Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Desarrollo de Aplicaciones Web utilizando patrones de Análisis y Diseño

---

Tesis presentada para obtener el título de:

Ingeniero en Ciencias de la Computación

Presenta:

Iván Mendieta López

Asesor:

Dr. Abraham Sánchez López

Puebla, Pue. Noviembre de 2014

Agradecimientos

A Dios que me ha permitido gozar de vida y salud. Porque me ha demostrado que su presencia es real y me acompaña en cada momento de mi vida.

A mis padres y a mi hermano que me han apoyado a cada paso que doy, demostrándome que su amor es infinito e incondicional.

A mi hermosa novia Vanessa que me ha enseñado el significado de la felicidad. Estar a tu lado es una bendición.

A José Luis Méndez y Sergio Velásquez de quien he aprendido un sin número de cosas del ámbito profesional y personal.

Al Dr. Abraham Sánchez por su apoyo y confianza, porque gracias a sus enseñanzas me encuentro en estas instancias.

A mis sinodales la M.C. Consuelo Molina García y el M.C. Carlos Armando Ríos Acevedo, quienes con sus recomendaciones mejoraron la calidad de este trabajo.

A mis compañeros de MOVIS de quienes he recibido un gran apoyo.

Resumen

Las características inherentes a las Aplicaciones Web originan diferencias entre el desarrollo de software tradicional y el de carácter Web. Temas como el medio de ejecución, la tecnología para la implementación, el proceso de desarrollo y el equipo de construcción promueven estas diferencias.

El trabajo actual se encuentra encaminado a estudiar y evaluar estas peculiaridades atendiendo también el modo de resolverlas. Para cumplir el propósito se hace uso de las directivas presentes en la Ingeniería Web, las cuales indican la manera de guiar un desarrollo de esta índole.

La Ingeniería Web promueve el uso de un proceso de desarrollo ágil donde el equipo de construcción es multidisciplinario y las liberaciones son continuas. En este sentido se favorece el uso de técnicas y procesos incluidos en el marco de trabajo de Scrum.

El Análisis y Diseño orientado a objetos origina el uso de patrones y su tratamiento se ajusta perfectamente a un marco con estas características. Son los patrones los que representan el grueso del esfuerzo en el presente trabajo.

Los Patrones de Análisis se introducen como una alternativa de solución a problemas de conceptualización recurrentes. Las estructuras propuestas en estos patrones sirven de apoyo en la identificación de conceptos de dominio en un problema.

Existen Patrones para varias áreas del Diseño, pero los sujetos a estudio son los que esta dirigidos a la definición de conceptos de software de acuerdo a sus responsabilidades, roles y colaboraciones.

Índice General

Índice de figuras

Introducción

La World Wide Web fue diseñada y concebida como un medio informativo. Los primeros sitios creados por Tim Berners-Lee, mientras trabajaba en CERN (Consejo Europeo para la Investigación Nuclear por sus siglas en inglés), formaban un sistema de hipermedia que ponía al alcance de investigadores documentos publicados por sus colegas. En la actualidad y tras una continua evolución aquel sistema se ha convertido en un medio de aplicaciones, de sistemas web complejos que provee interacción, intercambio de datos y servicios personalizables, accesibles a través de diferentes dispositivos.

La web y el internet que lo soporta se han convertido en uno de los desarrollos más importantes e influyentes, no solo en el ámbito de la computación, sino en la historia del ser humano. Existen sitios y aplicaciones como google.com, wikipedia.com, amazon.com, youtube.com, facebook.com y twiter.com que han cambiado al mundo. Algunas de estas aplicaciones registran más de 500 millones de usuarios (facebook.com) y visitas diarias por arriba de los 191 millones (google.com).

A pesar de la evolución que ha experimentado la Web de un medio informativo a uno de aplicación, la situación actual del desarrollo Web evoca el estado que vivía la producción de software durante la década de los 60s, antes de que el mundo se percatara que se requería más de un experto para el desarrollo de aplicaciones. Con regularidad el desarrollo de aplicaciones Web se concibe como una actividad espontánea, basada en el conocimiento propio, la experiencia y la práctica individual, limitado al re uso en el sentido del paradigma “Copy& Paste” y a una inadecuada documentación del diseño. A pesar de que este procedimiento pudiera parecer viable, tales técnicas de desarrollo derivan con frecuencia en graves problemas asociados a la calidad, operación y mantenimiento. Las aplicaciones elaboradas bajo este esquema son fuertemente dependientes de la tecnología en la cual se desarrollan, son susceptibles a errores, tienen un bajo desempeño, no son confiables, no se pueden escalar, no son amigables con el usuario y por consiguiente tienen una nula aceptación.

En este sentido la mayoría de los desarrolladores no toman en cuenta la singularidad de los requerimientos vinculados a las aplicaciones Web y tampoco son capaces de reconocer las diferencias entre una aplicación Web y un sistema de software tradicional. Muchos desarrolladores y clientes todavía conciben el desarrollo Web como la creación de una simple página Web empleando HTML y herramientas de software como Dreamweaver, incorporando algunas imágenes e hipervínculos a documentos y a otras páginas. Desestiman requerimientos a nivel de sistema y consideraciones de diseño puntuales, tampoco hacen uso de metodologías de desarrollo y procesos. Es necesario identificar las diferencias y tomar las medidas pertinentes para satisfacer los requerimientos particulares.

Una característica determinante de las aplicaciones Web es su entorno de ejecución, sus tecnologías y estándares que actúan como plataforma de desarrollo y a su vez como plataforma de usuario. Por lo anterior una aplicación Web puede definirse como un sistema de software basado en tecnologías y estándares del W3C (World Wide Web Consortium), que suministra recursos tales como contenido y servicios a través de una interfaz de usuario.

La manera de desarrollar aplicaciones Web y el incremento en la complejidad de las mismas, particularmente aquellas orientadas a la asistencia en procesos de negocios críticos, ocasionan una creciente preocupación con respecto a este tipo de práctica y a la endeble durabilidad de las aplicaciones producidas, que por cierto ya forman la mayor parte del desarrollo de software de hoy en día.

Muchos estudios han descubierto que las áreas de mayor problema en proyectos Web a gran escala son el incumplimiento de los requerimientos del negocio (84%), retrasos en los tiempos establecidos (79%), sobrecostos (63%), falta de funcionalidad (53%), cualidad despreciable de los entregables (52%). En consecuencia, se podría hablar de una nueva crisis de software, pero de carácter Web. Debido a la omnipresencia de las aplicaciones Web y su fuerte interdependencia, esta crisis Web podría ser considerada más seria y extensa que la crisis de software de los años 60.

¿Pero cómo reducir los efectos de esta problemática?

La propuesta de solución sugiere la aplicación de una correcta Ingeniería Web y el uso de patrones de análisis y de diseño para la conceptualización del problema y su correspondiente solución.

La Ingeniería Web proporciona las herramientas necesarias para la ejecución de un adecuado análisis de requerimientos, diseño, implementación, fase de pruebas, operación y mantenimiento de aplicaciones Web.

La Ingeniería Web hace posible la planeación e iteración del proceso de desarrollo, lo que facilita la continua evolución de las aplicaciones Web. Promueve la reducción de costos y riesgos durante el desarrollo e incrementa la capacidad de evaluación de la calidad en cada fase. En este sentido el proceso de desarrollo debe ser de carácter ágil, soportar liberaciones continuas y promover la participación de un equipo de desarrollo multidisciplinario. Por lo anterior se propone el uso Scrum, que no es un proceso de desarrollo estrictamente, sino un marco de trabajo que comprende varias técnicas y procesos para el desarrollo ágil.

En el marco de un proceso de desarrollo ágil, el análisis, diseño, implementación y liberación se ejecutan continuamente durante cada iteración. Este esquema favorece el análisis y diseño orientado a objetos.

El Análisis Orientado Objetos es una actividad vinculada a la identificación de conceptos y relaciones del dominio de un problema. No tiene que ver con la solución del problema sino con la compresión del mismo.

Los patrones de Análisis apoyan en la labor de conceptualización del dominio mediante la introducción de estructuras de conceptos aplicables a escenarios diferentes.

El Diseño de Software Orientado a Objetos está dirigido a la identificación de conceptos de software y sus relaciones Inherentes.

Los Patrones de Diseño favorecen esta orientación a objetos a través de la introducción de planteamientos que describen un problema de diseño y una propuesta de solución. Los patrones aparecen en varias áreas del diseño, pero los que se abordan a detalle son los orientados a responsabilidades.

La identificación de Los patrones de diseño son planteamientos que describe por su parte también tienen un enfoque orientado a objetos, esto quiere decir que su labor está dirigida a la identificación de conceptos de software y vínculos inherentes. Los patrones sujetos a estudio son los que definen objetos en razón de sus responsabilidades, roles y dependencias (GRASP).

El Capítulo 1 explica el nacimiento y evolución del medio en donde se desenvuelven las Aplicaciones Web.

El Capítulo 2 se enfoca en la descripción de las características inherentes a las aplicaciones Web y sus categorías.

El Capítulo 3 aborda tópicos vinculados a la Ingeniería Web, en particular el del Proceso de Desarrollo de una Aplicación Web y Scrum como alternativa.

El Capítulo 4 tiene que ver con el Análisis Orientado a Objetos y el uso de Patrones de Análisis.

El Capítulo Cinco está encaminado a la definición y uso de Patrones de Diseño, en particular de los orientados a responsabilidades (GRASP).

1 Evolución de la Web

Desde su concepción y publicación el 6 de Agosto 1991 por Tim Berners-Lee, la Web ha evolucionado de manera formidable en un periodo de tiempo relativamente corto.

Es el sistema que convirtió al Internet en un medio para la divulgación de información científica, y que actualmente representa un recurso indispensable para muchas personas y organizaciones alrededor del mundo.

La evolución de la Web puede ser evaluada en diferentes dimensiones y desde diferentes perspectivas: el crecimiento en el número de páginas, la cantidad de usuarios, el nivel de funcionalidad e interactividad que ofrecen las aplicaciones, las tecnologías empleadas para su desarrollo y el impacto socioeconómico.

Es posible entender la evolución en términos de diseño Web, realizando una clasificación basada en características clave y tecnologías dedicadas a la creación de aplicaciones Web.

- Web estática
- Web dinámica
- Web 2.0
- Web móvil
- Web semántica

1.1 Web 1.0

La Web estática es un conjunto de páginas Web diseñadas con HTML que proporcionan información de productos y servicios. Con el paso del tiempo la web adquirió un carácter dinámico, con páginas creadas sobre la marcha. La habilidad para crear páginas Web a partir de contenido almacenado en bases de datos permitió a los desarrolladores Web presentar información personalizada a los visitantes, este tipo de sitios representan la Web Dinámica. A pesar de la capacidad que tienen los usuarios para obtener información en función de sus requerimientos, la comunicación se da en una sola dirección, limitando la interacción.

1.2 Web 2.0

En los últimos años ha surgido un nuevo conjunto de aplicaciones, que figuran como la Web 2.0. Esta clase de aplicaciones soportan tanto el intercambio de información como la interacción con el usuario de nuevas maneras.

Esta generación introduce interfaces de usuario inteligentes y funcionalidad que le permite al usuario generar y editar el contenido que se publica, favoreciendo principalmente la producción de contenido por parte de este último. Las aplicaciones Web creadas bajo este esquema aceptan una gestión de la información basada en las preferencias del usuario y la integración de múltiples servicios dentro de una misma interfaz.

Con el surgimiento de nuevas tecnologías como AJAX y HTLM5 la Web adquiere mayor dinamismo y un carácter altamente interactivo, donde lo usuarios pueden seleccionar contenido y también contribuir con este.

1.3 Web Móvil

Los avances hechos en el área del cómputo móvil y las comunicaciones inalámbricas, así como la proliferación de dispositivos móviles, han ocasionado un incremento en el número de usuarios que acceden a la Web a través de este tipo de dispositivos.

En poco tiempo los teléfonos móviles podrían convertirse en la plataforma dominante de acceso a internet. Los factores que pueden acelerar el cambio son la reducción en el costo de teléfonos inteligentes y computadoras de bolsillo, el auge en el uso de dispositivos cada vez más avanzados y la adecuación de aplicaciones Web para su uso en entornos móviles.

1.4 Web Semántica

Las aplicaciones Web de hoy en día presentan información en lenguaje natural, es decir, aquel que los seres humanos entienden, pero que las computadoras no pueden procesar. La Web semántica tiene como objetivo superar esa barrera, incorporando información con un significado propiamente definido, lo que permite que personas y computadoras trabajen de manera conjunta. Intenta crear un medio universal para el intercambio de información, mediante la introducción de documentos con significado procesable para las computadoras. El agregar semántica a la Web modifica su naturaleza, de un punto en el que la información se presenta vagamente a otro en el que es posible interpretarla, intercambiarla y procesarla. Asociar significado al contenido o establecer una capa de datos tratables por computadora incrementa el grado de automatización e inteligencia de las aplicaciones, y facilita la interoperabilidad de los servicios. Los tres componentes que constituyen la Web semántica son el lenguaje de marca semántico, las ontologías y los agentes de software inteligentes.

2 Aplicaciones Web

Las aplicaciones Web emplean tecnologías que agregan dinamismo al contenido y permiten la modificación del estado del servidor. La diferencia entre Sitios web y aplicaciones Web es simple, pero determinante, radica en la capacidad que posee el usuario para cambiar el estado lógico del servidor. Naturalmente que si no existe una lógica de negocios en el servidor, el sistema no puede calificar como aplicación Web. La mayoría de los sistemas web se constituyen de otros, basados en procesamiento transaccional o servicios web, bases de datos, lo cual genera un aumento en la complejidad del diseño, desarrollo, liberación y mantenimiento de las aplicaciones.

2.1 Aspectos específicos de las aplicaciones Web

Las aplicaciones Web poseen características únicas que hacen del desarrollo Web no solo una tarea diferente sino más desafiante en comparación con el desarrollo de Software tradicional. El entorno de ejecución de las aplicaciones Web y la velocidad con que estas se desarrollan y entregan son factores que determinan la diferencia. Otro punto a destacar es la Seguridad, debido a quelas aplicaciones de este tipo son más susceptibles a ataques que las aplicaciones tradicionales. Las siguientes son algunas características de las aplicaciones Web:

- La mayoría de las aplicaciones Web evolucionan por naturaleza, lo que requiere constantes modificaciones en su contenido, funcionalidad, estructura, navegación, presentación e implementación. La evolución se da en términos de los requerimientos y funcionalidad, mismos que pueden surgir después de entrar en operación. En la mayoría de las aplicaciones Web el grado y la frecuencia de cambio son más altos que en las aplicaciones de Software tradicional, en muchos casos no es posible especificar de principio todos sus requerimientos. Se entiende entonces que para administrar exitosamente la evolución y los nuevos requerimientos de las aplicaciones Web se requiere de una mayor técnica, organización y administración.
- Las aplicaciones Web se conciben para el consumo de una comunidad extraordinaria de usuarios con diferentes requerimientos, expectativas y habilidades. Por esta razón la interfaz de usuario y las características de funcionamiento deben satisfacer una colectividad anónima diversa. A lo anterior se suma el número de usuarios que ingresan a las aplicaciones de forma concurrente, lo que es inevitable y puede variar considerablemente, originando problemas de desempeño.
- Los sistemas Web dependen de la presentación de contenido variado, texto, gráficos, imágenes, audio, y video. De tal manera que su desarrollo comprende la creación, administración y presentación de contenido de forma atractiva, así como el continuo abastecimiento del mismo después del desarrollo inicial y la implementación.
- En general los sistemas web reclaman una buena presentación y una navegación sencilla.
- Las aplicaciones Web, especialmente aquellas dirigidas a una audiencia global, deben considerar aspectos sociales y culturales, lo cual implica contemplar múltiples idiomas y diferentes estándares.
- En términos de seguridad y privacidad los sistemas Web son en general más exigentes que los de software tradicional, por lo que existe una gran demanda en este rubro.

- Las aplicaciones Web necesitan ser compatibles con un gran número de dispositivos y formatos, soportar hardware y software variado, y funcionar en redes con una amplia gama de velocidades.
- Las consecuencias originadas de los fallos y la insatisfacción producida en los usuarios por las aplicaciones Web son más graves que en los sistemas convencionales. Por otro parte las aplicaciones Web pueden fallar por muchas y diversas razones.
- Los tiempos de desarrollo para las aplicaciones Web son más cortos, lo que influye significativamente en el diseño, la metodología de desarrollo y el proceso.
- La difusión de nuevos recursos para el desarrollo Web, los estándares y la constante presión de la competencia para utilizarlos atrae beneficios, pero también retos adicionales.
- La naturaleza evolutiva de las aplicaciones Web demanda un proceso de desarrollo incremental y progresivo.

2.2 Categorías de las aplicaciones Web

Las aplicaciones Web tienen distintos grados de complejidad. Pueden ser de carácter informativo o enfocarse a la ejecución de tareas complejas. Es necesario entender la relación que existe la cronología del desarrollo y la complejidad, lo que implica una concepción de desarrollo superior.

El producto de un desarrollo Web puede clasificarse en cualquiera de las categorías descritas en la Figura 2.1, pero puede crecer en complejidad. Cada una de las categorías tiene su propio campo de aplicación. Las categorías más recientes son en general más complejas, pero no significa que sustituyan por completo a las categorías anteriores. En consecuencia las aplicaciones Web complejas pueden estar comprendidas en más de una categoría a la vez.

Figura 2.1 Categorías de Aplicaciones Web

Los **Sitios Web basados en documentos** son los precursores de las aplicaciones Web. Las páginas Web se elaboran y almacenan en el servidor Web, listas para ser invocadas por el cliente. Las páginas de esta clase son actualizadas de forma manual, lo que atrae serios costos cuando se trata de sitios que sugieren constantes cambios o un gran número de páginas. La ventaja principal es la sencillez y estabilidad con la que operan. Ejemplos de esta categoría son páginas de pequeños negocios.

Con la introducción de CGI (Common Gateway Interface) y los formularios HTML, surgen las **aplicaciones Web interactivas**. Las páginas Web y vínculos a otras páginas son generados dinámicamente de acuerdo a lo que introduce el usuario. Presentaciones virtuales y sitios de noticias son un ejemplo.

Las **aplicaciones Web transaccionales** no solo proveen interactividad, también hacen posible la realización de cambios sobre el contenido subyacente. Para su ejecución es necesario contar con un sistema de base de datos que facilite la gestión y consulta de la creciente información vinculada a la aplicación Web. Ejemplos de este tipo de aplicaciones son la banca, el comercio en línea y los sistemas de reservaciones.

Las aplicaciones **Web basadas en flujo de trabajo** facilitan la gestión del tráfico entre diferentes compañías, instancias públicas y usuarios privados. Una necesidad inherente es el suministro de servicios Web que garanticen la interoperabilidad. Los retos principales son el manejo de la

complejidad de los servicios en cuestión, la autonomía de las compañías y la necesidad de un flujo de trabajo robusto y flexible. Las soluciones Business to Business del comercio electrónico y la administración pública son ejemplo de este clase de aplicaciones.

Se utilizan **aplicaciones Web Colaborativas** cuando el nivel de cooperación para la gestión del contenido es alto y no se requiere de una estructura en los procesos y operaciones. En este tipo de entorno la comunicación entre los usuarios es muy importante, también lo es el soporte para la información compartida y los espacios de trabajo en común. Su implementación contempla actividades como la administración de entradas y actualizaciones (Weblogs), la gestión de reuniones o la toma de decisiones (plataformas e-learning).

A pesar del carácter anónimo de la Web hay una tendencia creciente que promueve una Web social, en la que las personas hacen publica su identidad en una comunidad con la que comparte intereses.

Las **Aplicaciones Web tipo portal** representan un punto de acceso a diferentes fuentes de información y servicios. Existen aplicaciones creadas por grandes compañías como Microsoft y Yahoo que bajo este esquema proveen acceso a la Web. Otros portales se especializan en cierto ámbito, es el caso de los portales de negocios y de compras en línea. Los portales de negocios brindan acceso a información de interés para empleados y asociados de forma pública o través de una intranet. Los portales de compras en línea se dividen horizontal y verticalmente, los horizontales tienen un formato negocio-consumidor, en donde los productos se ofrecen al público en general, los verticales por otro lado tienen un formato negocio-negocio en el que las operaciones se realizan con compañías de otros sectores.

Las **aplicaciones Web ubicuas** suministran servicios personalizados, accesibles desde cualquier lugar a cualquier hora. Un ejemplo puede ser la presentación del menú del día en los dispositivos móviles de los clientes que ingresan a un restaurante de las 11 am a las 2 pm. El grado actual de ubicuidad en aplicaciones de esta categoría aun es limitado debido a que solo atacan un aspecto a la vez, es decir, que soportan solo la personalización, la localización o la operación multiplataforma.

2.3 Características de las aplicaciones Web

Las aplicaciones de carácter Web difieren de otras en varios aspectos importantes. Hay características relacionadas a las aplicaciones Web (navegación no lineal) que no se presentan en aplicaciones tradicionales, de hacerlo estas tienen diferentes repercusiones (frecuencia de actualización). Por esta razón es necesario que los conceptos, métodos, técnicas y herramientas de la ingeniería de software tradicional se adapten para su uso en la Ingeniería Web.

Figura 2.2 Dimensiones de acuerdo a ISO/IEC 9126-1 para la clasificación de las aplicaciones

La Figura 2.2 proporciona una visión general de las características y las organiza a lo largo de tres dimensiones producto, uso y desarrollo, e incorpora una cuarta asociada a la evolución que comprende a las tres primeras. La clasificación de las características a través de las diferentes dimensiones permite determinar el grado de influencia que tienen en la calidad de las aplicaciones, y se pueden utilizar como punto de partida para la definición de los requerimientos en la ingeniería Web. A continuación se describen las características de acuerdo a estas dimensiones.

2.3.1 Características orientadas al producto

Este grupo de características constituyen los principales bloques de desarrollo de las aplicaciones Web, que se componen del contenido, estructura de hipertexto (navegación) y presentación (interfaz de usuario). De acuerdo con el paradigma orientado objetos, cada uno de estos elementos se compone de una estructura y un comportamiento.

2.3.1.1 Contenido

Actividades como la disposición, integración y actualización del contenido figura como una de las tareas más importantes del desarrollo. El uso de las aplicaciones Web se encuentra directamente asociado al contenido que en ellas se publica, situación que obliga a los desarrolladores a actuar no solo como programadores sino también creativos. Los aspectos a considerar son la constante variación en la estructura del contenido y la demanda de calidad por parte de los usuarios.

- Enfoque en el contenido y multimedia. La administración del contenido depende de la estructura y puede presentarse en forma de tablas, texto, gráficos, animaciones, audio o video. En aplicaciones Web de este carácter la mayor parte de la información se suministra y tiene como objetivo un grupo de usuarios en específico. El resto de la información se genera y actualiza de manera dinámica.
- Demanda de calidad. El tipo de contenido de una aplicación Web se da en función del área de servicio, se sujeta a diferentes frecuencias de actualización y a diversas métricas de calidad, tales comprenden el grado de actualización, exactitud, consistencia y confiabilidad. Es necesario considerar no solo las exigencias de calidad durante la definición de requerimientos, sino también el grado en que estas se satisfacen.

Los nuevos sitios registran un gran número de actualizaciones y de solicitudes de contenido. A diferencia de otros medios como la televisión, la radio, la prensa, la Web posee una gran capacidad para resolver esta clase de exigencias mediante la personalización.

2.3.1.2 Hipertexto

Una de las características más significativas de las aplicaciones Web es la naturaleza no lineal de sus documentos de hipertexto. Existen diferentes modelos de hipertexto, pero el que la Web define es muy simple y comprende 3 elementos nodos, vínculos y anclas.

El carácter no lineal del paradigma de hipertexto se traduce como su propiedad fundamental.

No linealidad: Los hipertextos contemplan estereotipos de lectura sistemática parcial, lo que establece una diferencia fundamental entre las aplicaciones Web y los sistemas de software tradicional. Es posible observar esta característica en las distintas formas de navegación de un sistema de compras en línea. Este modo de lectura, adaptable a las necesidades y el comportamiento es ideal para el aprendizaje humano. Las anclas y vínculos que permiten la conducción se generan de manera previa o dinámicamente de acuerdo a las preferencias y el comportamiento del usuario.

Desorientación y sobrecarga cognitiva. El tratamiento correcto de estos dos conceptos es fundamental en el desarrollo de aplicaciones Web. La sobrecarga cognitiva surge a partir de la concentración adicional de esfuerzo para retener en memoria los diversos caminos y tareas. El mapa del sitio, la búsqueda basada en palabras clave y la exhibición del tiempo transcurrido desde el ingreso al sitio son excelentes herramientas de apoyo para la orientación del usuario.

2.3.1.3 Presentación

En relación a la presentación dos son las peculiaridades que distinguen a las aplicaciones Web, son la estética y la fácil navegación.

