentro del sistema, será necesario hacer el cambio únicamente en el Modelo Independiente de Plataforma (PIM), y a partir de éste generar nuevamente los Modelos Específicos de Plataforma (PSM) y el código de la aplicación. Si anteriormente se realizaron modificaciones al código, en esta regeneración del mismo, los cambios anteriores se mantienen.

Si se necesita agregar un nuevo método, esto se lo realiza en el PSM de EJB, se genera nuevamente el código, y se añade la funcionalidad en la clase correspondiente.

La aplicación de un plan de pruebas ha permitido constatar que al tener modelos claramente especificados, se hace más fácil la corrección de errores y cambios en los modelos y código, además de solo afectar al componente donde se detectó el error.

De esta manera, de los resultados obtenidos en base a la aplicación de la estrategia de MDA aplicada a LPS en el caso de estudio usando los modelos B2C y B2B, se puede concluir que el proceso propuesto ha cumplido con el objetivo de difundir técnicas modernas para el desarrollo de software que permiten la reducción en el tiempo de desarrollo y reducción de costos.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- MDA permite solucionar las carencias detectadas en algunos métodos para definir la lógica del negocio de manera independiente a la plataforma de implementación.

- La estrategia propuesta de MDA aplicado a LPS hizo posible una descripción con mayor precisión de cómo se elabora el producto final, permitiendo obtener modelos más visibles y de fácil validación.

- Aplicar MDA a LPS en el desarrollo de software ha permitido producir un prototipo confiable y en menor tiempo, por medio de la reutilización de componentes, lo que conlleva en la práctica la reducción de costos en el desarrollo de software

- Con la ejecución de la estrategia de MDA aplicado a LPS en el desarrollo del caso de estudio planteado, se ha podido demostrar que es factible generar de forma automática componentes reutilizables de software.

4.2 RECOMENDACIONES

- Siendo MDA un instrumento de modelación muy utilizado, se recomienda hacer investigación y comparación entre nuevas herramientas de transformación basadas en MDA, analizando sus características y su funcionalidad, para terminar generando guías de uso.

- Al ser LPS una propuesta nueva, todavía no existen sólidos fundamentos teóricos y un adecuado manejo de terminología, por lo que se recomienda nuevos trabajos de investigación en LPS permitiéndolo aplicar conjuntamente con otras tecnologías.

- Se recomienda poner en práctica la estrategia de MDA aplicada a LPS en el desarrollo de sistemas basados en software libre, dado que en el Ecuador se ha establecido como política de estado el uso de este tipo de software.

- Las mejoras futuras del proceso de desarrollo deben estar encaminadas hacia una integración total entre los entornos MDA y LPS, de tal forma que el proceso sea transparente y natural para el desarrollador.

5. BIBLIOGRAFÍA

- [1] Object Management Group (OMG).2009 http://www.omg.org/.
- [2] Giromé,F. 2009. "Diseño e Implementación de un Entorno MDA para la Producción de Aplicaciones Web" Tesina del Master de Ingeniería del Software, Métodos Formales y Sistemas de Información. Valencia-España
- [3] http://msdn.microsoft.com/es-es/library/jj135054.aspx, 2009
- [4] Parnas, D.L .1976. “On the Design and Development of Program Families”. IEEE Transactions on Software Engineering, SE-2:1-9,
- [5] SG. 2009. “Líneas de Productos de Software”. http://sg.com.mx/content/view/830
- [6] Gomaa, H. 2004. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures, Boston: Addison-Wesley.
- [7] Jacobson, I., M. Griss, and P. Jonsson. 1997. Software Reuse: Architecture, Process and Organization for Business Success. Reading, MA: Addison-Wesley.
- [8] Fowler, M. 2004. UML Distilled: Applying the Standard Object Modeling Language, 3rd ed. Boston: Addison-Wesley
- [9] Douglass, B. P. 2004. Real Time UML: Advances in the UML for Real-Time Systems, 3rd ed. Boston: Addison-Wesley.
- [10] Gomaa, H. 2000. Designing Concurrent, Distributed, and Real-Time Applications with UML. Boston: Addison-Wesley
- [11] Booch, G., J. Rumbaugh, and I. Jacobson. 2005.* The Unified Modeling Language User Guide, 2nd ed. Boston: Addison-Wesley. (*In press, to be published Fall 2004).
- [12] Carrillo,C.2006. “Arquitectura Dirigida por Moderlos (MDA) y su Aplicación en un caso de estudio” Proyecto previo a la obtención de título de Ingeniero en Sistemas Informáticos y de Computación, Quito-Ecuador.

Trabajos citados

- Protege. (04 de Agosto de 2006). Obtenido de Entorno de trabajo colaborativo: http://protege.cim3.net/cgi-bin/wiki.pl?XMIBackendTechnicalBackground#nid7HJ
- Alberto, C. R. (2009). Desarrollo de Sistema de Administración Estudiantil para el Colegio Segré usando el Desarrollo Dirigido por Modelos. Quito.
- Giromé, F. V. (02 de 05 de 2009). Francisco Valverde Giromé, "Diseño e Implementación de un Entorno MDA para la Producción de Aplicaciones. valencia, España.
- Grajales, G. (Agosto de 2009). SG Virtual Conference. Obtenido de Lineas de Producción de Software: http://sg.com.mx/content/view/830
- Guambo, C. E. (2006). Arquitectura Dirigida por Modelos y su aplicación en un caso de estudio. Quito.
- Vallecillo, A. (s.f.). Icc Uma. Recuperado el 20 de Septiembre de 2011, de http:Icc.uma.es/~av
- W3C.css. (s.f.). BdS - Blog de Superhéroes: animación, cine, cómics, series, videojuegos... . Recuperado el 23 de Septiembre de 3013, de Blog de Superheroes: http://blogdesuperheroes.es/