Estética: En las aplicaciones de software tradicional el aspecto visual de la interfaz de usuario no representa un factor central como en el caso de las aplicaciones Web, una razón es el entorno en que operan estas últimas, que se caracteriza por una alta competencia. La apariencia de las páginas Web está sujeta a tendencias de moda que con frecuencia determinan el éxito o fracaso de una aplicación.

Navegación sencilla: Además del aspecto, es esencial que la navegación y el comportamiento dentro de la aplicación sean consistentes, de tal manera que los usuarios se familiaricen fácilmente con su uso. La meta es conseguir que los usuarios aprendan a usar la aplicación sin necesidad de emplear alguna documentación.

2.3.2 Características de uso

El uso de las aplicaciones Web es más heterogéneo en comparación con las aplicaciones tradicionales. El número y la diversidad de usuarios, el hardware y el software de los dispositivos, el tiempo y la ubicación desde donde se accede a la aplicación establecen la diferencia.

Por otro lado las aplicaciones Web se caracterizan por una continua adaptación a las diferentes condiciones de uso o contextos. Es necesario que la transición se haga considerando los diferentes elementos de una aplicación, el contenido, el hipertexto y la presentación. Es como resultado de la adaptación que las características de uso se dividen en tres grupos: contexto social, contexto técnico y contexto natural.

2.3.2.1 Contexto Social

El contexto social atiende aspectos específicos de los usuarios. La frecuencia y la diversidad cultural por ejemplo, incrementan el grado de heterogeneidad.

Frecuencia: Los usuarios pueden emplear una aplicación un determinado número de veces y después dejarlo de hacer, probablemente por la competencia. Los motores de búsqueda facilitan el acceso a aplicaciones competitivas, promoviendo que los usuarios ingresen exclusivamente a aquellas que les parecen útiles y atractivas.

Diversidad cultural: Las aplicaciones Web están orientadas a diferentes grupos de usuarios. Cuando se trata de un grupo de usuarios conocido, como en el caso de una intranet es posible trabajar bajo el esquema de una aplicación tradicional. Si se desarrolla una aplicación Web para un grupo de usuarios anónimo se incrementa la heterogeneidad en términos de sus habilidades, conocimientos y preferencias. Para lograr una correcta personalización del producto es indispensable hacer consideraciones sobre los contextos de usuario durante la fase de desarrollo de la aplicación. El gran número de usuarios y grupos hace difícil definir un conjunto representativo para un análisis de requerimientos.

2.3.2.2 Contexto Técnico

El contexto técnico comprende las características de hardware y software de los dispositivos utilizados para acceder a aplicaciones multiplataforma y la conexión de red considerando la calidad de Servicio.

- Funcionalidad multiplataforma. Las aplicaciones Web ofrecen servicios para diferentes tipos de dispositivos, pero principalmente y de manera creciente para aquellos de carácter móvil. La enorme variedad de navegadores presentan un reto, debido a sus diferentes funcionalidades y restricciones. Son estos factores los que acomplejan el desarrollo de interfaces estables. De manera adicional la configuración personalizada de los

navegadores, en su presentación y control de acceso afecta el desempeño, la transaccionalidad y la interacción por nombrar algunas.

- Calidad de servicio. Las aplicaciones web se basan en el principio cliente/servidor. Las características del medio de transferencia, el ancho de banda y la inestabilidad de la conexión son factores que deben tomarse en cuenta para garantizar la calidad de Servicio en una aplicación de índole Web.

2.3.2.3 Contexto natural

El contexto natural incluye aspectos de tiempo y ubicación. El carácter global y la disponibilidad incrementan el grado de heterogeneidad.

- Carácter global. La ubicación desde la cual se ingresa a una aplicación Web es importante para su internacionalización, esto en razón de las diferencias culturales y lingüísticas.
- Disponibilidad. La naturaleza de acceso inmediato que las aplicaciones Web heredan del Internet las hacen disponibles instantáneamente. Pueden ser utilizadas en el momento que así lo desee el usuario, por lo que su calidad debe ser asegurada.

2.3.3 Características inherentes al desarrollo

El desarrollo de aplicaciones Web se caracteriza por el empleo de recursos esenciales, tales como el equipo de desarrollo, la infraestructura técnica, el proceso de desarrollo y la integración de soluciones pre existentes.

2.3.3.1 El equipo de desarrollo

Un factor relevante en la construcción de aplicaciones Web es el equipo de desarrollo, el cual se caracteriza por la juventud de sus integrantes, mismos que emanan de diferentes disciplinas. Estas peculiaridades contribuyen a la introducción de una nueva forma de colaboración.

Se debe entender que para el desarrollo de aplicaciones Web es necesario contar con grupo multidisciplinario de personas que contribuyen con habilidades y conocimientos propios de sus diferentes áreas. Es necesario contar con gente experta en TI que pueda ocuparse de la implementación del sistema, gente con conocimientos en lenguaje de hipertexto y diseño para el manejo de la presentación y otro tipo de profesionales que se puedan participar en la gestión del contenido.

2.3.3.2 Infraestructura

Existen circunstanciasen las que se ve afectado tanto el desarrollo como el uso de las aplicaciones Web.

La utilización de componentes afecta la construcción debido a que algunos de estos fueron liberados de forma prematura acarreando bugs y produciendo comportamientos no deseados. Con respecto al uso se debe destacar que siempre existen dos componentes externos, el servidor y el navegador. Los desarrolladores pueden configurar el servidor de aplicaciones de una forma específica, pero no pueden manipular el tipo de navegador que emplea el usuario ni la forma en que este lo configura.

2.3.3.3 Proceso

La construcción de aplicaciones web satisface características inherentes al desarrollo de software, de manera particular se vincula con el paralelismo y la flexibilidad.

La flexibilidad trasciende en razón de que resulta imposible incorporar una planeación rígida durante el desarrollo de aplicaciones de esta índole. Es vital dar soporte a cambios.

El concepto de paralelismo surge a partir de factores como el tiempo reducido con el que se cuenta para el desarrollo. Es por tal que la labor suele distribuirse a través de varios equipos que trabajan de forma paralela en la construcción de distintos componentes. Contrario a lo establecido en el desarrollo de software tradicional los grupos se dividen en razón de los componentes elaborados y no en razón de su experiencia profesional.

Adicionalmente a la construcción paralela de las diferentes piezas del producto se encuentran tareas como el diseño y la implementación del aseguramiento de la calidad que son ejecutados de forma simultánea a lo largo de las diferentes versiones. El avance en paralelo de las diferentes fases del desarrollo introduce nuevos requerimientos para la planeación y liberación de un producto web.

2.3.3.4 Integración

Un requerimiento común de muchas aplicaciones Web es la interacción interna y externa. A este respecto no sólo se toman en cuenta aspectos técnicos sino también de contenido y organizacionales.

Integración interna. Con frecuencias las aplicaciones Web deben integrarse con sistemas legados cuando existe contenido, por ejemplo ofrecer catálogos de productos a través de la aplicación.

Integración externa. Las aplicaciones Web participan de contenido y servicios de otras aplicaciones repetidamente. A pesar de la similitud con la que operan los sistemas de base de datos, existe un gran número de ellos que cambian continuamente en materia de disponibilidad y esquema.

La incorporación de servicios externos a las aplicaciones consiste en el consumo y suministro de servicios Web. Un servicio web se define como un componente con una interfaz y funcionalidad

2.3.3.5 Evolución

El carácter evolutivo de las aplicaciones Web es consecuencia del cambio continuo de circunstancias y requerimientos, de la presión que ejerce la competencia y de la incesante marcha del desarrollo.

Variabilidad Continua. El cambio rápido y permanente de las tecnologías y estándares empleados para el desarrollo web motiva una permanente adecuación. Las razones son simples, el usuario desea utilizar la tecnología más reciente para lo cual es necesario disponer de herramientas tecnológicamente apropiadas. La variabilidad se encuentra directamente asociada a las tres dimensiones de una aplicación Web, el producto, el uso y en particular el desarrollo.

Presión de la competencia. La enorme competencia en la Web, la presión del mercado y la necesidad por establecer presencia en la Web ocasiona la introducción de ciclos de vida del

producto más cortos y ciclos de desarrollo aún más breves, lo que aparentemente no da cabida al empleo de procesos de desarrollo sistemáticos.

Curso Acelerado. La presión excesiva en relación al tiempo de desarrollo de una aplicación Web se debe al vertiginoso cambio en la Web, al corto tiempo de vida de las aplicaciones y a las frecuentes actualizaciones.

A diferencia del software convencional donde la evolución se da en términos de versiones planeadas plenamente, las aplicaciones Web se caracterizan por liberaciones continuas. Esto quiere decir que las aplicaciones de carácter Web se encuentran en permanente mantenimiento. Por lo anterior es necesario que para su elaboración se utilicen versiones modificadas de procesos de Ingeniería de Software tradicional.

3 Ingeniería Web

No es viable emplear metodologías de propósito específico para la producción de aplicaciones Web, toda vez que estás experimentan un gran auge y realizan tareas cada vez más críticas. La complejidad surgida de las interrelaciones entre las aplicaciones Web representa un reto importante para nuestra habilidad de comprensión, desarrollo, y gestión. Aunado a esto, el alto costo generado por el fracaso de una aplicación Web ha motivado la demanda de recursos que puedan mejorar el diseño, la calidad y la confiabilidad de un aplicativo.

La ausencia de un planteamiento para el desarrollo de aplicaciones Web puede derivar en un producto que no satisface la funcionalidad requerida, no tiene el desempeño deseado o carece de calidad. El proceso de desarrollo tiende a complicarse, se dificulta la gestión y los tiempos se desfasan gravemente.

La ingeniería Web busca resolver la problemática vinculada al desarrollo de sistemas Web mediante el establecimiento de una base para la creación sistemática de aplicaciones Web, la cual se constituye de un colectivo de conocimientos teóricos y empíricos para el desarrollo, liberación y soporte de la continua evolución de las aplicaciones Web.

La ingeniería Web es la aplicación de conceptos, métodos, técnicas y herramientas para la correcta ejecución del análisis, diseño, implementación, pruebas, operación y mantenimiento de aplicaciones Web de alta calidad. Representa un proceso sistemático para la administración de la complejidad y la heterogeneidad de las aplicaciones. Se interesa por el desarrollo y la organización de conocimientos vinculados a las aplicaciones Web, así como también de la forma en que se aplican en la resolución de nuevos requerimientos.

Contrario a la percepción de algunos profesionales, la Ingeniería Web no es una copia de la Ingeniería de Software, sin embargo ambas comprenden disciplinas como la programación y el desarrollo de software. La Ingeniería Web utiliza principios de Ingeniería, comprende nuevos planteamientos, metodologías, herramientas, técnicas y guías para satisfacer los requerimientos específicos de los sistemas Web.

La naturaleza y características inherentes a las aplicaciones Web demandan el empleo de una Ingeniería Web multidisciplinaria, que incluya aportaciones de diversas áreas como el análisis y diseño de sistemas, ingeniería de software, ingeniería de hipermedia/hipertexto, ingeniería de requerimientos, interacción humano computadora, interfaces de usuario, ingeniería de la información, indexación y recuperación de información, pruebas, modelado y simulación, gestión de proyectos, diseño gráfico y presentación.

Un sistema Web al que se le ha aplicado una correcta ingeniería

- es completamente funcional y correcto
- es usable
- es robusto y confiable
- es posible darle mantenimiento
- es seguro
- funciona de forma aceptable bajo condiciones de sobrecarga

- es escalable
- es portable, cuando se requiere, es compatible con múltiples navegadores
- es reusable
- es interoperable con otros usuarios
- esta correctamente documentado

3.1 Proceso de Desarrollo de una Aplicación Web

Un proceso de desarrollo de Software es una metodología para la construcción, liberación y tentativamente el mantenimiento de Software. Permite estimar recursos y tiempos además de monitorear el avance del proyecto.

El ritmo de crecimiento de las aplicaciones Web hace inminente su constitución como sistemas de software complejos. Por esta razón es importante la concepción de un proceso de desarrollo estructurado que soporte las características relativas a este tipo de aplicaciones.

El desarrollo de Sistemas Web origina requerimientos específicos que representan un factor determinante para la ejecución un proceso de desarrollo.

3.1.1 Administración de Ciclos Cortos de Desarrollo

Numerosos estudios demuestran que el tiempo de desarrollo de las aplicaciones Web es sumamente corto, con frecuencia no supera los seis meses y tiene un promedio de duración menor a tres meses. La administración de ciclos cortos de desarrollo en las aplicaciones Web figura como primer requerimiento para el proceso de desarrollo.

Un elemento que motiva la tendencia de ciclos cortos es la frecuente competencia en la Web. La Web actúa como un medio de intercambio comercial de grandes dimensiones. El mecanismo de liberación inmediata que surge como cualidad natural del entorno Web favorece la rápida publicación de productos, pero también refuerza la necesidad de mantenerse siempre a la vanguardia. Los ciclos cortos de desarrollo limitan el uso de un proceso de desarrollo sistemático.

3.1.2 Administración de Requerimientos Variables

Muchos de los requerimientos de las aplicaciones Web aparecen durante el desarrollo y a menudo cambian con respecto a su contenido y la forma en que se implementan. Es común que los desarrolladores se enfrenten no sólo a conceptos de negocio desconocidos sino también a requerimientos que cambian dramáticamente. Aquellos requerimientos que se originan a partir de un cambio en la situación del mercado deben satisfacerse rápidamente para evitar que la competencia se coloque un paso adelante.

A partir de lo expuesto se entiende que es importante reducir el tiempo de desarrollo, lo que hace imposible la tarea de definir requerimientos de manera precisa. No es factible asegurar que los requerimientos permanecerán estables, por lo que es innecesario invertir tiempo en su especificación cuando no existe un beneficio tangible. Una forma de evaluar requerimientos es mediante la retroalimentación del usuario final, actividad que se lleva a cabo a través de la liberación continua de versiones.

La tarea de definir requerimientos en proyectos Web es con frecuencia una actividad compleja para la mayoría de los clientes debido a que no se encuentran familiarizados con el dominio del problema ni con sus posibles soluciones. Por esta razón muchas veces no se perciben los beneficios de construir cierta funcionalidad o exponer parte del contenido. Lo anterior ocasiona que los datos relativos a la aplicación se actualicen continuamente, situación que origina cambios frecuentes en la lógica funcional de los requerimientos.

El ritmo de cambio en el desarrollo también se ve afectado por la constante actualización en las tecnologías y estándares. La lucha por emplear los recursos más actuales también contribuye a la intensa competencia.

Con el tiempo algunos de los elementos que colaboran con el patrón de cambio podrían disminuir su nivel de injerencia o incluso desaparecer. Pero existen otros que no dejaran de ser factor, tales son la necesidad de experimentar con software nuevo, los efectos de la evolución en los requerimientos y la presión de la competencia.

La condicionante de cambio afecta el proceso de desarrollo, ya que promueve una fuerte integración entre el equipo de desarrollo. Como resultado del carácter emergente de los requerimientos y de su peculiar inestabilidad, es necesario que se el cliente este bien informado del estado del producto.

3.1.3 Liberaciones de Contenido Flexible en Plazos Definidos

De manera previa se introdujo el concepto de liberación continua como alternativa para la validación de requerimientos. Esta actividad se enfoca en la publicación de una clase especial de prototipos durante la implementación del proceso de desarrollo. A partir de una descripción elemental que hace el usuario, se construye rápidamente un prototipo con fines ilustrativos. Este tipo de práctica fomenta la comunicación con el usuario, lo que a su vez permite identificar aspectos importantes de los requerimientos.

El tiempo que transcurre entre cada liberación es significativamente corto. Trasciende más la planificación de los intervalos de entrega que la propia especificación de los requerimientos. Si una funcionalidad no se puede incluir dentro de cierta liberación, esta se analiza y de ser prioritaria se contempla para la siguiente.

Un desarrollo orientado a liberaciones solventa la necesidad de un mecanismo para la estimación de costos y tiempo.

3.1.4 Desarrollo Paralelo

El desarrollo paralelo es una práctica recurrente durante la producción de aplicaciones Web. Varios equipos de desarrollo pueden laborar en la construcción de diferentes funcionalidades, que contribuyen a una liberación o el producto final. Esto quiere decir que actividades asociadas al diseño, implementación y aseguramiento de la calidad se ejecutan paralelamente a través del desarrollo de distintas funcionalidades. Un resultado inmediato es la necesidad de una cuidadosa asignación de recursos.

Se entiende que la comunicación es una constante en la construcción de aplicaciones Web. Por lo que es necesario que el proceso de desarrollo comprensa mecanismos que promuevan la comunicación, comenzando por ejemplo con la restricción en el volumen de los equipos de desarrollo.

3.1.5 Integración y Reutilización

Un hábito común durante el desarrollo de aplicaciones Web es la reutilización de componentes, que se ve motivado por la enorme presión que se genera con respecto al tiempo. Con frecuencia los componentes que se integran son elaborados por otras áreas de la organización o por un tercero.

Cuando se construye un componente reusable dentro de un proyecto, es necesario trabajar de manera conjunta con los proyectos que lo consumirán. Si por ejemplo, se desarrolla una arquitectura para la producción de varias aplicaciones, se deben coordinar tanto las expectativas de resultado como los recursos para alcanzar el objetivo. Por supuesto que las dependencias entre proyectos Web incrementan el riesgo de propagación de errores, lo que debe ser mitigado por el proceso.

Se sugiere que la planeación de dependencias entre proyectos se genere durante el modelado, desafortunadamente no existe un estrategia definida para expresar la reutilización de conceptos en aplicaciones Web.

3.1.6 Adecuación al Nivel de Complejidad

De acuerdo a lo expuesto de manera previa se sabe que durante el desarrollo de aplicaciones Web se favorece el empleo de ciclos cortos con liberaciones continuas en lugar de la especificación detallada de los requerimientos. Las funcionalidades que adquieren mayor prioridad son aquellas que agregan valor directo al entregable. Los requerimientos que no se atienden en los primeros ciclos se canalizan para su atención en los próximos.

Este ejercicio es útil mientras la complejidad del aplicativo no se incremente. A medida que la aplicación se va integrando al proceso de negocio, el desarrollo se hace más complejo. Por esta razón es importante utilizar un proceso de desarrollo que administre este tipo de variaciones.

3.2 Análisis de Scrum

Scrum se ha empleado para la construcción de productos complejos desde principios de la década de los 90's. Es importante señalar que Scrum no es propiamente un proceso, sino un marco de trabajo que comprende varios procesos y técnicas. Su objetivo es potenciar la eficacia de las prácticas de desarrollo.

Scrum maneja un enfoque iterativo e incremental que optimiza la predictibilidad y el control de riesgos. Satisface tres requerimientos fundamentales para un control de proceso exitoso.

La **transparencia** promueve que las entidades a cargo de la gestión de los resultados conozcan los elementos que afectan los entregables. Todos los involucrados en la construcción deben estar de acuerdo en el significado de terminado.

La **inspección** frecuente del proceso garantiza la identificación temprana de resultados con variaciones inaceptables. El éxito de la tarea radica en la habilidad y la pericia para desarrollarla.

Si durante la inspección se identifican varias actividades con resultados por debajo de lo aceptable, el responsable debe **adaptar** el proceso o el material procesado. El ajuste debe hacerse rápidamente para evitar futuras complicaciones.

Tres son las prácticas que apoyan la inspección y la adaptación. El Daily Scrum meeting que permite evaluar el estatus del Sprint Goal y realizar las adecuaciones para incrementar el valor de la siguiente jornada de trabajo. Los meetings del Sprint Review y Sprint Planning que se hacen con respecto al próximo Release Goal y tienen como objetivo optimizar el valor del próximo Sprint. Por último se encuentra el Sprint Retrospective que permite determinar las adecuaciones necesarias para hacer más productivo y completo al próximo Sprint.

3.3 El Contenido de Scrum

Scrum comprende un conjunto de Scrum Teams que interactúan con diferentes Eventos, Artefactos y Reglas.

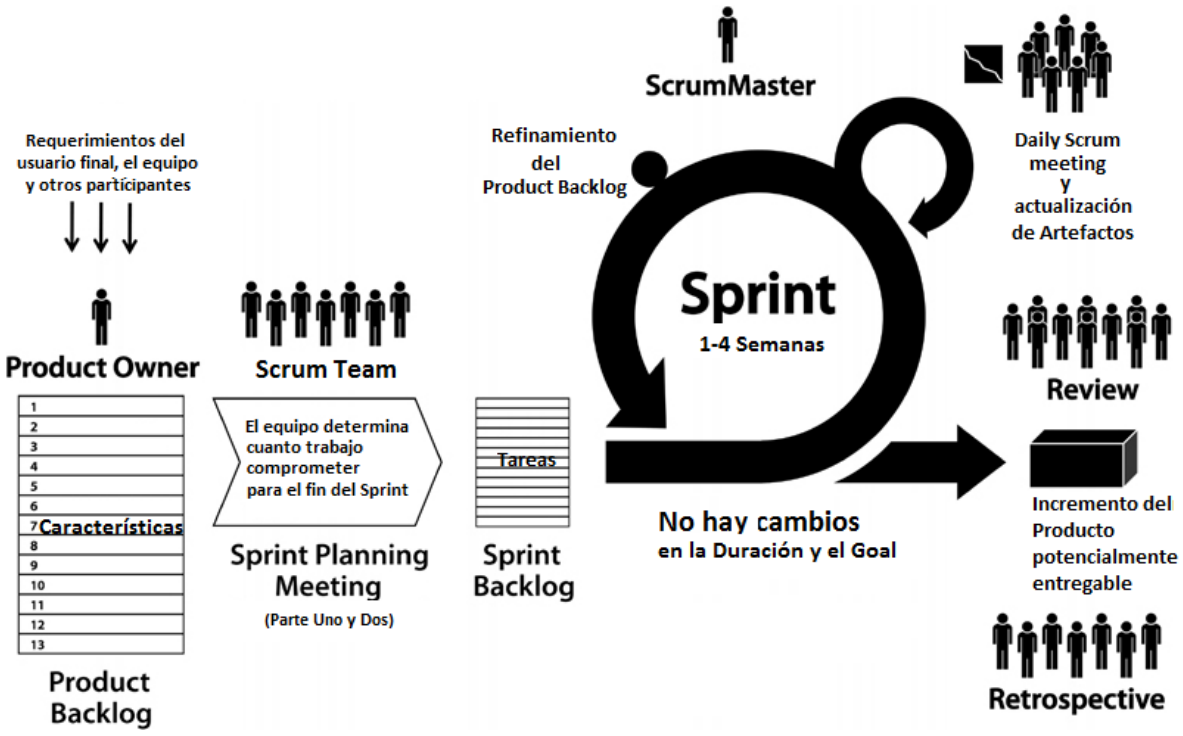

Figura 3.1 Los roles, artefactos y eventos de Scrum

Los Teams de Scrum tienen como objetivo potenciar la flexibilidad y la productividad. En este sentido se caracterizan por ser auto gestionables, multidisciplinarios y trabajaren iteraciones.

En Scrum se manejan tres roles principales, el Scrum Master quién vigila que el proceso se entienda y se ejecute correctamente, el Product Owner que es responsable de optimizar el resultado del trabajo desarrollado por el equipo y finalmente el Team o equipo que hace el trabajo. El equipo es un grupo de desarrolladores que cuenta con habilidades suficientes para

convertir los requerimientos del Product Owner en un producto potencialmente liberable al final de cada Sprint.

Scrum promueve la asignación de tiempo específico para tareas vitales (Time-Boxes). El Release Planning Meeting, el Sprint Planning Meeting, el Sprint, el Daily Scrum, el Sprint Review y el Sprint Retrospective cuentan con tiempo definido para su ejecución. El corazón de Scrum es el Sprint, que representa una iteración con una duración determinada por el esfuerzo para culminar el trabajo comprometido.

Scrum emplea cuatro artefactos principales. El Product Backlog es una lista con prioridades de todo aquello que el producto necesita. El Sprint Backlog es una lista de tareas obtenidas a partir de elementos del Product Backlog que se convierten en un incremento. Un Release Burndown es un mecanismo de medición que permite evaluar lo que resta del Product Backlog de acuerdo al tiempo de un plan de liberación. Un Sprint Burndown mide el restante de las tareas de un Sprint Backlog de acuerdo al tiempo de duración del Sprint.

Las reglas permiten integrar los eventos, roles y artefactos. Por ejemplo, los asistentes al Sprint Planning Meeting deben ser el Product Owner y los miembros del equipo de desarrollo.

3.3.1 Roles

El Scrum Master se encarga de que todos los integrantes del equipo, incluyendo el Product Owner entiendan y sigan las prácticas de Scrum. Auxilia al equipo y a la organización en el proceso de adopción de Scrum.

El **Scrum Master** hace todo lo que se encuentra a su alcance para que el equipo sea más productivo y genere productos de mejor calidad. El Scrum Master no es el manager del equipo o el administrador del proyecto, de hecho ambos roles no existen en Scrum. Tampoco es recomendable que el Product Owner juegue el papel de Scrum Master dado a que existen serias diferencias entre sus responsabilidades.

El **Product Owner** es el responsable de optimizar el valor del producto desarrollado por el equipo. Es de su competencia identificar, enlistar y priorizar los elementos del Product Backlog. Solo el tiene la facultad para determinar qué elementos se atenderán durante el próximo Sprint. Es su responsabilidad actualizar y refinar el Product Backlog continuamente.

El **Team** es un grupo de desarrolladores enfocados a la producción de incrementos potencialmente liberables. Scrum promueve equipos formados por personas con habilidades múltiples, que puedan participar en diferentes áreas del desarrollo, lo cual no significa que no puedan especializarse en un área en particular. Los recursos con frecuencia se especializan en programación, arquitectura, diseño de interfaces de usuario, diseño de base de datos o análisis del negocio.

Un equipo con las características mencionadas tiene la capacidad de auto organizarse, no necesita que alguien le indique la forma de trabajar para producir buenos resultados. Cada uno de los miembros aporta su expertise en la solución de problemas.

El número óptimo de integrantes de un Team de Scrum es 7, 2 más o 2 menos.

3.3.2 Time-Boxes

Los Time-Boxes de Scrum son el Release Planning Meeting, el Sprint, el Sprint Planning Meeting, el Sprint Review, el Sprint Retrospective y el Daily Scrum.

El propósito del **Release Planning** es establecer un plan y un objetivo que tanto el Scrum Team como el resto de la organización entiendan y puedan transmitir. Durante este se estudia la manera de superar las expectativas del cliente con respecto a su inversión. El plan de liberación establece el objetivo de la liberación, los elementos del Product Backlog con mayor prioridad y las funcionalidades sobresalientes de la liberación. Se especifica una fecha de liberación y un costo que prevalecerá si no surgen cambios radicales. De este modo es posible monitorear el progreso y realizar cambios sobre el plan de liberación conforme avanzan cada Sprint.

La mayoría de las organizaciones cuentan con un proceso de planeación, que comúnmente favorece el ejercicio de esta actividad durante la fase inicial del proyecto sin afectar el resultado al transcurrir el tiempo. Scrum por su parte recomienda definir únicamente el plan de liberación, el objetivo principal y los resultados probables, dejando el resto de la planeación para otros instantes, por ejemplo durante el Sprint Review y el Sprint Planning meeting, incluso en el Daily Scrum meeting.

El **Sprint** es una iteración con un periodo de duración definido. El trabajo realizado a lo largo del Sprint contribuye a la liberación de un producto o sistema. La definición del tiempo de la iteración juega un papel fundamental en el éxito o fracaso del producto resultante. Si se elige un periodo demasiado largo, se corre el riesgo de que surjan cambios y no estar sujeto a ellos. Scrum promueve el uso de periodos con una duración no mayor a un mes. Este tiempo es viable para controlar los riesgos y predecir el curso del desarrollo.

No es común que se cancele un Sprint, pero en caso de ser necesario la única entidad con facultad para hacerlo es el Product Owner. La causa de una interrupción puede ser resultado de un objetivo obsoleto.

Si un Sprint se termina prematuramente se analizan los elementos del Product Backlog que fueron “terminados” y se admiten si representan un incremento liberable. El resto de elementos se colocan de regreso en la lista con sus valores iníciales, tomando en cuenta que cualquier esfuerzo empeñado en su desarrollo es una perdida. El proceso de reubicar a un equipo en otra dirección, sin duda acarrea perdidas.

Durante el **Sprint Planning Meeting** se organiza el trabajo del Sprint. El tiempo programado para la actividad es de ocho horas para iteraciones de un mes y de un 5% del tiempo total del Sprint para aquellas con menor duración. Se divide en dos partes de 4 horas cada una, la primera está orientada a la identificación del “Que” y la segunda al “Cómo”.

En la primera parte del Sprint Planning el Product Owner le plantea al equipo cuales son los elementos del Product Backlog que tienen mayor prioridad. En conjunto determinan que funcionalidades serán construidas durante el próximo Sprint. Los insumos necesarios para llevar a cabo la tarea son el Product Backlog, el último incremento, el equipo y el registro de su

desempeño anterior. La cantidad de trabajo que compromete el equipo es directamente su responsabilidad.

Una vez que se precisan los elementos del Product Backlog que se van a satisfacer, es necesario especificar un Goal u objetivo. Plantear un objetivo en forma de enunciado permite mantener al equipo en la dirección correcta durante el curso el Sprint. El objetivo del Sprint es una parte del objetivo general de la liberación. Si el esfuerzo para alcanzar un objetivo es demasiado grande se puede pactar con el Product Owner para desglosarlo y entregar una funcionalidad parcial.

En la segunda parte se averigua el modo en que se van a satisfacer los elementos del Product Backlog comprometidos. Con frecuencia el equipo comienza por identificar las tareas, que describen fragmentos de trabajo detalladamente que en su momento se convierten en software funcional. Las tareas deben segregarse lo suficiente para que puedan ejecutarse en un tiempo menor al de una jornada laboral. A la lista de tareas resultante se le denomina Sprint Backlog. El modo en que se ponderan y organizan los elementos de la lista es responsabilidad del equipo de desarrollo.

Es importante que el Product Owner esté presente durante la segunda parte para darle claridad al contenido del Product Backlog. El equipo de desarrollo tiene la facultad para elegir la cantidad de trabajo que puede comprometer de acuerdo a sus capacidades, únicamente necesita acordarlo con el Product Owner. Incluso puede solicitar apoyo de otras entidades para recibir asesoría técnica o de dominio. Esta actividad le permite al equipo reconocer sus capacidades y detona su auto organización.

Una vez que el Sprint culmina, se desarrolla el **Sprint Review meeting**. El tiempo programado para esta actividad es de cuatro horas para iteraciones de un mes, y de un 5% del tiempo total del Sprint para aquellas con menor duración. En este punto el equipo y el Product Owner interactúan nuevamente. De esta manera el Product Owner averigua el estado del producto, y el equipo conoce el estado del Product Backlog.

A través de la presentación del incremento producido durante el Sprint, el Product Owner puede identificar los elementos que se culminaron y los que no. El equipo reconoce lo que se realizó con éxito y los problemas que enfrento, sin olvidar las posibles soluciones. Posteriormente el Product Owner explica el estado del Product Backlog. En conjunto, apoyados en el desempeño mostrado y los resultados obtenidos, se planea con respecto al siguiente Sprint.

Después del Sprint Review meeting continúa el **Sprint Retrospective**, una práctica muy útil que usualmente se omite. Sirve como herramienta para la identificación de los resultados surgidos del empleo de Scrum. Durante un periodo de tres horas se analiza el proceso y las actividades comprendidas en el marco de trabajo de Scrum para optimizar el proceso de desarrollo. El Product Owner es bienvenido a participar en el evento, pero es prescindible.

El propósito de la actividad es inspeccionar el desempeño mostrado en el último Sprint en razón de los participantes, la interacción, los procesos y las herramientas. La inspección facilita la discriminación de los elementos desarrollados con éxito de los que fracasaron. Los elementos

susceptibles a análisis son la composición del Scrum Team, los eventos, las herramientas y la definición de “Terminado”.

Una buena práctica es organizar los elementos en dos columnas, los que funcionaron correctamente y los que podrían mejorar. Cada asistente postula dos elementos en cualquiera de las columnas. Entonces se analizan las posibles causas y las alternativas de solución. Para agregar valor a la actividad se pueden marcar los elementos de las columnas con una “C” si fue causado por Scrum, “E” si fue expuesto por Scrum o “U” (Unrelated) si es ajeno a Scrum.

Una vez que el Sprint comienza, el equipo de desarrollo colabora en una de las actividades más relevantes de Scrum, el Daily Scrum. Es una actividad que se realiza diariamente a una hora específica y no consume más de 15 minutos. En este punto cada miembro expone al resto del equipo las actividades que culmino satisfactoriamente, los obstáculos que halló y las actividades programadas para la jornada subsecuente. A pesar de que el equipo de desarrollo es responsable de conducir el evento, el Scrum Master se asegura de que sea breve mediante la aplicación de las reglas y supervisando que la cada participación sea corta

El Daily Scrum no es una reunión para reportar el estatus del desarrollo. Es un evento exclusivo para los que laboran en la producción de un Incremento liberable. Es útil para evaluar el avance con respecto al objetivo del Sprint, lo que permite adecuar la forma de trabajo para alcanzar la meta.

3.3.3 Artefactos de Scrum

Los artefactos de Scrum comprenden el Product Backlog, el Release Burndown, el Sprint Backlog, y el Sprint Burndown. Adicionalmente se encuentran la Definición de Terminado (Definition of Done), el Release Plan, el Resumen de horas disponible, entre otros.

3.3.3.1 Product Backlog

Los requerimientos del producto se enlistan en el Product Backlog. El Product Owner es el responsable del contenido, la disponibilidad y la priorización del Product Backlog. Al Product Backlog nunca se le puede catalogar como completo, porque evoluciona al mismo ritmo que el producto y su entorno.

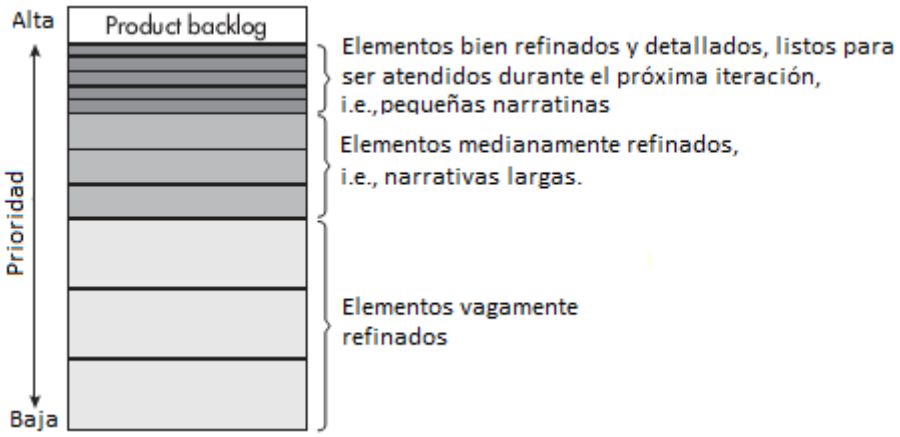

Figura 3.2 La priorización del Backlog determina el nivel de detalle

El contenido del Product Backlog no está limitado a los requerimientos, puede comprender funcionalidades, recursos, mejoras, actividades, en otras palabras elementos que contribuyen a la constitución del producto final.

Los elementos del Product Backlog se estiman en puntos o días. Dimensionar el tamaño de los elementos ayuda a priorizar y a planear la liberación. La forma en que se estiman los elementos del Product Backlog no es tan precisa como en el caso de los elementos del Sprint Backlog.

El Product Backlog es de carácter orgánico. El contenido del Product Backlog cambia recurrentemente, mediante la adición de elementos o la modificación de los existentes. La edición se da en términos, de la prioridad, el refinamiento, o incluso la remoción.

El Product Backlog se detalla convenientemente. Los elementos que tienen mayor prioridad se describen con mayor detalle, es decir que a mayor prioridad, mayor detalle. De esta forma el Product Backlog se mantiene conciso, y los elementos próximos a implementar siempre están listos. En general los requerimientos son identificados, descompuestos y refinados a lo largo del proyecto.

A pesar de que el Product Owner es el responsable directo del Product Backlog, Scrum favorece la participación del Scrum Team durante su elaboración y mantenimiento. Trabajar de manera conjunta mejora la calidad de los requerimientos y unifica conceptos. La actividad puede producirse diariamente, semanalmente, en una sesión completa antes de la culminación del sprint o durante el Sprint Review.

La tarea de descubrir y describir los elementos del Product Backlog es un proceso continuo. Para tener éxito es necesario entender que los requerimientos no son estáticos y que nuevos requerimientos surgen todo el tiempo. La actividad no está limitada a una fase inicial, sino que comprende todo el proyecto.

Las siguientes son sugerencias para la identificación de elementos del Product Backlog.

- No desgastarse tratando de incluir todos los elementos posibles.

- Concentrarse en la mínima funcionalidad para alcanzar un producto funcional.
- De ser posible evitar los elementos grandes y complejos.
- Resistir la tentación de agregar mayor detalle del necesario.
- Los elementos con menor prioridad deben permanecer con el mínimo detalle hasta que cambien de prioridad o se resuelvan los elementos críticos.
- Agregar únicamente elementos que representan una necesidad para el usuario.

La forma de **describir**los elementos del Product Backlog en Scrum es libre. Una alternativa es emplear narrativas que describan escenarios en los que el Usuario o el Cliente interactúan con el Producto. Su definición comprende el nombre, una breve narrativa, criterios de aceptación y parámetros de culminación. En el caso de los requerimientos de usabilidad es preferible utilizar prototipos o bocetos.

Trabajar con el Product Backlog no condiciona el uso de otros artefactos, por ejemplo las narrativas de casos de uso, la definición de roles de usuario y reglas de negocio, hojas de cálculo para la descripción de operaciones complejas, bocetos de la interfaz de usuario, storyboards, diagramas de navegación de la interfaz de usuario, y prototipos de la interfaz de usuario. El empleo de otros artefactos ayuda a precisar el contenido del Product Backlog. La recomendación es utilizar solo aquellos artefactos que contribuyen a la construcción de un producto funcional.

Una manera de **organizar** el contenido del Product Backlog es por temas. La agrupación de elementos relacionados por temas, facilita la priorización y el acceso a la información. El escenario de la aplicación de Moda Online, los elementos del Product Backlog se agruparon de la siguiente manera, Arquitectura, Productos, gestión de Contenido, Carrito de Compras, Pedido y Administración de Pedidos.

Figura 3.3 Ejemplo de Organización del Product Backlog por Tema

Tema	Elemento	Elemento detallado (Narrativa)	Prioridad	Esfuerzo
Administración de Pedidos	Administrar Pedidos	Como Administrador quiero gestionar la información relevante a los Pedidos de los Socios.	41	13
	Administrar Empresas de Paquetería	Como Administrador quiero gestionar la información correspondiente a las Empresas de Paquetería.	42	5
	Administrar Guías	Como administrador quiero gestionar la información concerniente a las Guías de Envío.	43	5

Para **priorizar** es necesario identificar el nivel de importancia de cada elemento. La participación del Scrum Team en la actividad promueve la difusión y estandarización de la información.

La priorización conduce al equipo hacia la resolución de los elementos más importantes, lo que origina una ventana de tiempo para el análisis, la toma de decisiones y la retroalimentación del cliente con respecto al resto de elementos. Una toma de decisiones más productiva deriva en un mejor producto.

El valor juega un papel importante para la priorización. Un elemento es valioso si es primordial para la liberación de un producto funcional. De otra manera puede ser desplazado al fondo del Product Backlog o incluso pospuesto para otra versión.

Si se concluye que un elemento es necesario se sugiere averiguar si existe una alternativa de solución viable, que reduzca el esfuerzo, el tiempo o el costo.

Las **dependencias** que surgen entre los elementos del Product Backlog limitan la libertad para priorizar. Por ejemplo, existen requerimientos funcionales que dependen de otros requerimientos funcionales o no funcionales, esto obliga a que culmine el desarrollo de uno para continuar con el otro. Una manera de disminuir el impacto es identificar la funcionalidad que vincula a los requerimientos y gestionarla de forma independiente.

Un excelente auxiliar para la priorización es la **liberación temprana** y frecuente. Esta práctica facilita el reconocimiento de la necesidad real del cliente y del Usuario. Si el Scrum Team no tiene la certeza sobre si implementar una funcionalidad o como implementarla, una liberación temprana es un apoyo. Una liberación puede satisfacer completa o parcialmente un Tema del Product Backlog.

No es necesario empeñarse en hallar todos los requerimientos, es más provechoso reexaminar los existentes. Con regularidad surgen más requerimientos durante el desarrollo de la solución y como resultado de una mejor concepción de las necesidades del cliente.

El riesgo es un componente intrínseco del Desarrollo de Software. Tiene una relación directa con la incertidumbre, ya que a mayor incertidumbre mayor es el riesgo en el proyecto. La incertidumbre es producida por la falta de conocimiento, si no conocemos los que desarrollamos y como desarrollarlo se incrementa la incertidumbre. **Conocimiento, incertidumbre y riesgo** están asociados consecuentemente.

Debido a que el riesgo afecta el éxito del proyecto, los elementos que generan incertidumbre y riesgo deben tener mayor prioridad. Preponderar estos elementos promueve un enfoque orientado a riesgos, hecho que refuerza el fracaso temprano. Equivocarse de manera temprana acelera la generación de conocimiento, reduce la incertidumbre y el riesgo. Es posible, por ejemplo modificar la arquitectura, cambiar de tecnología o modificar la composición del equipo cuando todavía hay oportunidad. Para algunos individuos u organizaciones es complicado reconocer los errores como oportunidades de aprendizaje y mejora, porque están acostumbrados a procesos de desarrollo tradicionales donde los errores aparecen de forma tardía y representan complicaciones.

Calcular el **tamaño** de los elementos del Product Backlog permite reconocer la cantidad de esfuerzo necesario para construirlos. Existen dos tipos de estimación, una más superficial que se hace con respecto a los elementos del Product Backlog y una más detallada que tiene como objeto los elementos del Sprint Backlog. En la primera se emplean puntuaciones y en la segunda horas.

Las puntuaciones representan estimaciones de esfuerzo y tamaño. Pero como ambos factores son subjetivos se recomienda emplear las puntuaciones de la tabla X.X.

Figura 3.4 Rangos de puntuaciones para narrativas

Puntuaciones	Tamaño	
0	Elemento implementado	
1	XS	Extra Small
2	S	Small
3	M	Medium
5	L	Large
8	XL	Extra-Large
13	XXL	Double extra-large
20	XXXL	Huge
40		Rango Tentativo
100		Rango Tentativo

Emplear valores predeterminados en el proceso de estimación reduce el tiempo invertido en la tarea. Las estimaciones hechas por un equipo no tienen que ser equiparables a las de otro. Para que un elemento reciba una magnitud es necesario que todo el equipo esté de acuerdo.

Figura 3.5 Product Backlog Inicial de la Aplicación de Moda Online

Tema	Elemento	Elemento detallado (narrativa)	P	E	Requerimientos no Funcionales
	Elaborar Modelo EER	Elaborar Modelo Inicial de EER de la Base de Datos	1	5	
	Elaborar Propuesta de Diseño	Elaborar Propuesta de Diseño de las principales piezas que componen el aplicativo.	2	3	
Arquitectura	Elaborar Esquema de Directorios y Paquetes	Elaborar esquema de paquetes y directorios para la organización de los Componentes tanto de Vista como de Back.	3	3	
	Integrar JSF con Spring	Integrar el Framework de JSF con Spring	4	3	
	Elaborar Template JSF	Elaborar un template JSF que establezca la organización general de los Componentes de Vista.	5	5	
	Integrar MyBatis con Spring	Integrar el Framework de MyBatis con Spring	6	5	
	Integrar Mail Sender Spring	Integrar el Servicio de envío de Correos electrónicos de Spring.	7	8	
	Integrar Spring Security	Integrar Framework Spring Security.	8	13	La autenticación debe generarse a través de un Popup

	Manejar Intentos fallidos	Manejar el número de Intentos fallidos y el bloqueo del Usuario.	9	8	
	Manejar Excepciones de Autenticación	Manejar las diferentes excepciones surgidas de un login fallido.	10	8	
	Autenticar mediante URL	Incorporar Autenticación mediante el uso de una URL válida.	11	5	
	Prueba de concepto para consumo de Imágenes	Realizar Prueba de Concepto para conseguir que las imágenes del aplicativo se consuman desde una ubicación diferente a la del aplicativo.	12	8	
Productos	Administrar Categoría	Como Administrador quiero gestionar las Categorías de Productos.	13	5	
	Administrar Proveedores	Como Administrador quiero gestionar los Proveedores de Productos del Negocio.	14	5	
	Agregar Descripción de Producto	Como Administrador quiero agregar la Descripción de un nuevo Producto.	15	2	
	Calcular Importe del Producto	Como Administrador quiero que al ingresar el Costo de Compra del Producto, se calcule automáticamente el Importe Total.	16	5	
	Agregar existencias de un Producto	Como Administrador quiero agregar la cantidad en existencia de un producto con Talla X.	17	5	
	Agregar Imágenes a un Producto	Como Administrador quiero agregar las imágenes de muestra de un Producto.	18	8	El producto puede tener un máximo de 4 imágenes asociadas. Deben guardarse en una ubicación fuera de los directorios del Proyecto
	Consultar Inventario	Como Administrador quiero visualizar en una tabla la lista de Productos del Inventario.	19	2	El renderizado de la lista de Productos debe de darse en un tiempo razonable. Debe ser exportable a formato de Excel.
	Editar Descripción de un Producto	Como Administrador quiero editar los datos descriptivos de un Producto.	20	2	
	Editar existencias de un Producto	Como Administrador quiero incrementar la existencia de un Producto.	21	2	
	Presentar Productos en Stock	Como Visitante o Socio quiero ver en un grid redimensionable los productos en Stock.	22	8	
Gestión de Contenido	Agregar Banner a la Página de Inicio	Como Administrador quiero agregar un Banner a alguna de las Secciones gestionables de la Página principal.	23	13	
	Establecer un producto en promoción.	Como administrador quiero especificar que un producto se encuentra en promoción.	24	2	
	Integrar Apache Velocity	Integrar Apache Velocity para el manejo de plantillas en Correos electrónicos.	25	8	

	Integrar pago con Paypal	Integrar el pago mediante Paypal	26		
	Manejar destrucción de la Sesión	Determinar una estrategia para el manejo de eventos surgidos a partir de la destrucción de la sesión.	27	5	
Carrito de Compras	Consultar disponibilidad de Productos Inmediata	Como Socio o Visitante quiero consultar la disponibilidad de un producto directamente desde el grid de su Categoría.	28	2	
	Especificar tiempo de vida	Implementar un mecanismo que permita limitar el tiempo de vida del Carrito de compras.	29	13	El tiempo de vida es de 30 min a partir de que se agrega el primer producto.
	Comprar Producto inmediatamente.	Como Socio quiero agregar un Producto a mi Carrito directamente desde el grid de su Categoría.	30	3	Como Visitante es posible consultar la disponibilidad de un Producto, pero para agregar al carrito es necesario ser Socio. Desde esta instancia se puede agregar un solo Producto.
	Visualizar Producto en Carrito.	Como Socio quiero que el Producto recién agregado se vea inmediatamente reflejado en mi Carrito.	31	13	Una vez que un Producto se ha agregado al Carrito, el número de existencias se debe ver afectado inmediatamente, tanto en la BD como en la vista.
	Gestionar Carrito de Compras	Como Socio quiero poder gestionar la cantidad y la talla de un Producto en mi Carrito. También quiero eliminar desde esta instancia.	32	8	
	Comprar Producto desde el Detalle.	Como Socio quiero comprar un Producto desde el Detalle del mismo.	33	5	En esta instancia es posible agregar un número mayor de uno de ocurrencias de un Producto.
Usuarios Socios	Registro de Socios	Como Visitante quiero convertirme en Socio mediante el registro.	34	13	De la misma manera que la Autenticación el flujo se da desde un Popup.
	Restaurar Contraseña	Como Socio o Administrador quiero restaurar mi contraseña.	35	5	El acceso a esta sección se da desde el propio Popup de Autenticación.
	Consultar Pedidos	Como Socio quiero consultar mis Pedidos.	36	5	
	Edición de Información Personal	Como Socio quiero editar mi información Personal.	37	5	
Pedido	Realizar Pedido	Como Socio quiero continuar con mi proceso de compra mediante la realización de un Pedido.	38	13	

	Notificar a Socio por Confirmación	Como Socio quiero recibir una notificación por medio de Correo electrónico acerca de la confirmación de un Pedido	39	5	
	Notificar a Socio por Envío	Como Socio deseo recibir una notificación por medio de Correo electrónico acerca del envío del envío de un Pedido.	40	5	
Administración de Pedidos	Administrar Pedidos	Como Administrador quiero gestionar la información relevante a los Pedidos de los Socios.	41	13	
	Administrar Empresas de Paquetería	Como Administrador quiero gestionar la información correspondiente a las Empresas de Paquetería.	42	5	
	Administrar Guías	Como administrador quiero gestionar la información concerniente a las Guías de Envío.	43	5	

3.3.3.2 Sprint Backlog

El Sprint Backlog es una lista de elementos comprometidos para su desarrollo y culminación durante el próximo sprint. Es vital que la identificación, selección y medición de los elementos la haga el equipo de desarrollo, debido a que es este quien se compromete a culminar las tareas.

El Sprint Backlog es un artefacto en movimiento que se actualiza diariamente para poder monitorear el progreso y el esfuerzo restante. La actualización comprende la adición o remoción de actividades y el registro del esfuerzo restante de cada una.

Una herramienta de apoyo para la estimación de esfuerzo de los elementos del Sprint Backlog es la **Definición de Terminado** (Definition of Done).La Definición de Terminado es una lista de requerimientos que debe cumplir un actividad de desarrollo de software para calificar como terminada. La elaboración o refinamiento del documento puede realizarse durante el Sprint Planning, lo cual ayuda al equipo a determinar la cantidad de trabajo a comprometer para el Sprint entrante. La Definición de Terminado que se acordó entre los miembros del equipo y el Product Owner se presenta en la Figura 3.6.

Figura 3.6 Definición de Terminado

Terminado con respecto a una narrativa

1. Código de acuerdo al Estándar especificado.
2. Pruebas Unitarias
3. Código versionado
4. Tablas y Campos de acuerdo al Estándar especificado.
5. Scripts de Base de Datos versionados (Si existen cambios).
6. Los componentes de Vista deben satisfacer el comportamiento solicitado (el diseño es relativo).
7. Documentar y comunicar (mediante email) cambios en la configuración fundamental del aplicativo.

Terminado con respecto a un Módulo

1. Diagramas Relevantes(Opcional)
   - Diagrama de Secuencia del Sistema
   - Diagrama de Clases
   - Diagrama Entidad Relación
   - Diagrama de Interacción
2. Código de acuerdo al Estándar especificado
3. Pruebas Unitarias
4. Código versionado
5. Tablas y Campos de acuerdo al Estándar especificado.
6. Scripts de Base de Datos versionados (Si existen cambios).
7. Los componentes de Vista deben satisfacer el comportamiento solicitado (el diseño es relativo).
8. Documentar y comunicar (mediante email) cambios en la configuración fundamental del aplicativo.

Terminado con respecto al Diseño de la Vista

1. Código versionado
2. El diseño debe mantener la línea especificada.

Otra actividad útil para la construcción del Sprint Backlog es el cálculo de horas disponibles por miembro. Es común que los integrantes de un equipo de desarrollo realicen actividades que no están directamente relacionadas al proyecto actual, por lo que es necesario contabilizar las horas que pueden invertir durante el próximo Sprint. El cálculo de horas correspondiente al Sprint Inicial se puede observar en la Figura 3.7.

Figura 3.7 Cálculo de horas disponibles por miembro para el Sprint Inicial

Duración del Sprint	4 Semanas

Miembro del Equipo	Días disponibles durante el Sprint	Horas Disponibles por día	Total de Horas disponibles
I	20	4	80
JL	20	4	80
W	5	2	10

Una vez que se define el tiempo con el cual dispone cada integrante para el próximo Sprint, el equipo arranca la construcción del Sprint Backlog. Recordar que la tarea se puede llevar a cabo en presencia del Product Owner, porque puede darle claridad al contenido del Product Backlog. El equipo selecciona los elementos con mayor prioridad para descomponerlos en tareas individuales y las coloca en el Sprint Backlog.

Figura 3.8 Sprint Backlog inicial

Objetivo de la Sprint (iteración):	Desarrollar la arquitectura para el soporte del aplicativo de Moda Online. Incluye la construcción de la funcionalidad básica para la administración y presentación de productos.

SPRINT BACKLOG			
Elemento (Item)	**Actividades**	**Responsable**	**EIE**
Elaborar Modelo EER	Elaborar el Modelo EER	JL,I	3
	Migrar el Modelo EER a Base de Datos	JL	1
Elaborar Propuesta de Diseño de las principales piezas que componen el aplicativo.	Plantear propuesta de Diseño para la vista de la página principal.	W	2
	Platear propuesta de Diseño para la vista de los Productos.	W	2
	Plantear propuesta de Diseño para la vista del detalle de cada Producto.	W	2
Elaborar esquema de paquetes y directorios para la organización de los Componentes tanto de Vista como de Back.	Diseñar el proyecto sobre del cual se construirá la arquitectura que soportará el proyecto.	JL	1
Integrar el Framework de JSF	Establecer la configuración correspondiente para el soporte de JSF con Spring.	JL	1
Elaborar un template JSF que establezca la organización general de los Componentes de	Elaborar un Template que especifique los diferentes componentes que conforman la vista de las páginas del aplicativo.	JL	3

Vista.	Incorporar los recursos para gestión de la vista del Aplicativo, Javascript, jQuery, Style Sheets.	JL	1
Integrar el Framework de MyBatis con Spring	Establecer la configuración correspondiente para el soporte de MyBatis con Spring.	I	4
	Anexar un par de ejemplos de acceso a datos con sus respectivas pruebas de unidad, empleando MyBatis.	I	2
Integrar el Servicio de envío de Correos electrónicos de Spring.	Establecer un diseño viable para un envío de correos genérico.	I	12
	Establecer la configuración para el empleo de Mail Sender de Spring para el envío de correos electrónicos.	I	4
	Anexar Pruebas de Unidad para el envío de correos electrónicos.	I	4
Integrar Framework Spring Security.	Establecer la configuración correspondiente para el empleo de Spring Security dentro de la solución.	I	8
	Hacer las adecuaciones necesarias para solventar el uso de un Popup para la Autenticación.	I	12
	Agregar funcionalidad para autenticación mediante URL.	I	6
Manejar el número de Intentos fallidos y el bloqueo del Usuario.	Integrar una solución que permita gestionar el número de intentos de autenticación fallidos, y permita darle seguimiento al posible bloqueo de los Usuarios.		6
Manejar las diferentes excepciones surgidas de un login fallido.	Estudiar las excepciones inherentes a la Autenticación de Usuarios del aplicativo y establecer una gestión centralizada de las mismas.	I	12
	Analizar la forma en que se deben presentar las excepciones al usuario, tomando en cuenta que deben ser dentro del mismo perímetro del Popup. Y aplicar propuesta.	I	6
Realizar Prueba de Concepto para conseguir que las imágenes del aplicativo se consuman desde una ubicación diferente a la del aplicativo.	Estudiar una solución para el almacenamiento y consumo de imágenes desde una ubicación diferente a la de los directorios del proyecto.	JL	3
	Establecer la configuración necesaria.	JL	2
	Realizar las pruebas correspondientes.	JL	3
Como Administrador quiero gestionar las Categorías de Productos.	Generar Mappers	JL	1
	Generar Servicio(s)	JL	0.5
	Generar Pruebas de Unidad	JL	0.5
	Generar ManagedBean(s)	JL	1
	Generar xhtml para la Inserción.	JL	1
	Generar xhtml que permita visualizar la lista de registros.	JL	1
	Generar xhtml para la Edición	JL	1
	Agregar la funcionalidad de edición al xhtml que	JL	0.5

	enlista los registros.		
Como Administrador quiero gestionar los Proveedores de Productos del Negocio.	Generar o editar Mappers	JL	1
	Generar o editar Servicio(s)	JL	0.5
	Generar Pruebas de Unidad	JL	0.5
	Generar ManagedBean(s)	JL	1
	Generar xhtml para la Inserción.	JL	1
	Generar xhtml que permita visualizar la lista de registros.	JL	1
	Generar xhtml para la Edición	JL	1
	Agregar la funcionalidad de edición al xhtml que enlista los registros.	JL	0.5
Como Administrador quiero agregar la Descripción de un nuevo Producto.	Generar o editar Mappers	JL	1
	Generar o editar Servicio(s)	JL	1
	Generar Pruebas de Unidad	JL	0.5
	Generar ManagedBean(s)	JL	1
	Generar xhtml para la Inserción.	JL	1
	Generar xhtml que permita visualizar la lista de registros.	JL	2
	Estudiar e implementar la funcionalidad para exportación del inventario a formato de Excel.	JL	2
Como Administrador quiero que al ingresar el Costo de Compra del Producto se calcule automáticamente el Importe Total.	Resolver e implementar la lógica y las operaciones que deben emplearse para realizar el cálculo.	JL	2
	Analizar el posible impacto en las entidades de Base de Datos y ejecutar los cambios necesarios.	JL	2
	Generar o editar Mappers	JL	0.5
	Generar o editar Servicio(s)	JL	1
	Elaborar pruebas unitarias.	JL	1
	Desarrollar la sección de vista en el xhtml correspondiente.	JL	1
Como Administrador quiero agregar la cantidad en existencia de un producto con Talla X.	Resolver e implementar la lógica y las operaciones que deben emplearse para realizar el cálculo.	JL	2
	Analizar el posible impacto en las entidades de Base de Datos y ejecutar los cambios necesarios.	JL	2
	Generar o editar Mappers	JL	2
	Generar o editar Servicio(s)	JL	2
	Elaborar pruebas unitarias.	JL	2
	Desarrollar la sección de vista en el xhtml correspondiente.	JL	2
Como Administrador quiero agregar las imágenes de muestra	Analizar el posible impacto en las entidades de Base de Datos y ejecutar los cambios necesarios.	JL	2

de un Producto.	Generar o editar Mappers	JL	1
	Generar o editar Servicio(s)	JL	0.5
	Elaborar pruebas unitarias.	JL	0.5
	Desarrollar la sección de vista en el xhtml correspondiente.	JL	2
Como Administrador quiero editar los datos inherentes a un Producto	Generar o editar Mappers	JL	2
	Generar o editar Servicio(s)	JL	2
	Elaborar pruebas unitarias.		2
	Agregar la funcionalidad de edición al xhtml que enlista los registros.	JL	2
	Desarrollar el módulo de Edición correspondiente.	JL	2
Como Visitante o Socio quiero ver en un grid redimensionable los productos en Stock.	Construir el Menú de Categorías de Productos que presentará de forma recursiva sus elementos y subelementos.	JL	3
	Construir el layout que hospedará los Productos de cierta Categoría.	W	2
	Estudiar e implementar una forma en la cual el layout de productos pueda reajustarse.	W	2

4 Análisis orientado a Objetos

En general la actividad de análisis está dirigida al estudio del problema y los requerimientos, no a la solución. Los desarrolladores formalizan los requerimientos mediante el análisis detallado de las condiciones que delimitan y que generan casos excepcionales.

El análisis orientado a objetos tiene como propósito la compresión del problema. En este tipo de análisis la recolección de requerimientos y la elaboración de artefactos no representan el total de actividades. Los casos de uso por ejemplo, que bien suelen ser importantes no tiene nada que ver con la orientación a objetos. Es importante que como parte del análisis se examinen los requerimientos para obtener un modelo mental que facilite tanto la comprensión como la simplificación del problema.

En un análisis orientado a objetos se pone énfasis en la descripción de los conceptos del dominio del problema. El resultado es un modelo que describe el dominio de la aplicación, el cual puede refinarse para describir la interacción entre los actores y la aplicación.

El modelo de análisis se compone de otros tres modelos, el modelo funcional representado por los casos de uso y escenarios, el modelo de objetos representado por diagramas de clases y de objetos, y el modelo dinámico representado por diagramas de secuencia.

4.1 Modelo de Dominio

La descomposición del dominio en conceptos significativos es una actividad primordial en el análisis orientado a objetos.

El modelo de dominio comprende clases conceptuales del mundo real, no componentes de software. En su estructura no figuran clases de software u objetos con responsabilidades.

Un modelo de dominio se compone por un conjunto de diagramas donde aparecen objetos del dominio o clases conceptuales, asociaciones entre clases y atributos de las clases. No se definen operaciones.

La información que ilustra bien podría expresarse mediante sentencias. Pero el lenguaje visual facilita la asimilación de conceptos y sus relaciones.

Un modelo de dominio útil contiene elementos e información que permiten comprender el dominio del problema, la terminología utilizada y relaciones, de acuerdo a los requerimientos planteados.

4.1.1 Identificación de Clases Conceptuales

De manera informal una clase conceptual representa una idea u objeto. En un contexto formal una clase conceptual se define en términos de su símbolo, intensión y extensión. El símbolo representa la clase conceptual en forma de palabras o imágenes. La intención es la definición de la clase conceptual. La extensión son los objetos para los que aplica la definición de la clase conceptual.

Cuando una solución de software es compleja, es recomendable dividir el espacio del problema en unidades fáciles de comprender. En un análisis estructurado la descomposición se hace de acuerdo a procesos o funciones, en cambio en un análisis orientado a objetos la descomposición se en razón de conceptos de dominio del problema.

Entonces para crear un modelo de dominio se deben identificar las clases significativas de los diferentes dominios de interés del aplicativo Moda Online. Se proponen los subdominios de Producto, Pedido, Carrito de Compras, Usuario y Layout. Los subdominios se pueden observar en la forma de paquetes en la Figura 4.1.

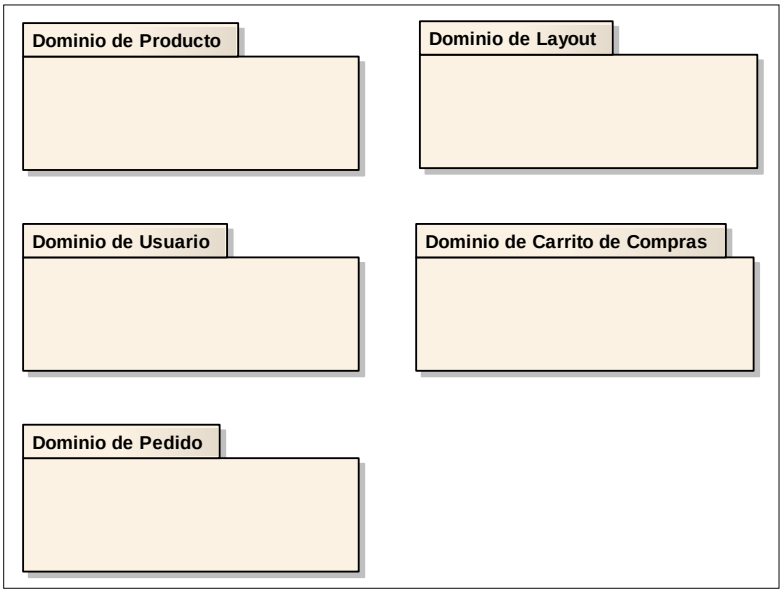

Figura 4.1 Dominios de interés para el aplicativo de Moda Online

La bibliografía sugiere tres técnicas para la identificación de clases conceptuales:

1. El reutilización o modificación de modelos existentes. Es la mejor alternativa, y con frecuencia la más sencilla. Existe un gran número de modelos de dominio bien elaborados y orientados a dominios comunes, por ejemplo, inventario, comercio, salud, entre otros. Adicionalmente se pueden utilizar patrones de análisis, que son modelos de dominio parciales, creados por expertos.
2. Empleo de una lista de categorías de clases conceptuales.
3. Identificación mediante clases nominales.

En el caso particular del desarrollo de la aplicación de Moda Online se optó por la primera opción, porque se entiende que no es necesario invertir tiempo en la búsqueda de la solución a un problema que ya fue analizado y resuelto por alguien más, y que el éxito de la solución está comprobado para entornos similares. La alternativa facilita el desarrollo de la actividad sin sacrificar la calidad en el modelo.

Tras realizar una búsqueda y selección de modelos de dominio vinculados a comercio electrónico, se realizó un estudio de los conceptos propuestos en cada uno de ellos y se determinó cuales coincidían con el dominio de la aplicación de Moda Online. Los conceptos que coinciden se pueden observar en la Figura 4.2.

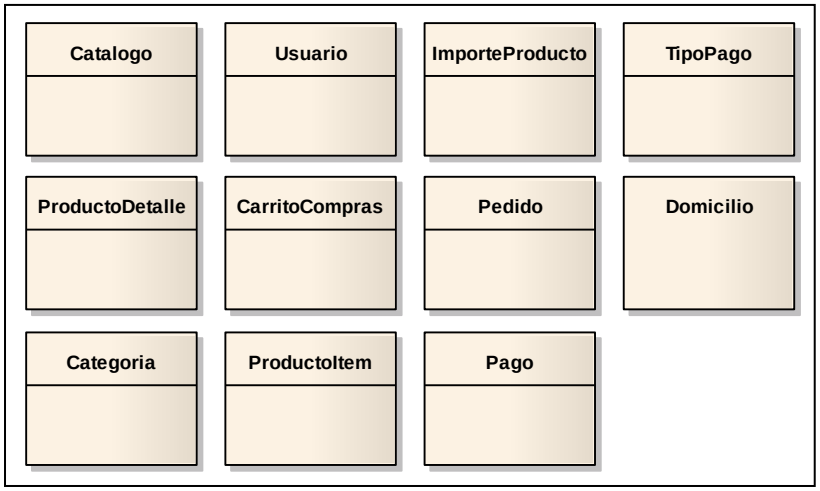

Figura 4.2 Conceptos comunes para el Modelo de Comercio Electrónico

Posteriormente cada miembro del equipo de desarrollo contribuyó con conceptos que considero significativos en el dominio, para su análisis y posible incorporación.

De acuerdo al marco de trabajo que se propuso para la construcción de la aplicación, que es de carácter iterativo, no es recomendable tratar de capturar todos los conceptos notables durante la elaboración del Modelo de Dominio inicial. El modelo se puede refinar de acuerdo al objetivo de cada Sprint y conforme avanza el desarrollo. En este sentido es admisible poner énfasis en la identificación de aquellas clases que tienen relación con el objetivo del Sprint en curso.

Uno de los objetivos del Sprint inicial para el desarrollo de Moda Online es la construcción de la funcionalidad para la administración y presentación de los productos, que predispone la identificación de conceptos relacionados con el Producto.

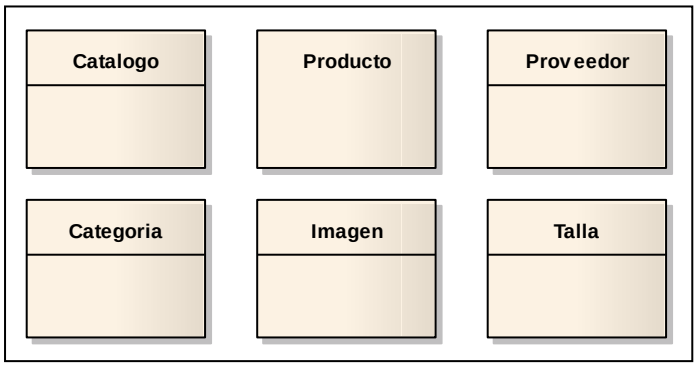

Figura 4.3 Modelo de Dominio Inicial del Producto

Otro objetivo del Sprint Inicial es la construcción de la arquitectura fundamental del aplicativo, que comprende la gestión de la seguridad. Para realizar la tarea es necesario determinar que conceptos del dominio están involucrados. Los conceptos que trascienden son los relacionados con la jerarquización e información vinculada al usuario.

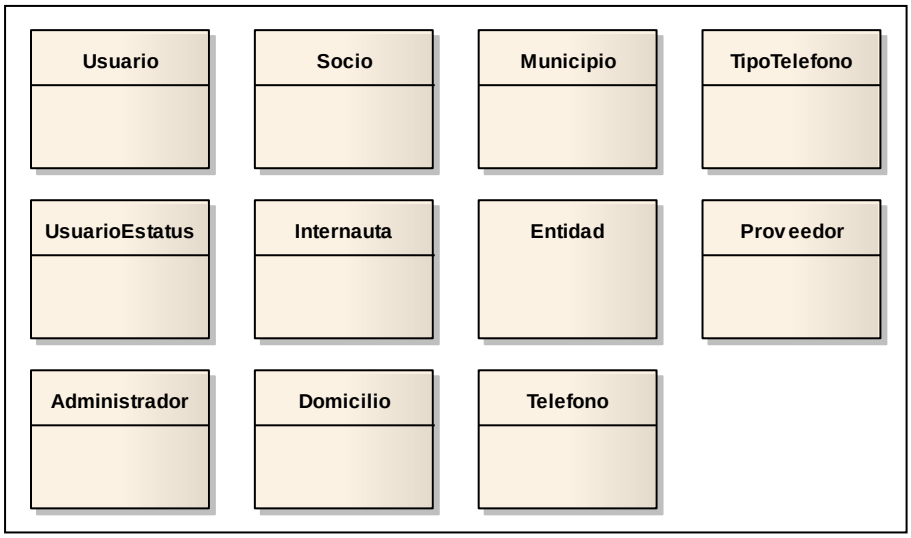

Figura 4.4 Modelo de Dominio Inicial de Usuario

El diseño fundamental de la vista es crítico para el diseño de la arquitectura, hecho que hace necesario pensar en la disposición de los componentes en la vista. La composición de la vista principal del aplicativo de Moda Online se divide en secciones que contienen banners.

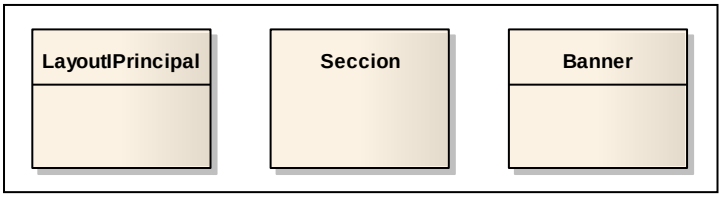

Figura 4.5 Modelo de Dominio Inicial de Layout de la Vista

Basado en el análisis del Sprint Backlog de la iteración inicial se decidió descomponer todos los dominios para entender mejor los conceptos del negocio y poder adquirir retroalimentación temprana de parte del cliente. Se puso mayor énfasis en los dominios que trascienden para el desarrollo del entregable más próximo (arquitectura y funcionalidad básica para administración de productos), dejando el refinamiento del resto para Sprints posteriores.

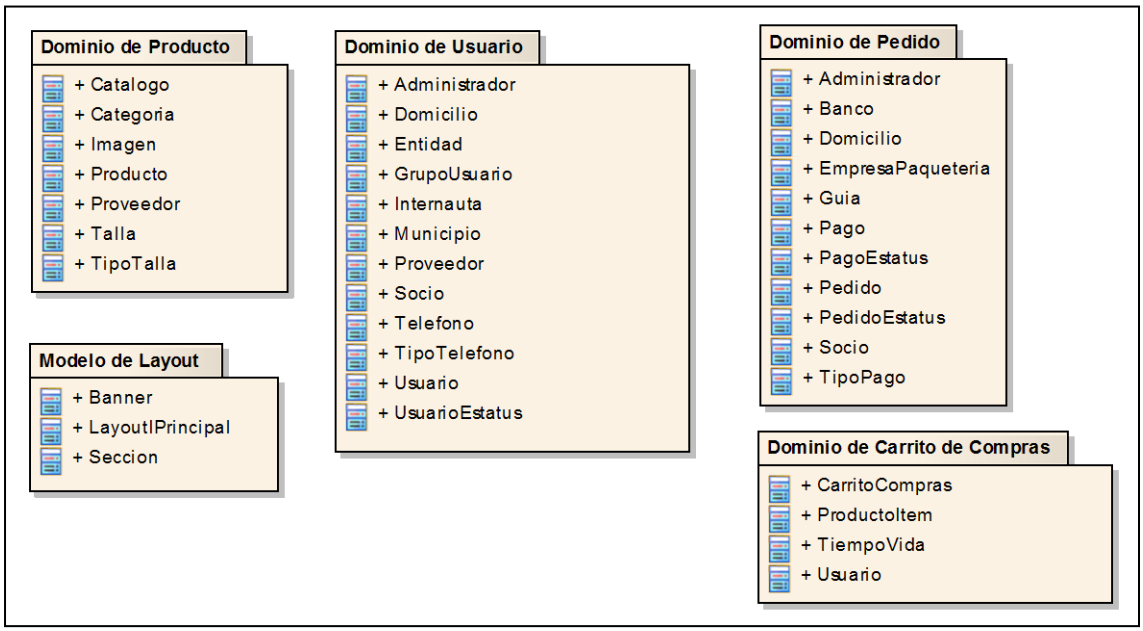

Figura 4.6 Descomposición Inicial de Subdominios de Moda Online

4.1.2 Asociaciones

Una asociación es una relación significativa entre tipos. En un sentido más formal se define como una relación semántica entre dos o más clasificadores.

Las asociaciones solo son útiles cuando su presencia favorece la comprensión del dominio. Se recomienda incluir asociaciones cuando:

- Es necesario saber y recordar de la relación.
- Se encuentra en la lista de Asociaciones Comunes (Figura 4.7).

Figura 4.7 Lista de Asociaciones Comunes

Categoría	Ejemplo
A es una transacción relacionada a otra transacción B	Pago-Venta Reserva-Cancelación
A es un artículo en una línea de transacción B	LineaVenta-Venta
A es un producto o servicio para una transacción	ReservacionVuelo
A es un rol relacionado a la transacción B	TicketPasajero
A es un parte Física o Lógica de B	AsientoAvion
A esta contenida física o lógicamente en B	Tablero, PasajeroAvion
A describe a B	DescripcionArticulo
A es conocido, registrado, salvado, capturado por B	PiezaTablero, ManifiestoVuelo
A es un miembro de B	CajeroTienda, PilotoAvion
A es una unidad de B	DepartamentoTienda
A usa, gestiona o posee a B	PilotoAvion, CajeroRegistradora

Recomendaciones para la introducción de asociaciones:

- Concentrarse en las asociaciones que vinculan información que trasciende en el dominio.
- Es más importante identificar las clases conceptuales que sus relaciones.
- Agregar muchas asociaciones acompleja la comprensión del dominio.
- Invertir demasiado tiempo en descubrir asociaciones es infructuoso.
- Evitar asociaciones redundantes o derivadas

Cada extremo de una asociación se denomina rol. Los roles pueden tener opcionalmente, nombre, expresión de multiplicidad y navegabilidad.

La multiplicidad determina cuantas instancias de una clase A pueden relacionarse con una clase B. El valor de la multiplicidad depende del dominio en análisis.

En la figura 4.8 se muestran algunas expresiones de multiplicidad comunes.

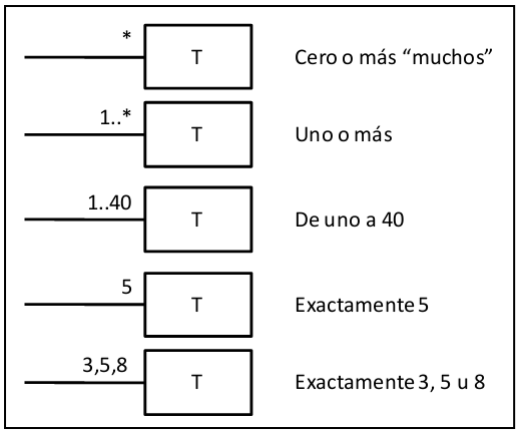

Figura 4.8 Multiplicidad de una asociación

Se sugiere que el nombrado de las asociaciones se haga empleando el siguiente formato NombreTipo-FraseVerbalNombreTipo. El nombre debe comenzar con una letra mayúscula, por ejemplo PagadoMediante o Pagado-Mediante.

Las asociaciones son expresiones puramente conceptuales. Indican una relación significativa que vale la pena conocer y recordar y su implementación no es estricta. Es posible que mientras se construye el modelo de dominio se definan asociaciones que no son necesarias durante la implementación. De manera opuesta se puede dar el caso en que existan relaciones que son demasiado obvias en el dominio que no sea necesario expresarlas en el modelo, pero que si se implementarán.

4.1.3 Atributos

Los atributos son útiles para satisfacer los requisitos de información de los diferentes escenarios del desarrollo. Al igual que el resto de los elementos opcionales del modelo de dominio, no es necesario sobre trabajar en su identificación e ingreso. Se debe considerar únicamente los atributos que expresan información que deba persistir.

Los atributos deben corresponder a tipos de datos primitivos, como enteros y booleanos. En el caso de atributos de carácter complejo es mejor modelarlos como clases conceptuales.

El hecho de preponderar la abstracción de un atributo como una clase, no significa que se deba tomar la misma acción durante la implementación de la solución en un lenguaje como Java o C#.

La sintaxis completa para la declaración de un atributo es la siguiente:

visibilidad nombre : tipo multiplicidad = default{cadena de propiedad}

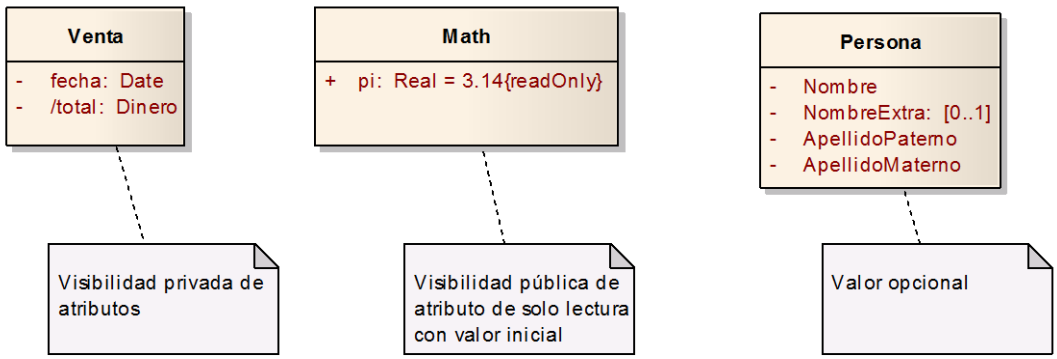

Figura 4.9 Ejemplos de atributos

Por convención se asume que los atributos tienen una visibilidad privada a menos que se señale lo contrario. La multiplicidad se utiliza para denotar la existencia opcional de valores.

Un atributo derivado obtiene su valor a partir del resultado de operaciones sobre valores relacionados con la clase que lo contiene.

En lo que corresponde a los modelos de dominio de los diferentes escenarios de Moda Online no se pondrá énfasis en la definición de atributos, en particular en sus tipos de datos puesto que no se considera productivo y el resultado puede adquirir un carácter relacional u orientado a objetos o una mezcla confusa. Sin embargo temas como la abstracción de clases complejas y el empleo de clases de magnitud se abordarán durante la aplicación de patrones de análisis.

La descomposición del subdominio de Producto es fundamental para la comprensión del negocio. Clases conceptuales que se pueden ver en la Figura 4.10 juegan un papel primordial en el flujo del negocio.

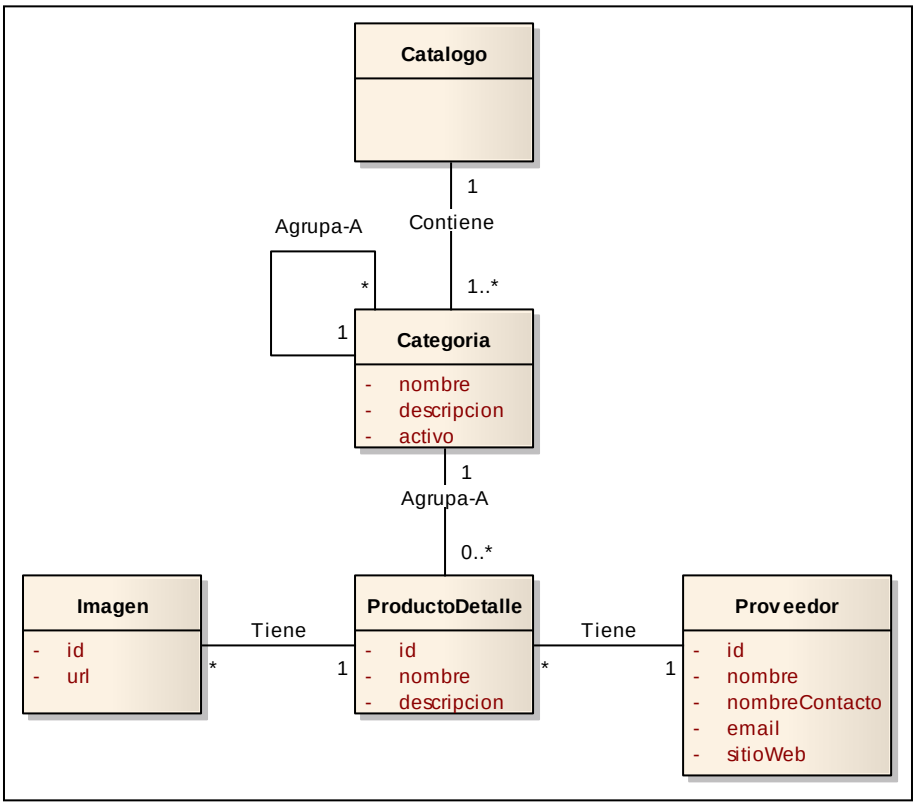

Figura 4.10 Modelo de Dominio de Producto con atributos y asociaciones

Se mencionó que para construir el modulo de seguridad de la arquitectura es importante abstraer los conceptos de Usuario, jerarquías de Usuarios e información del Usuario. La razón es que se planea emplear Spring Security, y el framework cuenta con su propia conceptualización de usuarios, de tal manera que se tiene que hacer una integración del dominio local. La propuesta de Dominio de Usuario de Moda Online se encuentra en la Figura 4.11.

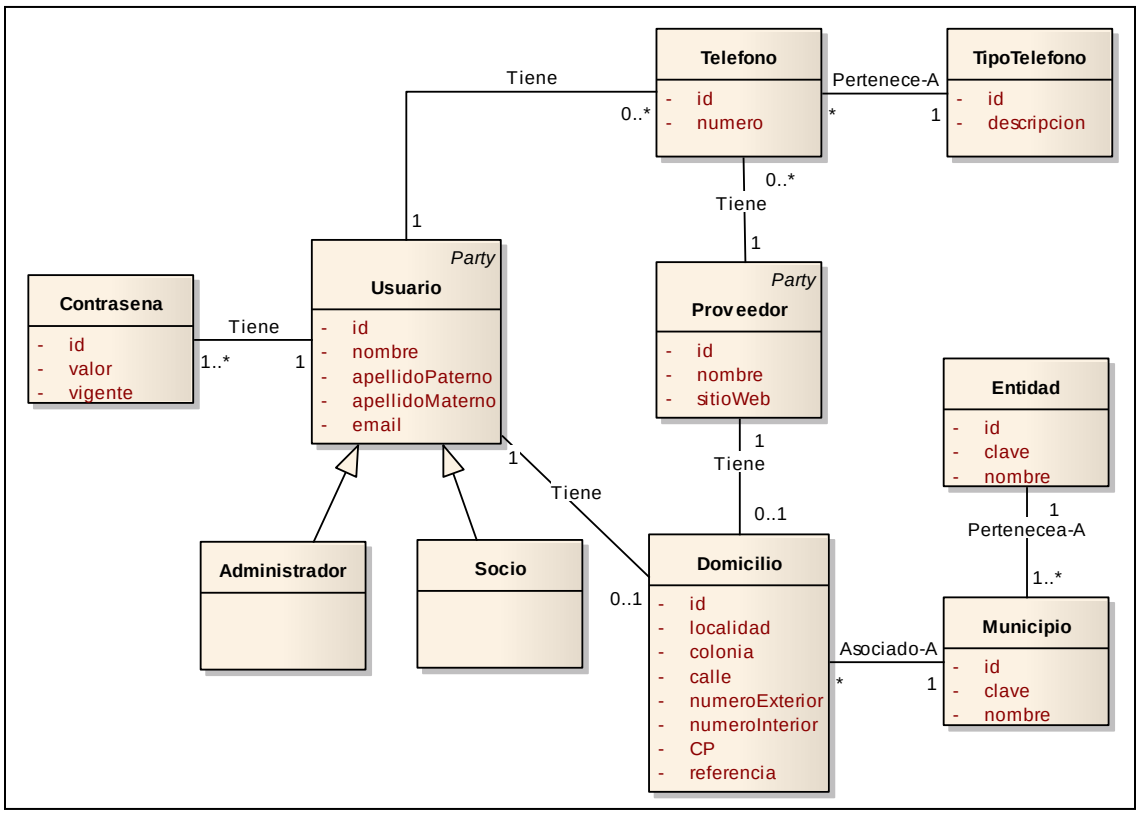

Figura 4.11 Modelo de Dominio de Usuario con atributos y asociaciones

De la misma manera que el subdominio de Usuario es relevante para el desarrollo de la arquitectura, también lo es el dominio de Layout. Los conceptos de este escenario permiten determinar la composición de la vista del aplicativo, mismos que se observan en la Figura 4.12.

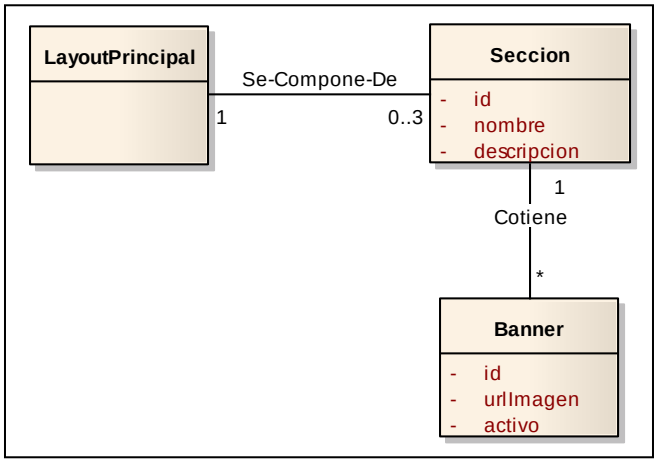

Figura 4.12 Modelo de Dominio de Layout con atributos y asociaciones

4.2 Patrones de Análisis

Los patrones de análisis son planteamientos que sirven de apoyo para la solución de problemas recurrentes. Son grupos de conceptos que forman estructuras comunes para el modelado de negocios.

Los patrones análisis surgen directamente de la experiencia con proyectos reales. Se puede afirmar que un patrón no se inventa, se descubre. Una vez que se comprueba el uso común de un planteamiento entonces se cataloga como patrón, y puede empleado en un contexto similar o diferente.

La definición de un patrón de análisis se hace en términos de los elementos o conceptos que forman parte del dominio. Los patrones sirven como apoyo para la construcción de modelos de dominio variados, por lo que pueden adecuarse, cuidando de no agregar flexibilidad innecesaria.

Los patrones de análisis y de diseño son notablemente diferentes. Los patrones de análisis describen problemas recurrentes empleando conceptos presentes en el dominio. Afectan la forma en que se produce el código, pero no influyen directamente en detalles de implementación como lo hacen los patrones de diseño.

Los patrones de análisis se organizan en las siguientes categorías:

- Accountability: Patrones para la definición de responsabilidades entre grupos.
- Observations y Measurements: Patrones para el registro de información.
- Refering to Objects: Patrones para la identificación de objetos.
- Inventory y Accounting: Patrones fundamentales para contabilidad.
    - Planning: Patrones que describen la relación entre una planeación estándar y una ocasional.
- Trading: Patrones enfocados a escenarios comerciales donde los costos fluctúan.

De acuerdo a los requerimientos del desarrollo de Moda Online los Patrones que se ajustan al Dominio del problema son los relacionados con Accountability, Observations y Measurements.

Se empleara el patrón de Party, que pertenece al grupo de patrones asociados a Accountability, de hecho la combinación entre este y el patrón de jerarquías organizacionales dan vida al patrón de Accountability.

También se utilizarán los patrones de análisis de Quantity y Observation que se encuentran en la categoría de Observations y Measurements.

En el caso del planteamiento de las Clases de Especificación se decidió abordarlo como un patrón, ya que cumple con las características de uno y se utiliza recurrentemente en la definición del resto de patrones, pero no recibe un nombre.

Los conceptos definidos en los patrones de análisis se integran a un modelo conceptual que tiene como objetivo facilitar la comprensión del dominio por parte del analista y generar un mecanismo de comunicación homogéneo entre los mismos analistas y con el cliente.

4.2.1 Modelos Conceptuales

Un modelo conceptual se puede construir utilizando técnicas de análisis y diseño para el modelado. Estas técnicas hacen que los involucrados en el modelado se concentren en los elementos conceptuales y no en cuestiones de diseño de software. Para conseguirlo se hace uso de componentes gráficos.

Las técnicas de análisis promueven la independencia de las tecnologías de implementación. De este modo la tecnología no afecta la comprensión del problema y el modelo resultante es útil a pesar de la tecnología. En la realidad el modelo no culmina cien por ciento libre de conceptos de implementación. La razón es que algunas personas están habituadas a usar técnicas de orientación a objetos, y otras técnicas relacionales para la abstracción de conceptos, lo que inevitablemente afecta el modelo.

Las siguientes son algunas características de los modelos conceptuales:

- El modelo es una herramienta para entender y para comunicar.
- Todos los modelos son aproximaciones del dominio que se trata de entender.
- Los modelos conceptuales se reducen a tipos y no a implementaciones.
- No se puede definir un modelo como correcto o incorrecto, es poco útil o muy útil.
- El mejor modelo es el más simple, aunque este no sea el primero que se considere.
- Los modelos conceptuales se construyen de acuerdo a los requerimientos.

Es a partir de este punto que se podrá observar la forma en que aplicaron las directivas y sugerencias para el ingreso de asociaciones, roles y atributos sobre los diferentes de modelos del dominio.

4.2.2 Clases de Especificación o Descripción

Las clases conceptuales de especificación son útiles para dominios de venta de productos o manufactura. Una clase de carácter descriptivo da soporte para la ocurrencia de entidades lógicas que no se afectan con la gestión de existencias.

Una clase de especificación alberga información relacionada al Producto, por ejemplo su identificador, nombre, descripción, etc. No representa al Producto físico que está directamente relacionado con el Inventario. De esta forma, a pesar de que se consuman todos los elementos inventariados, con sus correspondientes instancias de Software, prevalece la Descripción.

Basado en el estudio del dominio de Moda Online, se identificó que la clase conceptual que representa el producto físico puede comprender no solo un elemento, sino todos aquellos que coincidan en razón de su talla. Así es posible decir “del producto Blusa súper especial en talla Small hay 32 en existencia”. Lo planteado se puede ver claramente en el diagrama de la Figura

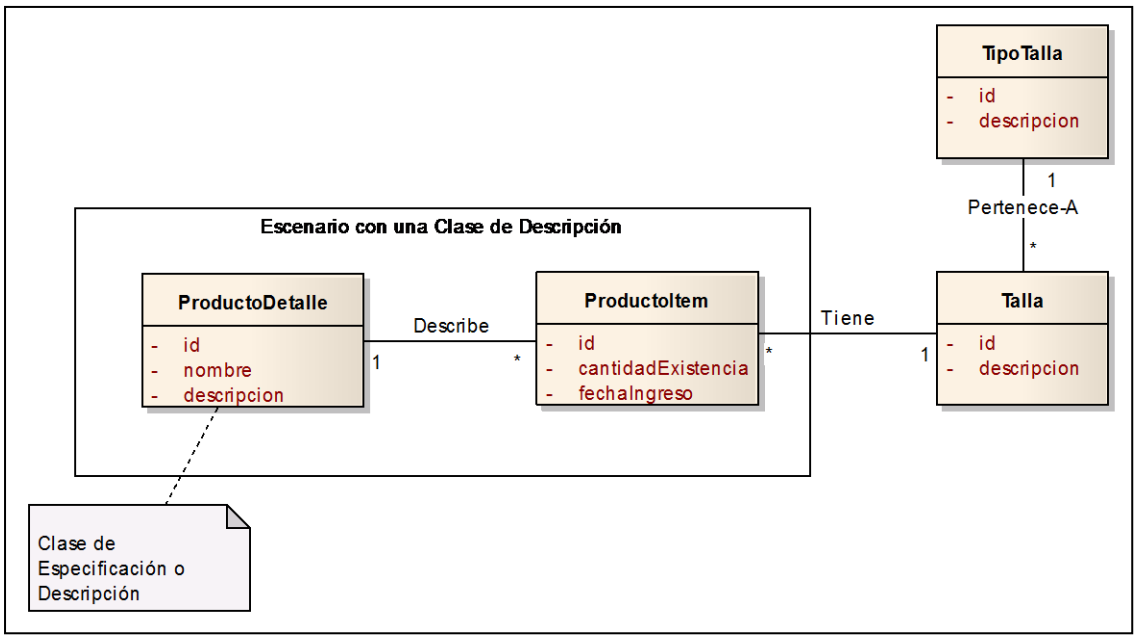

Figura 4.13 Clase de Especificación de Producto

En el diagrama también se percibe la existencia de un clasificador de Tallas, que se abstrajo empleando el patrón de análisis de Observation, que se verá más adelante.

4.2.3 Party

Party actúa sobre estructuras organizacionales y súper tipos. Hay un gran número de escenarios en los que se puede incluir a un individuo u organización en una misma definición. Se reciben cartas de personas y organizaciones, se realizan depósitos a personas u organizaciones. Tanto las organizaciones como las personas realizan operaciones, tienen cuentas de banco o pagan impuestos.

En el dominio de Usuario surgen dos entidades con similitudes, Usuario y Proveedor. Ambos comparten propiedades, como el Teléfono y el Domicilio. Es un hecho que un Proveedor puede ser una compañía y no una persona, de tal manera que si tratamos de establecer comunicación no obtendremos respuesta de una persona en particular, aunque si tal vez de un representante.

Lo primero que podemos hacer para modelar el escenario es establecer relaciones entre el Usuario, el Teléfono y el Domicilio, e introducir las mismas relaciones para el caso del Proveedor, como sucede en la Figura 4.11. Pero esto satura el modelo, por lo que se sugiere la introducción de un súper tipo.

En general el patrón Party plantea la introducción de una clase conceptual que funcione como súper tipo para una persona u organización. Así es posible asociar teléfonos o domicilios a individuos u organizaciones. El resultado de aplicar el patrón en el escenario del Usuario se puede ver en la Figura4.14. No olvidar que los diagramas generales de cada subdominio pueden sufrir cambios en cada Sprint.

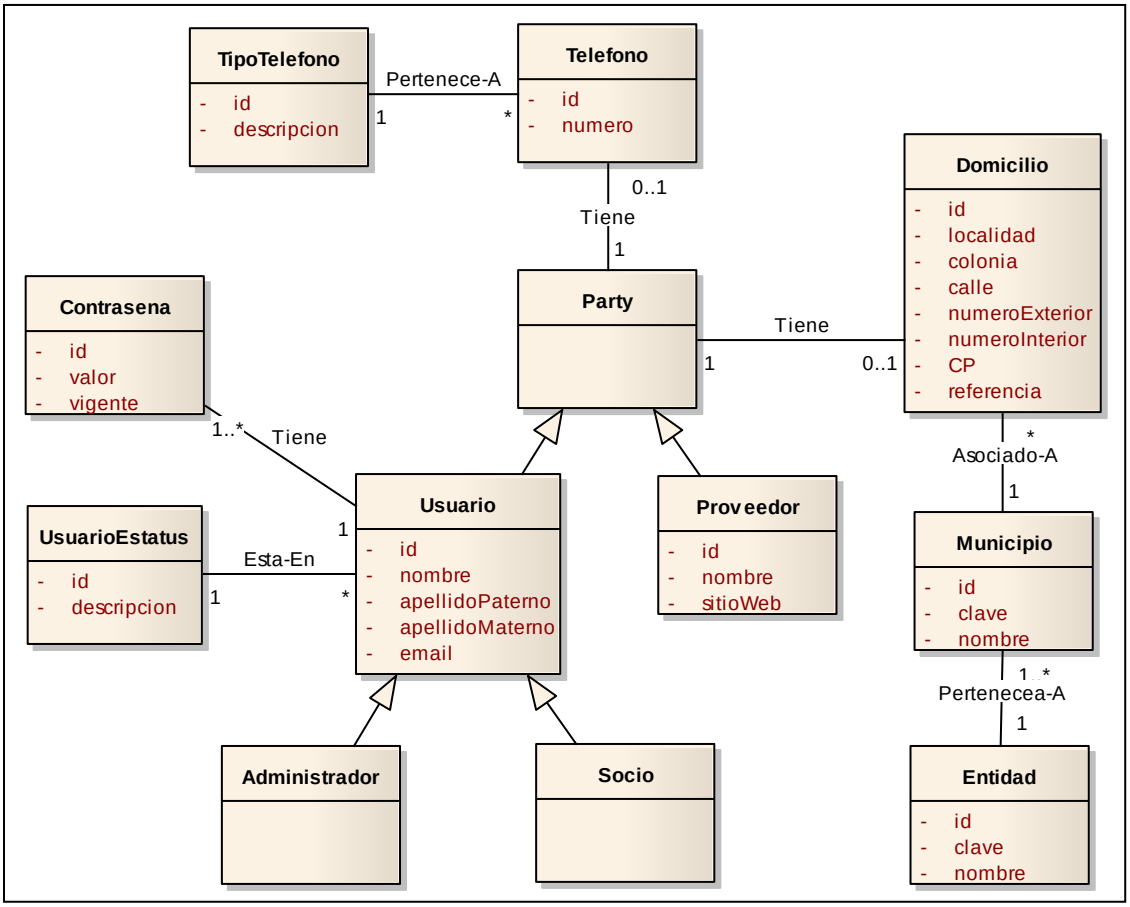

Figura 4.14 Subdominio general de Usuario empleando el Patrón de Análisis Party

4.2.4 Quantity

Cuando se realiza el análisis del dominio se puede hallar información que se expresa numéricamente. Pero utilizar solo un campo para persistir el dato no es la mejor opción, porque no hace sentido, por ejemplo decir que la altura de un individuo es 180 o su talla es 32. Es necesario introducir unidades para agregar sentido. Una alternativa es añadir la unidad al nombre del campo, lo que clarifica, pero limita el alcance de los conceptos (alturaEnMetros = 1.5). Otra desventaja es que al momento de ingresar la información es necesario contar con el dato exacto en la unidad referida.

El patrón de Quantity resulta útil en este tipo de escenarios. Porque sugiere representar una dimensión mediante un número y su correspondiente unidad. Combinar números y unidades ayuda a expresar lo que hay a nuestro alrededor con mayor precisión. Si las cantidades y sus unidades se modelan en clases independientes, es posible modelar conversiones o unidades compuestas.

La Figura 4.15 muestra un objeto con un tipo de dato que conjunta números y unidades, por ejemplo, 150 pesos ó 10 dólares. En este ejemplo se incluyen conceptos del dominio de Pedido de Moda Online, donde el Pago asociado al pedido tiene un Importe que puede encontrarse en pesos o quizás en dólares.

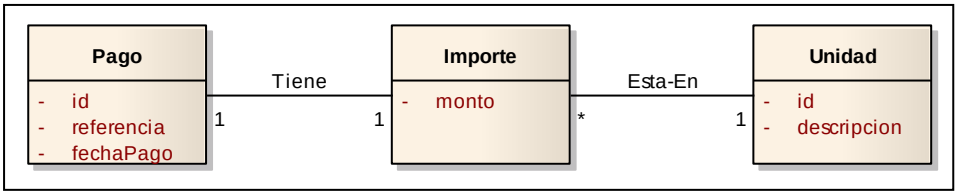

Figura 4.15 Conceptos del Dominio de Pedido que utilizan Quantity

Es común hallar dominios en los cuales se presentan relaciones entre números y unidades. Un ejemplo es el ámbito financiero, donde para expresar información monetaria podemos usar este planteamiento, porque produce flexibilidad con respecto al tipo de moneda que se utiliza.

Emplear Quantity es común en un análisis orientado a objetos. El analista suele emplearlo incluso sin percatarse. La forma en que se modela puede variar porque algunos prefieren emplear un atributo de tipo Quantity en lugar de una asociación como se puede ver en la Figura 4.16, pero es indistinto.

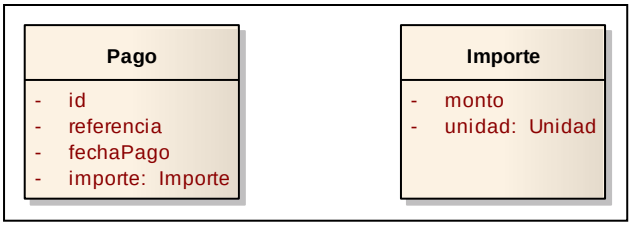

Figura 4.16 Conceptos del Dominio de Pedido que utilizan Quantity en forma de atributos

El Dominio de Producto presenta un par de conceptos que bien pueden ser modelados empleando el patrón de Quantity, pero tienen propiedades peculiares. El Importe asociado al producto no tiene solo un monto, tiene una lista de atributos que representan factores en una fórmula que se utiliza para el cálculo del monto. Atributos cómo el Costo de Compra, el Costo del Flete y el Porcentaje de Reinversión sirven para calcular el costo del Producto. Adicionalmente el Importe puede tener asociado varios Descuentos por diversos conceptos como se puede ver en la Figura 4.17.

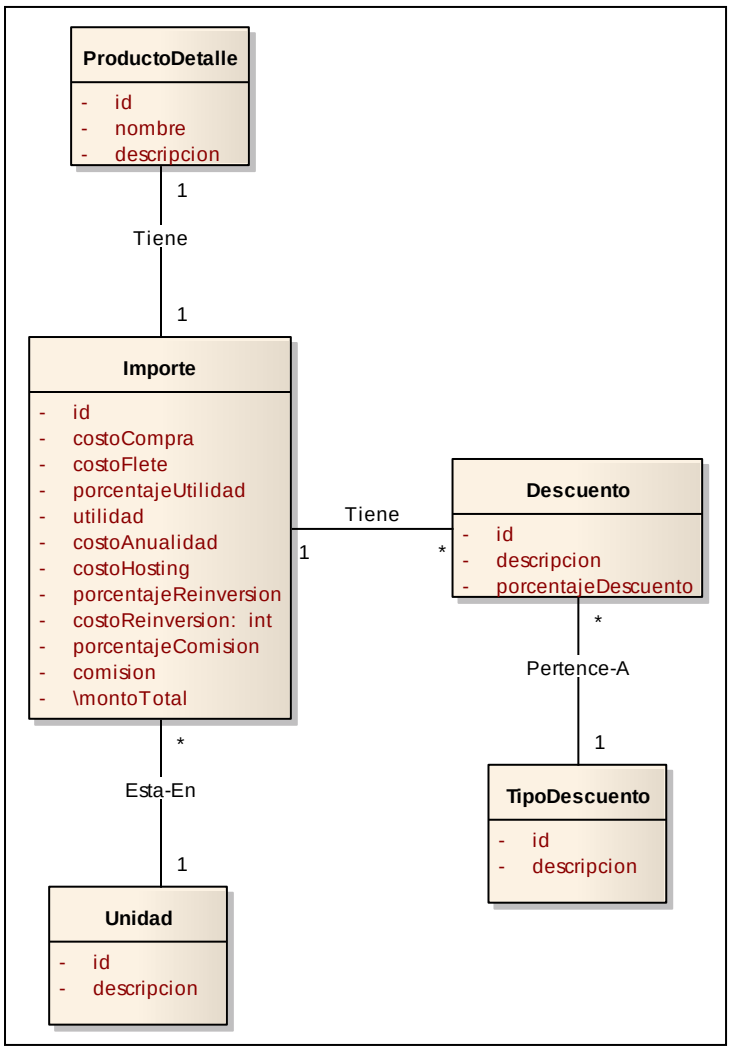

Figura 4.17 Conceptos de Dominio de Producto que utilizan Quantity y un Monto Total derivado

Recordando que se acordó elaborar los modelos de Dominio de los diferentes escenarios de Moda Online, en la Figura 4.18 se presenta el Modelo de Dominio del Pedido, el cual puede apoyar en la comprensión del patrón Quantity.

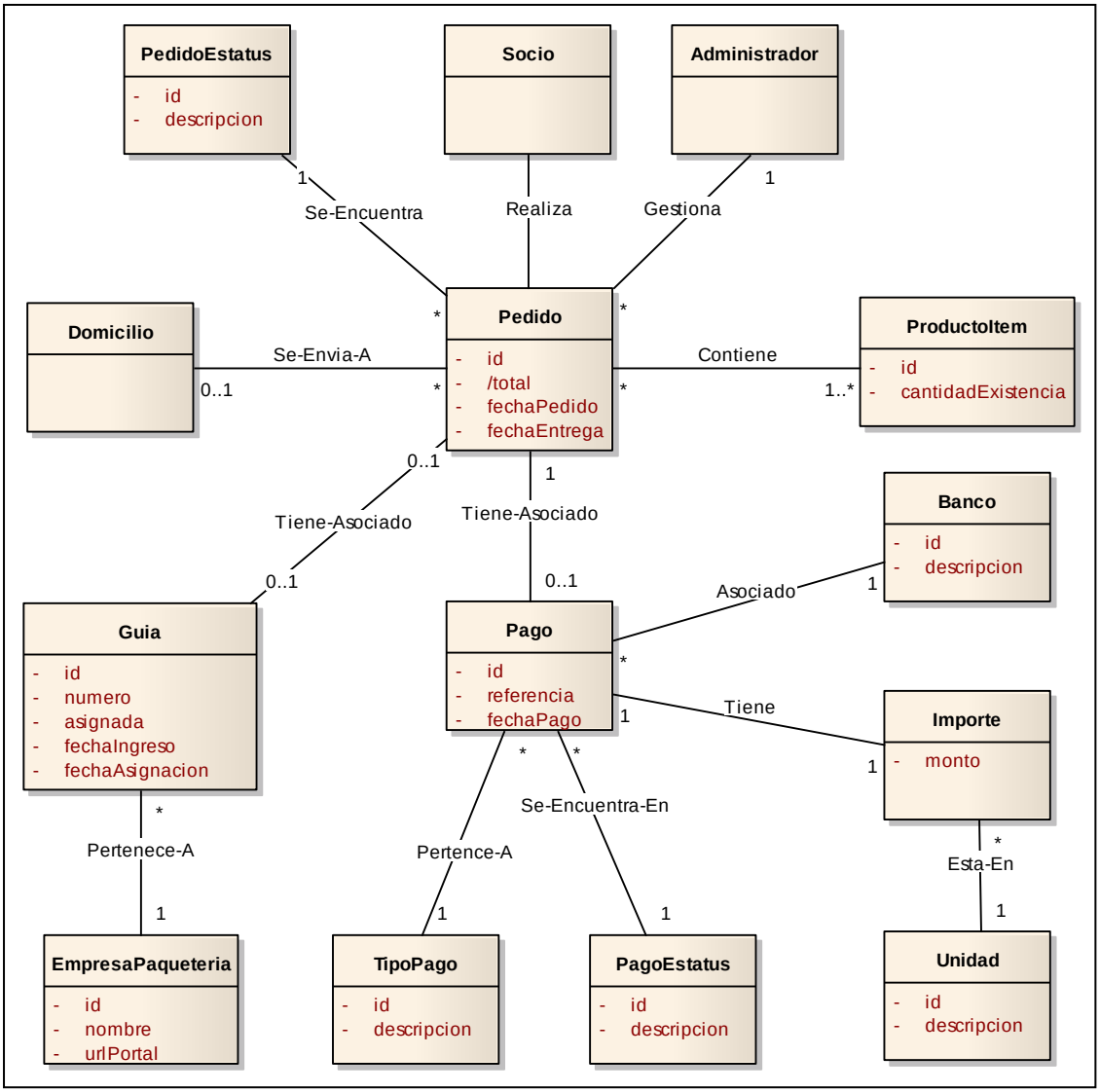

Figura 4.18 Subdominio general de Pedido

4.2.5 Observation

Existen propiedades que se miden cuantitativamente y otras cualitativamente, un ejemplo de lo último es el género en las personas o la talla en las prendas de vestir. No es posible usar atributos para representar estas propiedades, debido a que existe un gran número de opciones.

En inglés el término observation se define como la estimación de la magnitud de un evento o fenómeno. Se emplea regularmente en el ámbito de la medicina, pero es aplicable a otros dominios.

Una forma de implementar Observation es mediante la construcción de una clase que conjunta los posibles valores que puede tomar el resultado de una estimación. Para que el planteamiento tenga éxito es necesario que los valores estén bien definidos y varíen nada o mínimamente.

Si se considera el problema de expresar el género de una persona hay dos opciones: masculino o femenino. Se puede pensar en el género como aquello que se quiere estimar, y masculino y femenino son los valores aplicables. Es posible incorporar un concepto categoría que es similar a Quantity.

En el caso de la Talla de una prenda de vestir es posible establecer la Talla como la propiedad a estimar y S, M, L, etc. como opciones que pueden agruparse en una categoría. Esta solución es totalmente aplicable al domino de Producto de Moda Online, pero es necesario adecuarla el patrón de acuerdo a los requerimientos.

De acuerdo a los requerimientos de Moda Online la forma de estimar la talla de los productos, puede variar, porque se puede emplear un tipo de Talla Regular, Numerada Baja o Numerada Alta, dependiendo del tipo de prenda y del proveedor. Se propone el uso de los conceptos de la Figura 4.19, donde la Talla es la categoría aplicable a la medida de cada prenda y el Tipo de Talla es una categoría que agrupa las clases de Talla.

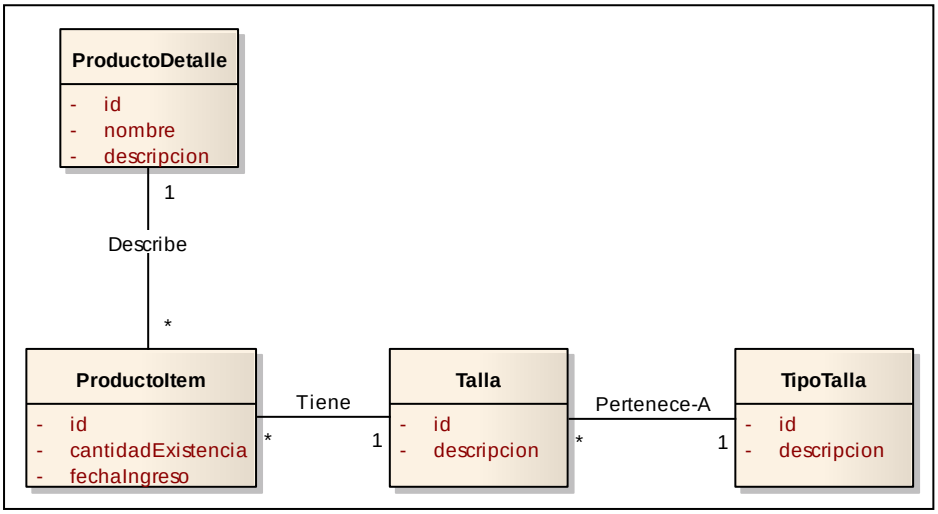

Figura 4.19 Conceptos del Dominio de Producto que emplean Observation

Las estimaciones se hacen sobre las propiedades de un objeto o concepto. En más de una ocasión se encontrará que las características de un dominio se pueden modelar igual que las de otro. Existen dos tipos importantes de estimaciones (Observations), las que tienen que ver con conjuntos de valores discretos, como el color o el género, que se dominan características. Y el otro tipo que se asocia con valores continuos, como la altura en centímetros o el costo en pesos, y que se denominan medidas.

Las medidas son estimaciones en las que se emplean cantidades como valores. Las cantidades representan la relación entre un número y una unidad. Recordar que este vínculo es el argumento para el uso del patrón Quantity. Se resume entonces que Observation aplicado a medidas emplea el patrón Quantity en su definición.

En la Figura 4.20 se encuentra el Modelo de Dominio de Producto utilizado para iniciar el desarrollo de la solución de Moda Online. En este se puede apreciarla forma en que interactúan las clases conceptuales del patrón de Observation y de Quantity con el resto de clases. De hecho

para ser precisos se emplea Observation aplicado a características para la definición de la Talla en un Producto de tipo Item. Se utiliza Observation aplicado a medidas para especificar el Importe del Producto, que a su vez precisa del uso de Quantity para establecer costos. Finalmente se utiliza el patrón de Descripción en relación al Producto, para lo que se define una clase estrictamente conceptual para la especificación de Información asociada al Producto.

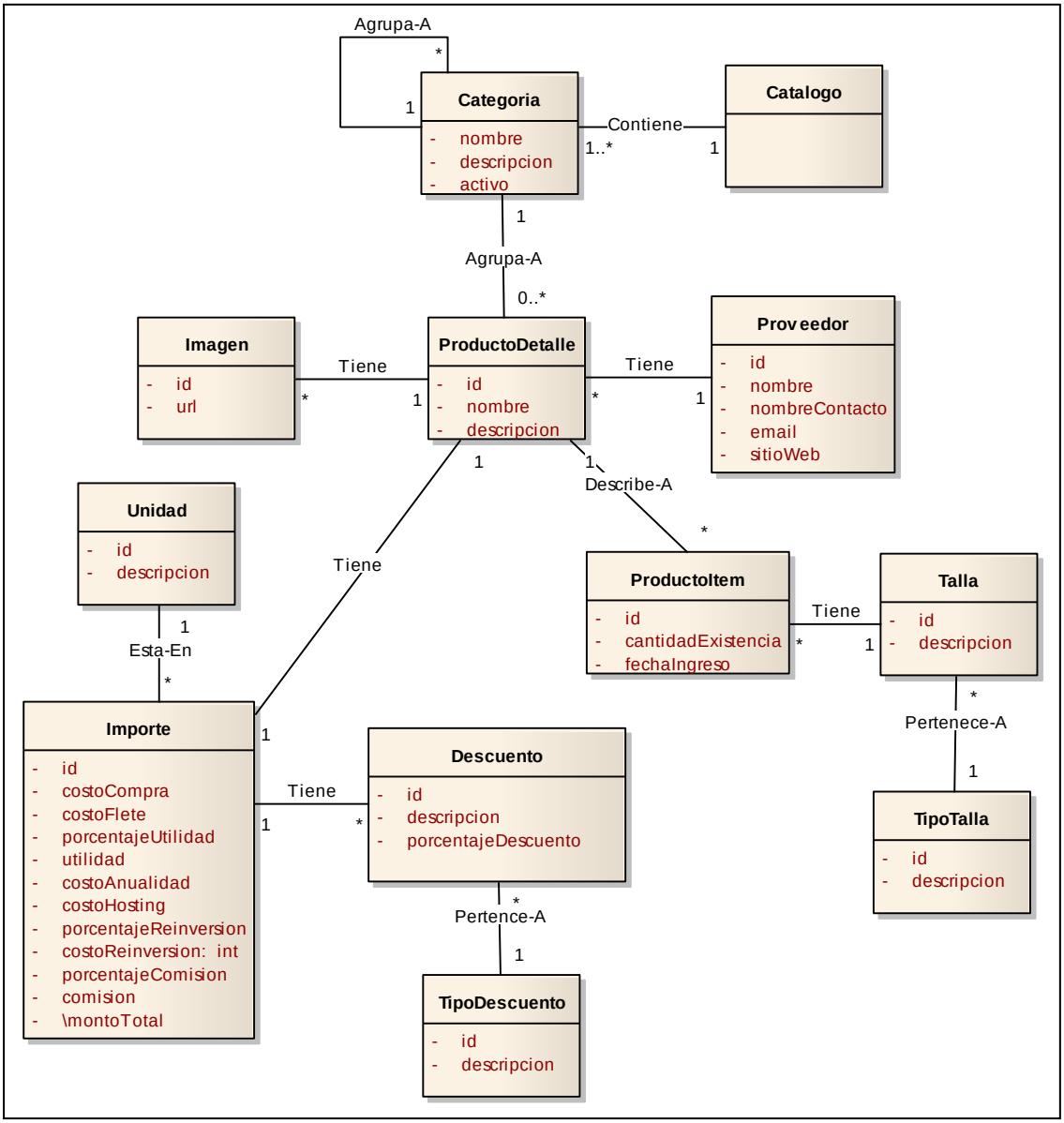

Figura 4.20 Subdominio general de Producto

El patrón de Observation puede emplearse para escenarios más complejos debido a que propone estructuras más elaboradas, pero se recurrirá, como en otras ocasiones a emplearlo únicamente lo necesario.

5 Diseño orientado a Objetos

Durante el análisis se describe el propósito del sistema. Particularmente el resultado de un análisis orientado a objetos es la identificación de los conceptos del dominio. Durante el diseño del sistema se describe la arquitectura, esto es la forma en que se compone el sistema, el flujo general y la gestión de la persistencia. Se define la plataforma de hardware y software.

En el diseño se pone énfasis en la definición de una solución que logre satisfacer los requerimientos. La solución, que es de carácter conceptual se concreta a través de su implementación.

Un diseño orientado a objetos se enfoca en la definición de conceptos de software y la forma que interactúan para satisfacer los requerimientos. Por ejemplo un objeto Carrito tiene asociado un objeto Pedido, y los métodos agregarProducto y eliminarProducto.

¿De qué manera los desarrolladores diseñan objetos? Aquí se presentan tres alternativas:

1. Codificar. Diseñar mientras se codifica. Se inicia la codificación a partir de la construcción de un modelo mental.
2. Dibujar y entonces codificar. Dibujar algunos diagramas UML (Modelar), después continuar con la opción 1 empleando un IDE potente.
3. Dibujar únicamente. Utilizar alguna tecnología que puede generar código a partir de los diagramas.

Para el desarrollo de Moda Online se procuró el uso de técnicas comprendidas en el marco de trabajo de Scrum. Este tipo de técnicas son de carácter ágil y por consiguiente también lo es el modelado.

Un modelado ágil promueve la reducción del tiempo invertido en actividades de diagramado. Recordar que en un enfoque ágil el objetivo del modelado es facilitar el comprensión y la comunicación, no la documentación. Para realizar la tarea se pueden emplear pizarrones, cartulinas, plumones y cámaras para documentar.

El modelado ágil favorece un modelado en grupo y la creación de modelos en paralelo.

5.1 Responsabilidades y un diseño orientado a responsabilidades

Una manera común de diseñar objetos y componentes es a través de la identificación de responsabilidades, roles y colaboraciones. Este planteamiento forma parte de un enfoque denominado Diseño Orientado a Responsabilidades DOR.

Las responsabilidades se definen en razón del comportamiento de un objeto y su rol, y se agrupan de acuerdo a lo que hacen y saben los objetos.

Responsabilidades de acuerdo a lo que hacen:

- Hacer algo por sí mismo, como crear un objeto o hacer un cálculo.
- Instanciar acciones de otros objetos.
- Controlar y coordinar actividades de otros objetos.

Responsabilidades de acuerdo a lo que saben:

- Saber sobre datos privados.
- Saber sobre objetos relacionados
- Saber sobre datos derivados o cálculos

Las responsabilidades se asignan a las clases durante el diseño de objetos. Por ejemplo, que Carrito de Compras es el responsable de agregar Productos.

La granularidad de una responsabilidad influye en el nivel de responsabilidad. Una gran responsabilidad comprende cientos de clases y métodos, una responsabilidad mínima implica un solo método.

Las responsabilidades no se definen únicamente en razón de los métodos, pero contribuyen en su consolidación.

En un Diseño Orientado a Responsabilidades también interviene la colaboración. Las responsabilidades se implementan mediante métodos, que bien pueden colaborar con otros o no. Por ejemplo, para obtener el Total del carrito se puede invocar el método getPrecioTotalPedidodelPedido asociado al Carrito, que a su vez invoca el método getPrecioProducto de cada Producto.

GRASP le da un nombre y una descripción a algunos de los principios básicos útiles para la asignación de responsabilidades.

5.2 ¿Qué son los patrones?

Un grupo de especialistas en el desarrollo orientado a objetos ha construido un conjunto de principios y soluciones que sirven como guía en la producción de software. Si estos principios se presentan en un formato que describa tanto el problema como la solución califican como patrones. Ejemplo

Nombre del Patrón:	**Information Expert**
Problema:	¿Cuál es el principio básico para la asignación de responsabilidades a objetos?
Solución:	Asigna la responsabilidad a la clase que tiene la información necesaria para soportarla.

En el diseño OO, un patrón es una descripción con nombre de un problema y su correspondiente solución aplicable a un contexto nuevo. Un patrón indica la forma que su solución puede aplicarse a escenarios diferentes, tomando en cuenta los factores que intervienen.

De manera estricta el término patrón se refiere a un evento que se repite continuamente. Un patrón de diseño no introduce un concepto nuevo, expresa conocimiento que ha sido plenamente probado.

El concepto de patrones con nombre en software surge a mediados de la década de los 80, pero experimenta un gran auge en 1994 con la publicación de libro Patrones de Diseño. El libro describe 23 patrones para el diseño OO, que hoy se conocen como patrones de diseño GoF por sus siglas en inglés (Gang of Four).

Entonces ¿GRASP es un grupo de patrones o principios? La respuesta se encuentra en el libro de Patrones de GoF y se cita a continuación:

Lo que representa un patrón para una persona bien puede representar un bloque de directivas de construcción para otra.

En este sentido, es mejor concentrarse en el valor práctico del uso de patrones como auxiliar en el nombrado y presentación de conceptos.

5.3 Emplear GRASP para el diseño de objetos

Los Patrones de Software para la Asignación General de Responsabilidades (GRASP, por sus siglas en inglés) son un grupo de principios que apoyan el diseño de software orientado a objetos.

Entender y aplicar los conceptos comprendidos en GRASP durante el modelado o la codificación es fundamental para la integración del enfoque orientado a objetos a nuestro diseño.

Los Patrones GRASP son nueve:

Creator	Controller	Pure Fabrication
Information Expert	High Cohesion	Indirection
Low Coupling	Polymorphism	Protected Variations

La forma en que se aplican los patrones de diseño es más estricta, si bien un patrón puede adecuarse para integrarse a un nuevo contexto, es mejor no forzar su implementación, tratar de hacerlo puede derivar en un mal diseño y posiblemente un re trabajo.

De acuerdo al Dominio de Moda Online y la tecnología propuesta para el desarrollo de la solución (J2EE) se empleo una sección de patrones GRASP, que a continuación se definen y describen prácticamente.

5.3.1 Creator

La creación de un objeto es una actividad recurrente en una aplicación orientada a objetos. Si la actividad se lleva a cabo satisfactoriamente, el diseño es más claro y puede soportar bajo acoplamiento, encapsulación y re usabilidad.

Problema

¿Quién es responsable de crear la instancia de una clase?

Solución

Asignar a la clase B la tarea de crear una instancia de A si se cumple una de las siguientes sentencias:

- B contiene A
- B registra a A
- B usa notablemente a A
- B tiene la información de inicialización de A
- B es un creador de objetos A.

En el Dominio de Moda Online es necesario contar con un Carrito de Compras que se encargue de la gestión de productos que el Usuario desea adquirir. Al analizar el problema se identificó que tanto el carrito como el Pedido contienen Productos de tipo Item, por lo que se decidió que el Carrito tenga asociado un Pedido con un estatus "en proceso" mientras no se formalice la compra. En el diagrama de la Figura5.1 se puede observar la forma en que interactúan las clases involucradas en la creación del Carrito y en la anexión de nuevos productos al mismo.

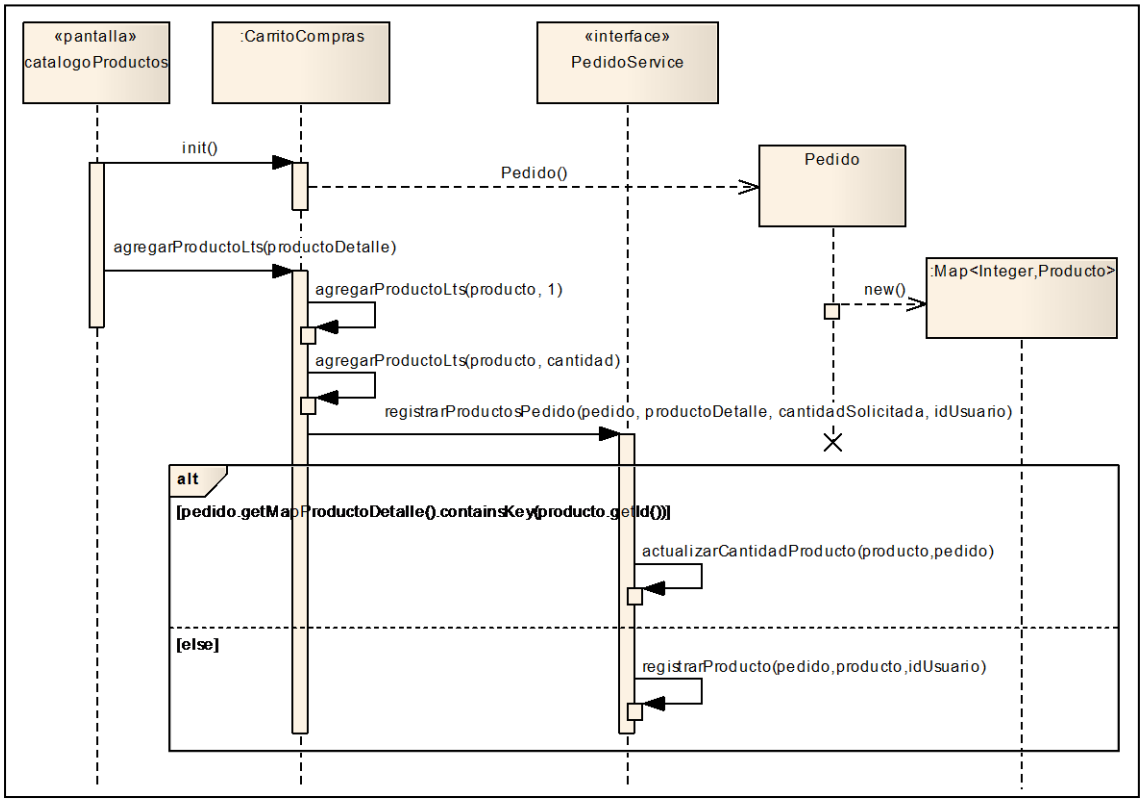

Figura 5.1 Diagrama de Secuencia que denota el uso de Creator en el Carrito de Compras

En el diseño para Moda Online se empleó recurrentemente el recurso de introducir un método init en la composición de las clases. Su funcionamiento es similar al de un constructor, pero es inherente a Spring y se ejecuta inmediatamente después de la inyección de dependencias. El método se invoca antes de que la clase se ponga en operación y permite realizar cualquier inicialización.

La clase que se utiliza para construir objetos de tipo carrito de compras tiene un scope de Sesión y un método init que crea una instancia de tipo Pedido, que a su vez crea un instancia de un mapa para objetos de tipo Producto (ítem) durante la ejecución de su constructor. De esta manera un usuario dispone de un carrito de compras una vez que hace una solicitud HTTP al aplicativo y se le asigna una sesión.

El resultado de la aplicación del patrón se puede ver en el diagrama de clases de la Figura 5.2. Es importante destacar que las clases comprendidas tanto este diagrama como en los próximos, carecen de algunos métodos, por ejemplo las duplas de getters y setters, y otros métodos no trascienden para la demostración del uso de los de diseño GRASP.

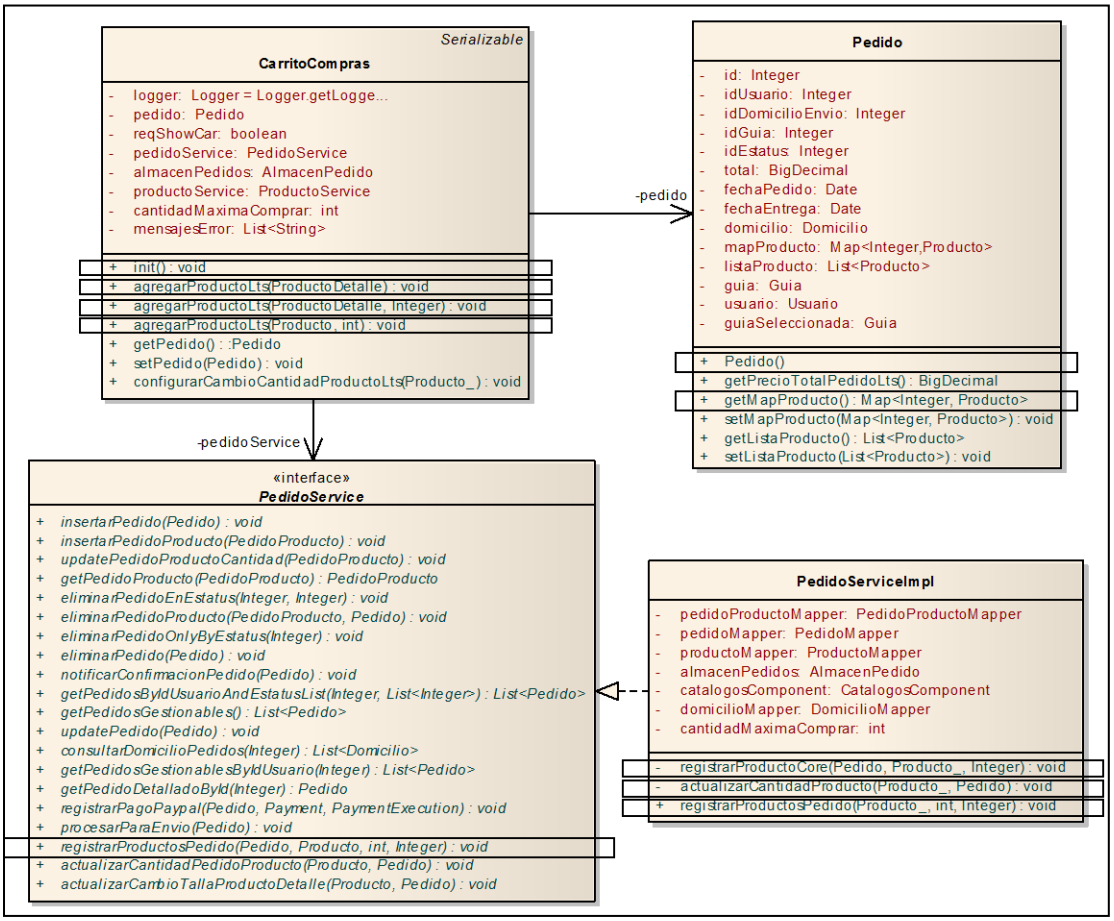

Figura 5.2 Diagrama de Clases involucradas en el proceso de compra o adición de productos al Carrito

5.3.2 Information Expert

En un modelo de diseño se pueden definir cientos de clases de software, pero para que una aplicación cumpla su objetivo debe soportar cientos o miles de responsabilidades. Durante el diseño de objetos, se descubren relaciones entre los objetos, lo que hace necesario decidir acerca de quién es responsable de qué. Si se asigna apropiadamente, el sistema es claro, puede crecer y sus componentes son susceptibles a reutilización en otras aplicaciones.

Problema

¿Cuál es el principio general para la asignación de responsabilidades?

Solución

Asignar una responsabilidad a la clase que tiene la información necesaria para solventarla.

En el escenario del cálculo del monto o precio total del Carrito, la operación se hace mediante el cálculo de la sumatoria de los importes de cada producto ingresado al carrito. Se sabe que el Carrito tiene un Pedido asociado, el cual dispone de un mapa donde se van anexando los productos (Items) que el Usuario desea adquirir. De aquí que se deje la tarea de calcular el costo total del Carrito al Pedido. La Figura 5.3 expresa la interacción surgida entre las clases involucradas en el cálculo y recuperación del costo total de Pedido.

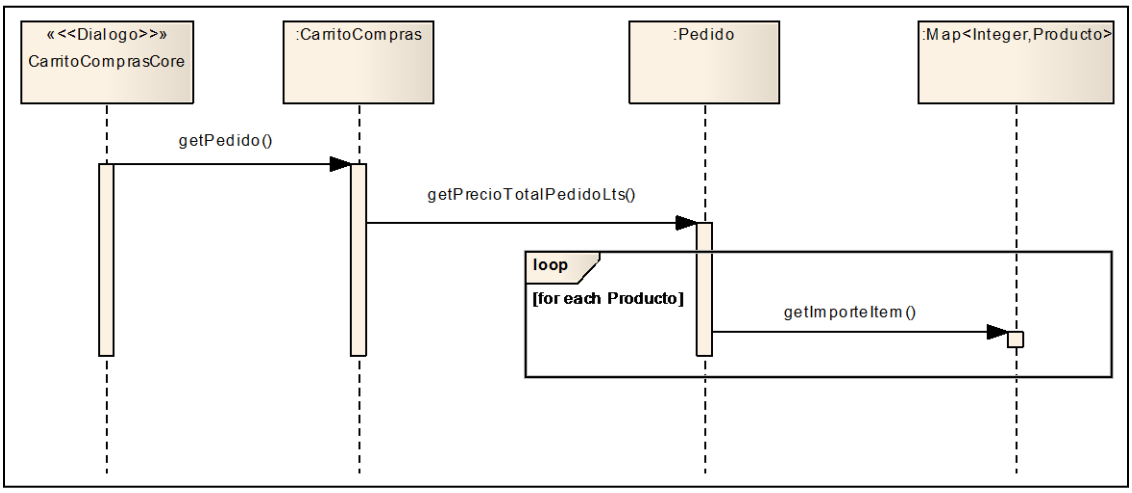

Figura 5.3 Diagrama de Secuencia de cálculo del total del Pedido empleando Information Expert

Es claro que si el Pedido alberga información asociada a los productos que el usuario busca adquirir, es correcto asignarle la responsabilidad del cálculo del monto total del Pedido y por consiguiente del monto del Carrito de compras. En el diagrama de clases de la Figura 5.4 se observan tanto las clases como los métodos originados a partir del análisis del procedimiento.

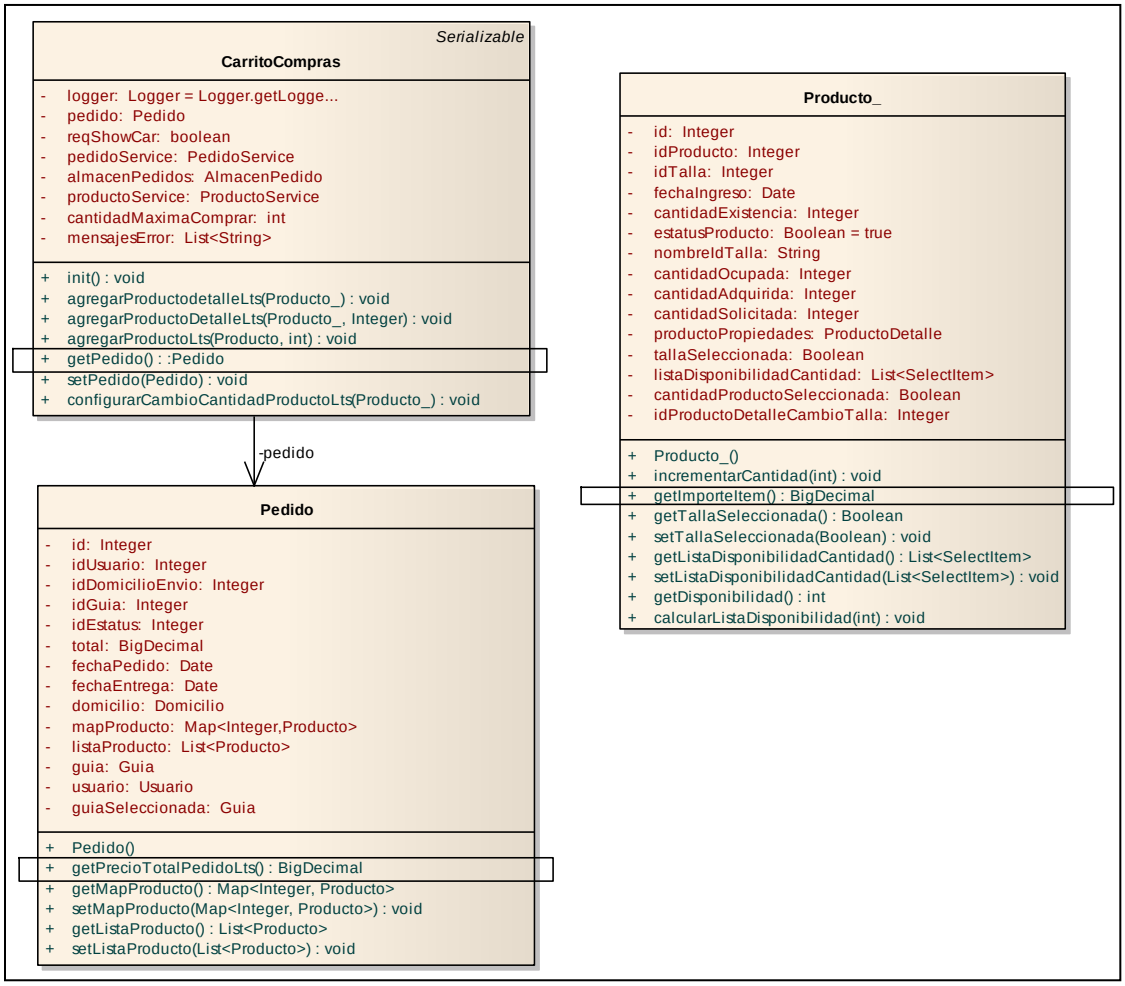

Figura 5.4 Diagrama de Clases involucradas en el cálculo del costo total del Pedido

5.3.3 Low Coupling

El acoplamiento se define en relación al grado de integración de un elemento, la cantidad de información que detenta o el nivel de dependencia. Un elemento con bajo acoplamiento no depende de otros elementos en exceso. Se entiende por elementos a las clases, subsistemas, sistemas, etc.

Una clase con alto acoplamiento necesita de otras clases para solventar sus responsabilidades. En este tipo de escenario surgen los siguientes problemas:

- Las modificaciones en otras clases generan cambios localmente.
- Es difícil entender una clase por sí sola.
- Es difícil reutilizar debido a que se necesita introducir clases adicionales.

Problema

¿De qué manera se reduce la dependencia y el impacto producido por el cambio?

Solución

Asignar responsabilidades apropiadamente para que prevalezca un bajo acoplamiento. Emplear este principio para evaluar alternativas.

El uso de Information Expert promueve el bajo acoplamiento, porque apoya en la correcta asignación de responsabilidades. Una responsabilidad se asigna a aquel objeto que cuenta con la información necesaria para resolverla.

Al delegar una responsabilidad arbitrariamente, se incrementa el acoplamiento general, como consecuencia de una colaboración informativa excesiva.

El uso de Information Expert es recurrente en la solución de Moda Online, por ejemplo en el escenario de cambio de cantidad de un Producto contenido en el carrito. Un Usuario puede añadir o incrementar productos a su carrito en diferentes instancias, desde el catalogo principal de la aplicación, desde el detalle del Producto o desde el carrito. Si se agrega un Producto y es diferente a los existentes, entonces se ejecuta un registro, de lo contrario un incremento. Pero no es todo, para poder habilitar el cambio de cantidad es necesario determinar la disponibilidad general del Producto considerando el número de elementos atrapados en los carritos de los Usuarios concurrentes, así como el límite de productos a adquirir definido por el cliente. El diagrama de secuencia de la Figura 5.5 muestra el comportamiento descrito.

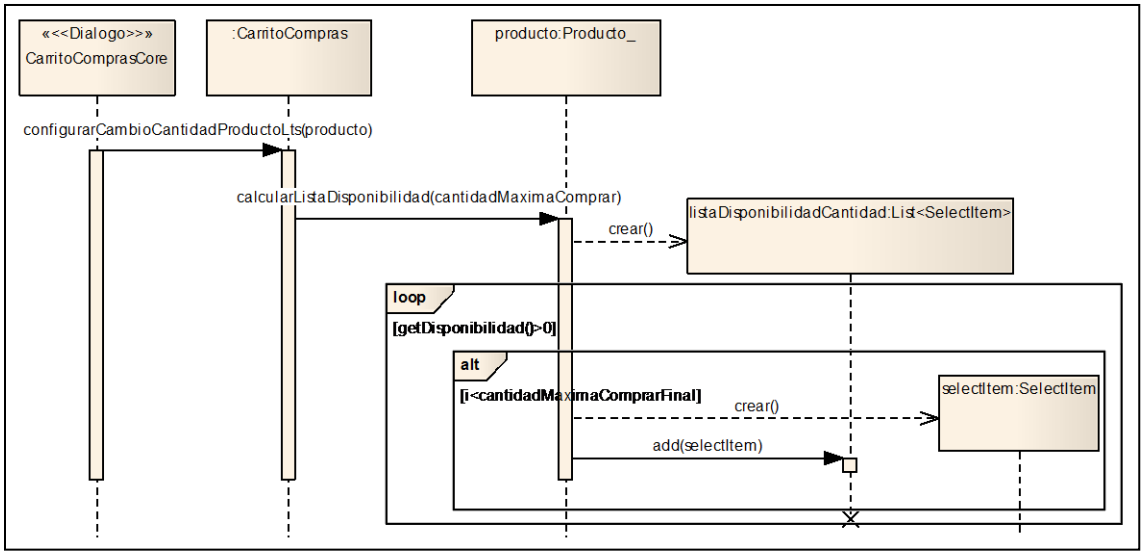

Figura 5.5 Diagrama de Secuencia de cambio de cantidad de un Producto usando Information Expert

En el cálculo de la disponibilidad de un Producto participan varias clases, pero la que tiene mayor injerencia es la clase Producto, porque el resto únicamente invoca la funcionalidad, lo que a plena vista promueve una alta cohesión y un bajo acoplamiento. En el diagrama de clases de la Figura 5.6 se pueden ver las clases y métodos que surgen del uso de Information Expert y consecuentemente de Low Coupling.

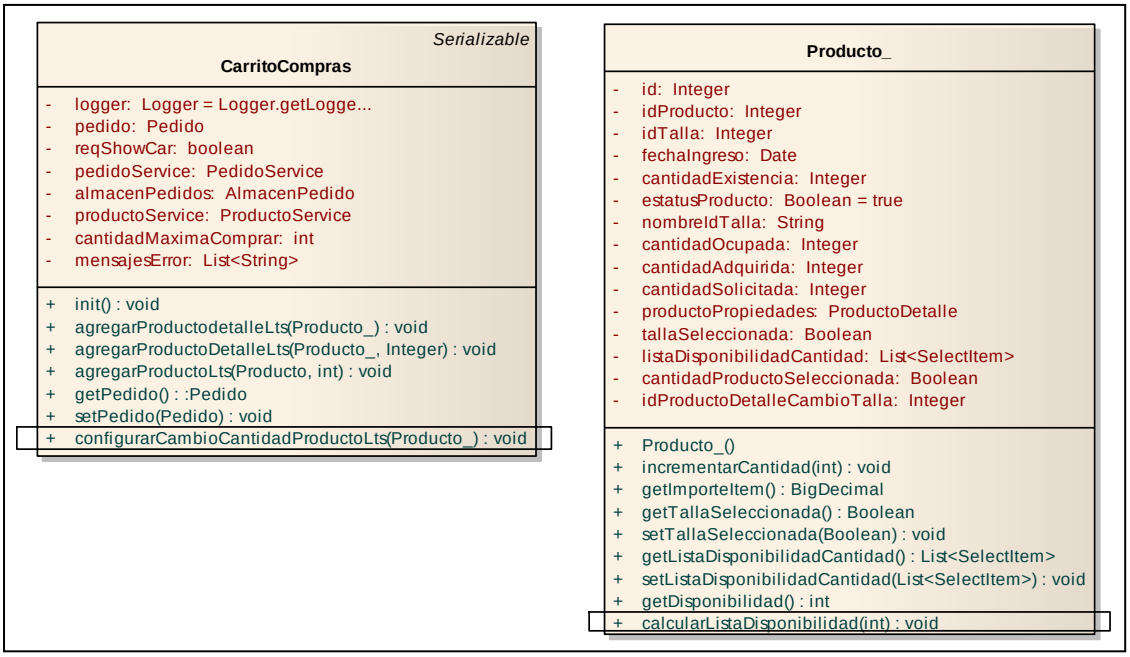

Figura 5.6 Diagrama de Clases involucradas en el evento de cambio de Cantidad de un Productos en el Carrito

5.3.4 Controller

El controlador proporciona acceso a la funcionalidad de la aplicación, interpreta las entradas del Usuario y las convierte en un modelo.

Problema

¿Qué objeto es el primero en interactuar con la Interfaz de Usuario, que recibe y coordina operaciones en la aplicación?

Solución

Asignar la responsabilidad a la clase con las siguientes características:

- Representa el sistema en general, es un objeto raíz, es un mecanismo que se ejecuta en el software.
- Representa el escenario de un caso de uso durante el cual se ejecuta un evento, nombrado recurrentemente como <NombreCasoUso>Handler, <NombreCasoUso> Coordinador o <NombreCasoUso>Controlador.

Las ventanas, vistas y documentos no representan clases en esta lista. Estas clases no satisfacen operaciones asociadas a eventos, en su lugar las delegan a los controladores.

Para el aplicativo de Moda Online se integraron un grupo de frameworks que conforman la arquitectura fundamental de la solución, Spring y JSF son un par de ellos. Ambas tecnologías se basan en una arquitectura MVC (Modelo Vista Controlador por sus siglas en Ingles). Tanto en una como en otra el papel de controlador (Front Controller) lo representa un Servlet, en Spring el DispatcherServlet y en JSF el FacesServlet. Debido a que la arquitectura utiliza JSF como motor

para la Vista y a Spring como integrador del resto de Frameworks, el controlador frontal que se ejecuta es FacesServlet.

El controlador frontal de JSF opera como punto de acceso para el Framework de JSF y como gestor del ciclo de vida de las solicitudes.

El ciclo de vida de una aplicación JSF se compone de seis fases:

- Fase de restauración de la vista.
- Fase de aplicación de valores.
- Fase de procesamiento de validaciones
- Fase de actualización de valores del modelo
- Fase de ejecución de la aplicación.
- Fase de renderizado de la respuesta

En el entendido de que JSF cuenta con su propio controlador y este participa en todas las fases del ciclo de vida de JSF, no es necesario implementar otro. Para entender la forma en que opera se abordará el mecanismo de funcionamiento de tres fases fundamentales, la de procesamiento de validaciones, actualización de los valores del modelo y la de ejecución de la aplicación.

En la fase de **procesamiento de validaciones** se toman los valores capturados en la fase de aplicación valores y se realizan las siguientes acciones:

- Conversión de valores capturados de String al tipo correspondiente.
- Análisis de validez de los valores capturados.
- Si los datos capturados son diferentes a los que contiene el modelo de presentación se generan eventos de cambio de valor (value change events).
- Establecer valor local al componente

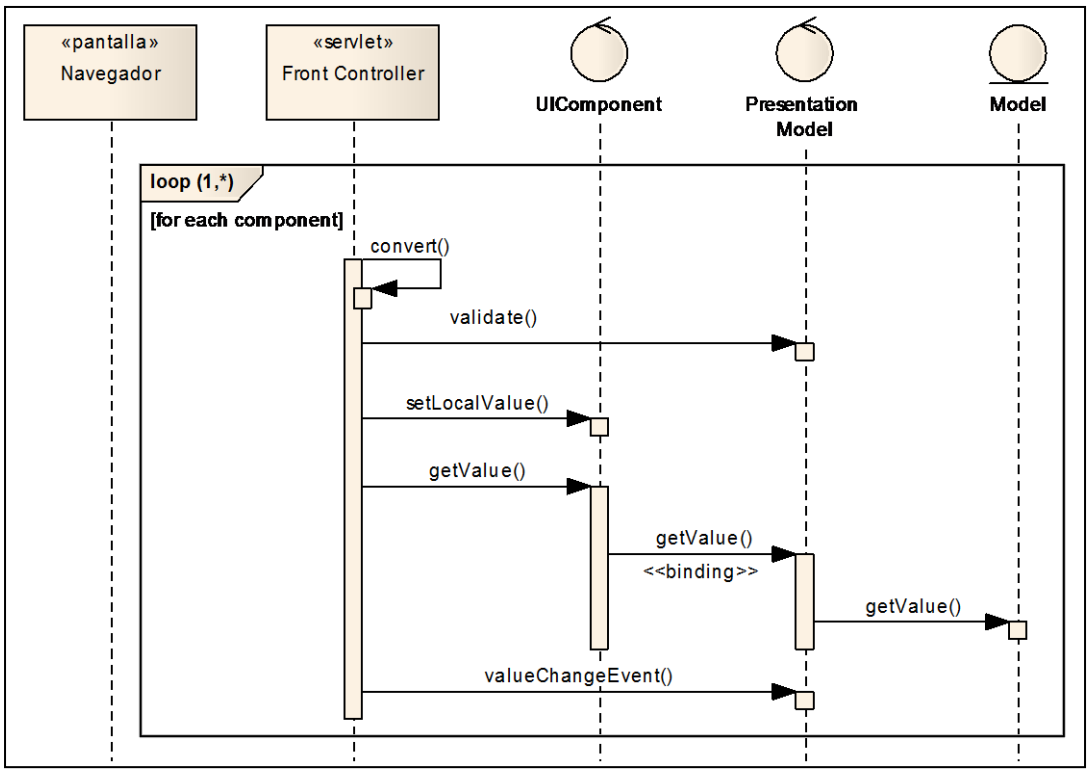

Figura 5.7 Diagrama de Secuencia de la Fase de Procesamiento de Validaciones de JSF

En la fase de **actualización del modelo** se trasladan los datos capturados hacia el Modelo de Presentación. El Modelo de Presentación recupera el estado y el comportamiento de la vista y los coloca en una clase modelo que encapsula el acceso al modelo de dominio.

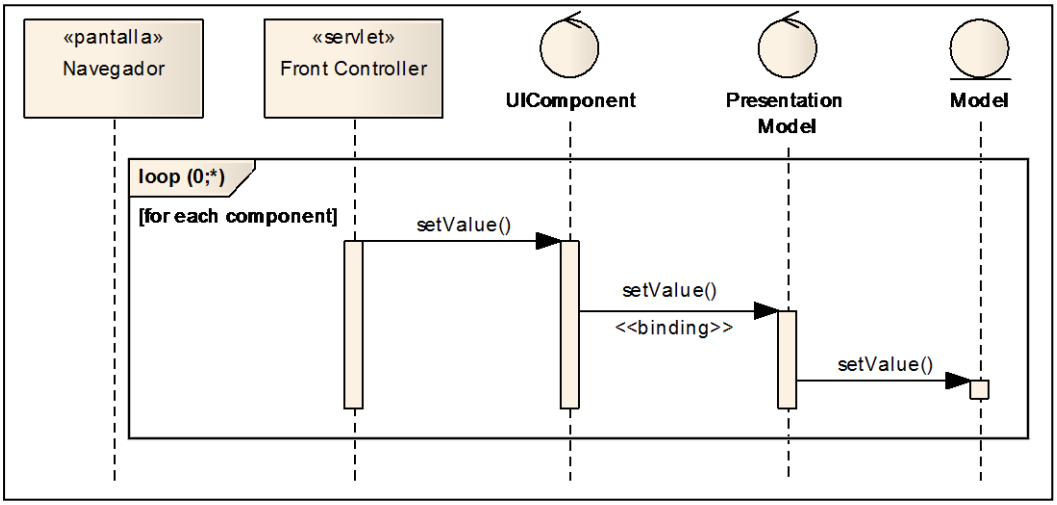

Figura 5.8 Diagrama de Secuencia de la Fase de Actualización de valores del Modelo de JSF

El Controlador Frontal traslada el valor local del componente, establecido durante la conversión de valores, hacia el Modelo de Presentación mediante una vinculación de valor (value binding). Inmediatamente después se limpia el valor local.

La fase de **ejecución de la aplicación** transfiere el action event al Modelo de Presentación y se produce la ejecución de lógica de negocio. Es responsabilidad del Modelo de Presentación interactuar con el Modelo de Dominio y modificar su estado en razón de las acciones que realiza el usuario.

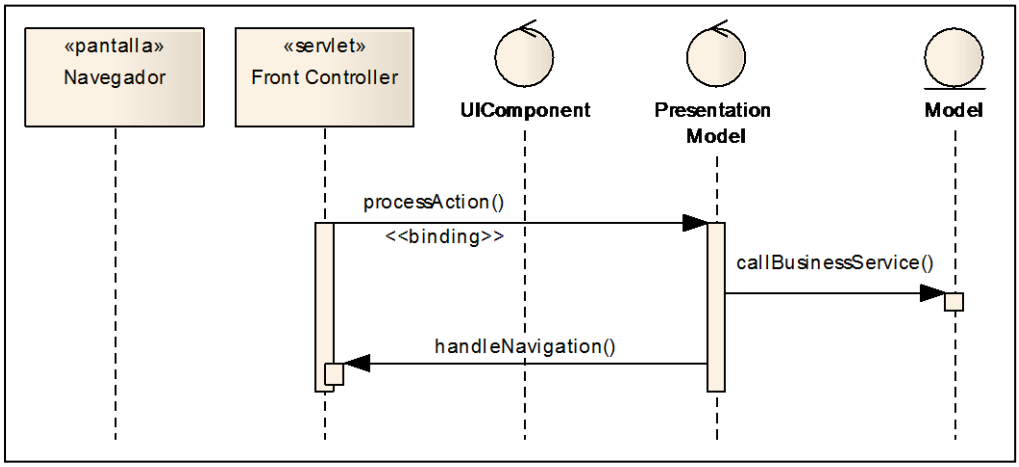

Figura 5.9 Diagrama de Secuencia de la Fase de Ejecución de la Aplicación de JSF

Si existen validaciones complejas en el Modelo de Presentación que no fueron ejecutadas durante la fase de validaciones se ejecutan al principio de la fase (ej. Validación de Contraseña actual). Es también durante esta fase que se determina el rumbo de la navegación.

5.3.5 High Cohesion

En términos de diseño orientado a objetos, se entiende como cohesión a la medida en que las responsabilidades de un elemento se encuentran vinculadas.

Una clase con baja cohesión tiende a realizar muchas tareas que no tienen relación o acopia simplemente demasiado trabajo. Una clase con estas características sufre de lo siguiente:

- Es difícil de entender
- Es difícil de reutilizar
- Es difícil de mantener
- Sufre cambios frecuentemente

Problema

¿Cómo hacer que nuestros objetos prevalezcan claros, administrables y enfocados?

Solución

Asignar responsabilidades de manera que prevalezca una alta cohesión. Usar este principio para analizar alternativas.

Durante el análisis del dominio de la aplicación de Moda Online se identificó la existencia de varios conceptos que pueden convertirse en tablas de base datos y que figuran como catálogos, por ejemplo Tipo de Talla, Talla, Banco, Grupo de Usuario, Estatus de Pago, Entidades, entre otros.

Si se considera que el número de usuarios concurrentes de la aplicación podría crecer con el tiempo, se debe crear un mecanismo que minimice el número de accesos a base de datos, como resultado de la consulta de los catálogos.

Una alternativa es introducir un objeto que realice la tarea de recuperar todos los catálogos de la base de datos durante el despliegue de la aplicación y los conserve en memoria. El objeto es visible y accesible para todos los usuarios de la aplicación y solo existe una instancia de este. La descripción denota el uso del patrón High Cohesion, pero también del patrón Singleton de GoF. Al emplear un objeto que tenga únicamente tareas vinculadas a la recuperación y entrega de datos relacionados a catálogos se favorece el uso del patrón High Cohesion. Pero para poder producir una solución de este tipo es necesario apoyarse de un patrón de diseño que ilustre la forma de implementar una clase de la cual se genera una sola instancia y que sirva de punto de acceso para todos los objetos.

El objeto en cuestión opera como un componente dedicado a la exposición de catálogos. Por esta razón se le asociará el nombre de CatalogosComponent. En la Figura 5.10 se muestra un diagrama de secuencia del escenario de Visualización del Detalle del Producto donde se carga un Combo con los tipos tallas disponibles del Producto seleccionado.

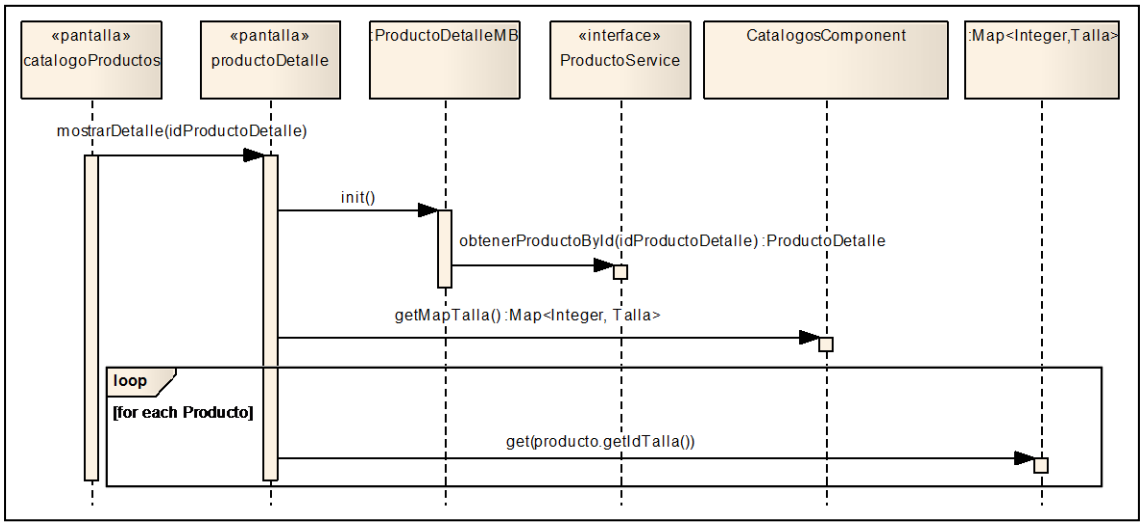

Figura 5.10 Diagrama de Secuencia de Visualización del Detalle del Producto

Cuando se accede a la pantalla de Detalle del Producto, se hace una solicitud HTTP y durante el ciclo de vida de JSF se recuperan los datos correspondientes a las Tallas disponibles del Producto, se trata de un mapa con Objetos de Tipo Talla, en donde la llave es el identificador del Tipo de Talla. En el momento que se hace la solicitud del catálogo este ya se encuentra cargado en el mapa de Tallas, por lo que no es necesario realizar una consulta a la base de datos.

Averiguar la forma en que funciona la recuperación de datos del catálogo de Tipo Talla durante la habilitación del Detalle del Producto, permite definir las clases y métodos del diagrama de clases de la Figura 5.11.

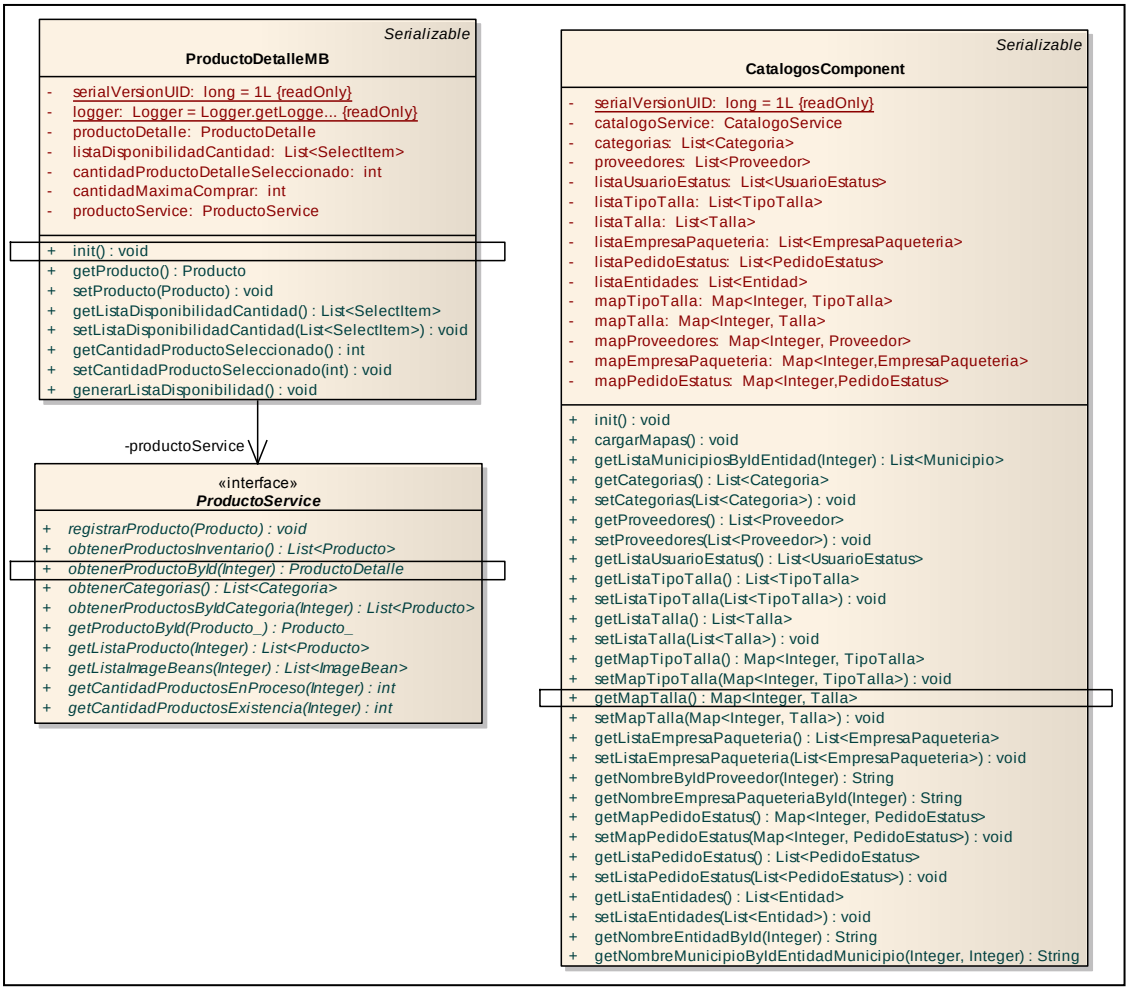

Figura 5.11 Diagrama de Clases involucradas en la recuperación de Tallas durante la visualización del Detalle del Producto

Otro de los escenarios en que se emplea el componente de Catálogos es durante el registro de un Domicilio, por ejemplo el del Proveedor de productos. Un Usuario Administrador puede crear o editar registros de proveedores. Durante la ejecución de la tarea es posible ingresar un Domicilio, el cual tiene asociado una Entidad y un Municipio, datos que se obtienen de los catálogos de la base datos. En la Figura 5.12 se observa el modo en que se realiza la recuperación de los datos de ambos Catálogos.

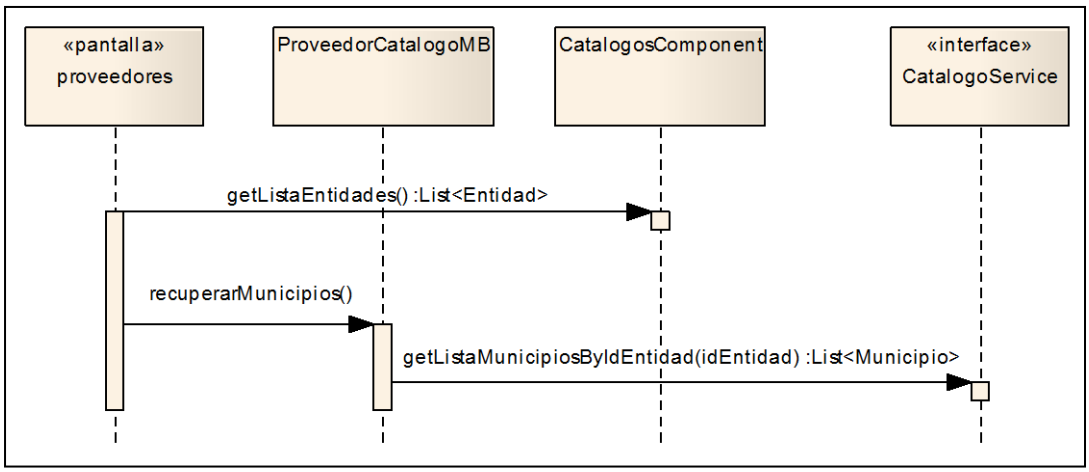

Figura 5.12 Diagrama de Secuencia de recuperación de catálogos durante la gestión de proveedores

En el diagramase observa que el componente de catálogos atiende la solicitud de la Lista de entidades, pero no sucede lo mismo en el caso de la Lista de municipios. La razón es que el catálogo de entidades es estático, no cambia en razón de ningún parámetro, por lo que se puede cargar en una sola ocasión durante la inicialización de CatalogosComponent. Por su parte el Catálogo de municipios es extenso y se calcula a partir de la Entidad seleccionada por el Usuario, de tal manera que no es fijo y no es prudente recuperarlo de principio y mantenerlo en memoria.

A partir de lo expuesto en el diagrama de secuencia de gestión de proveedores fue posible identificar las clases y métodos que participan en el escenario y que aparecen el diagrama de clases de la Figura 5.13.

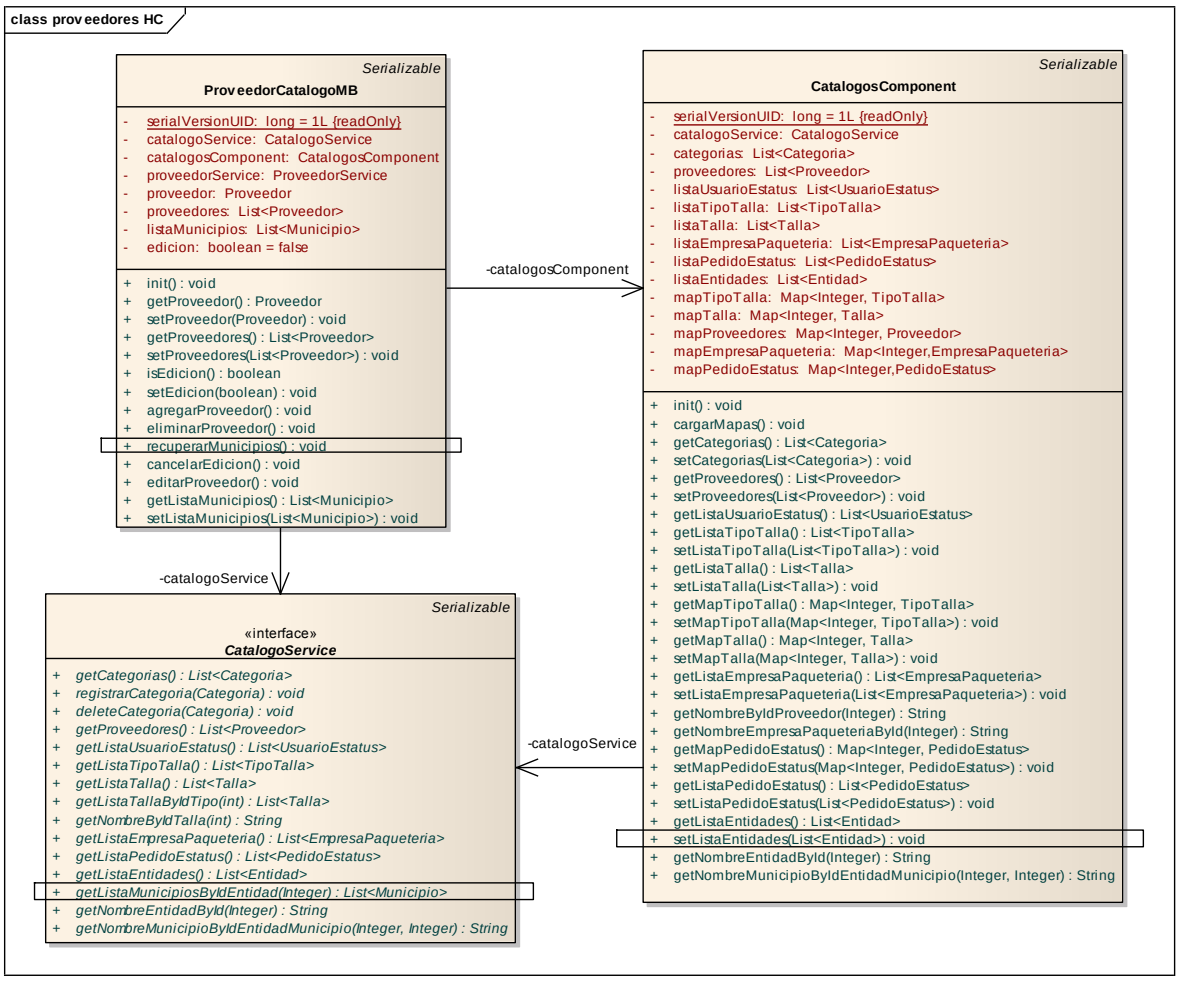

Figura 5.13 Diagrama de Clases involucradas en la recuperación de catálogos durante la gestión de proveedores.

Conclusiones

El estudio y exploración de los patrones permite reconocer el amplio rango de aplicación al que están sujetos. Existen patrones tanto para el análisis como para el diseño e implementación de proyectos de software, y tentativamente para el resto de fases del desarrollo. Los patrones contribuyen en la mejora del software producido, estableciendo estrategias de solución comprobadas para problemas recurrentes.

Los patrones no se crean, surgen del trabajo continuo y la experiencia en proyectos reales. El analista o desarrollador puede enfrentarse a un problema y solucionarlo sin saber que está empleando un patrón. La ventaja de emplear patrones es que no es necesario invertir tiempo valioso en resolver un problema para el cual alguien ya encontró una solución que comprobó y documentó.

Los patrones aparecen en varias áreas del desarrollo, los orientados a análisis por ejemplo, apoyan en la tarea de identificación de conceptos del dominio de un problema, a través del uso estructuras definidas para el modelado de escenarios comunes. Los patrones de diseño por su parte apoyan en la definición de objetos de software, sus responsabilidades y dependencias. Los patrones de diseño se presentan en un formato que incluye el nombre del patrón, la descripción del problema y su correspondiente solución.

El uso de patrones GRASP como auxiliar en la definición de conceptos, responsabilidades y relaciones produce excelentes resultados. Adicionalmente los patrones Creator e Information Expert refuerzan la aplicación de los patrones Low Coupling y High Cohesion. El uso de estos últimos promueve la alta cohesión y el bajo acoplamiento, principios que obedecen los patrones de GoF.

Como ya se mencionó, los escenarios susceptibles al uso de patrones son recurrentes. En la construcción de la aplicación, además de los patrones GRASP se emplearon patrones de GoF, el de Singleton para ser exactos. En la construcción de la arquitectura se empleó un patrón MVC orientado a aplicaciones Web. Y de acuerdo a la plataforma de implementación (J2EE) se emplearon los patrones de Front Controller, Data Access Object y Transfer Object.

De acuerdo al trabajo realizado se infiere que los patrones de análisis son completamente aplicables al desarrollo Web debido a que se enfocan en la identificación de conceptos de dominio, lo que es inherente al análisis y ajeno al entorno de implementación. Aunque el diseño de software por definición es independiente de la plataforma de implementación, no sucede así cuando se trata de un entorno Web. Una solución conceptual de este tipo debe tomar en cuenta conceptos relativos al entorno, la elección de la arquitectura por ejemplo, afecta la definición de los conceptos de software (Un controlador en una arquitectura Web es distinto al de una arquitectura de escritorio).

El uso de patrones en las diferentes fases del desarrollo incrementa la calidad del producto en varios aspectos. El diseño conceptual de las clases y la arquitectura son más digeribles y soportan cambios, el entregable por otra parte, se acerca más a lo que el cliente y el usuario desean.

Bibliografía y Referencias

[1] Roman PichlerAgile Product Management with Scrum. Addison Wesley 2010

[2] Ken Schwaber Scrum Guide. Scrum Alliance 2009

[3] Michael James Scrum DZonerefcardz #50

[4] GertiKappel, Birgit Proll, Siegfried Reich,WernerRetschitzegger
Web Engineering: The Discipline of Systematic Development of Web Applications. John Wiley & Sons, Ltd. 2006

[5] Gustavo Rossi, Oscar Pastor, Daniel Schwabe, Luis Olsina
Web Engineering: Modelling and Implementing Web Applications.
Springer-Verlag London2008

[6] Kim Hamilton & Russell MilesLearning UML 2.0 O'Reilly 2006

[7] Craig Larman
Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition. Addison Wesley 2004

[8] Craig Larman, Bas Vodde
Scaling Lean &Agile Development: Thinking and Organizational Tools for Large-Scale Scrum. Addison Wesley 2008

[9] Martin FowlerAnalysis Patterns: Reusable Object Models. Addison Wesley 2000

[10] Bernd Bruegge& Allen H. Dutoit
Object-Oriented Software Engineering Using UML, Patterns, and Java Third Edition. Prentice Hall 2010

[11] Joseph W. Yoder, Federico Balaguer, Ralph Johnson
From Analysis to Design of the Observation Pattern. 2000

[12] Xiaohong Yuan, Eduardo B. Fernandez
Patterns for Business-to-Consumer E-Commerce Applications

[13] JSF Quick Guide
http://www.tutorialspoint.com/jsf/jsf_quick_guide.htm

[14] Steven L. Murray
Designing JSF Applications: a Storyboard Approach — Part 2
http://www.jsfcentral.com/articles/storyboard_2.html

[15] Web MVC Framework
http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html