

# **Modelo Integral para la Construcción de Aplicaciones de Comercio Electrónico – MICACE**

## **ANEXO A MARCO CONCEPTUAL**



**Muller Alirio Rosero Palacios  
Holmes Giraldo Zambrano Melo**

Director: Ing. Mario Fernando Solarte Sarasty

**Universidad del Cauca**  
**Facultad de Ingeniería Electrónica y Telecomunicaciones**  
**Departamento de Telemática**  
Línea de Investigación en Ingeniería de Sistemas Telemáticos  
Popayán, Julio de 2003

## TABLA DE CONTENIDO

<b>1. PATRONES DE INTERFACES USADOS EN COMERCIO ELECTRONICO .....</b>	<b>1</b>
1.1 INTRODUCCIÓN.....	1
1.2 OPPORTUNISTIC LINKING .....	1
1.2 ADVISING .....	2
1.3 EXPLICIT PROCESS .....	4
1.4 EASY UNDO.....	5
1.5 PUSH COMMUNICATION .....	6
1.6 PRODUCT COMPARISON .....	8
1.7 IDENTIFY .....	10
1.8 SHOPPING CART .....	11
<b>2. ESTÁNDARES PARA ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE .....</b>	<b>14</b>
2.1 INTRODUCCIÓN.....	14
2.2 ESTÁNDARES PARA PRODUCTO .....	15
2.2.1 ISO/IEC 9126 .....	15
2.2.2 Nueva edición del estándar ISO/IEC 9126 .....	17
2.2.3 IEEE 1061 .....	20
2.3 ESTÁNDARES PARA PROCESO .....	24
2.3.1 CMM for Software (CMM-SW) Versión 1.1 .....	24
2.3.2 NTC – ISO 9003.....	27
<b>3. PATRONES DE DISEÑO PARA COMERCIO ELECTRÓNICO .....</b>	<b>29</b>
3.1 INTRODUCCIÓN .....	29
3.2 PATRÓN CATÁLOGO (CATALOG PATTERN).....	29
3.3 PATRÓN PROCESO DE COMPRA (SHOPING PROCESS PATTERN).....	31
3.4 PATRÓN OBSERVADOR (OBSERVER PATTERN).....	34
3.5 PATRÓN FACTORÍA ABSTRACTA (ABSTRACT FACTORY PATTERN).....	37
3.6 PATRÓN ESTRATEGIA (STRATEGY PATTERN).....	38

<b>4. AGENTES EN COMERCIO ELECTRÓNICO.....</b>	<b>40</b>
<b>4.1 CLASIFICACIÓN Y CARACTERIZACIÓN DE AGENTES PARA COMERCIO ELECTRÓNICO .....</b>	<b>41</b>
<b>4.2 AGENTES COMO INTERMEDIARIOS .....</b>	<b>44</b>
<b>4.3 BENEFICIOS DEL USO DE AGENTES EN COMERCIO ELECTRÓNICO .....</b>	<b>47</b>
<b>REFERENCIAS.....</b>	<b>49</b>

## INDICE DE FIGURAS

Figura 1.1 Estructura de Opportunistic Linking de Amazon.com.....	2
Figura 1.2 Estructura de Advising de cdnow.com.....	3
Figura 1.3 Implementación de Explicit Process en amazon.com .....	5
Figura 1.4 Implementación de Easy Undo en amazon.com .....	6
Figura 1.5 Suscripción a un servicio “push” en amazon.com .....	7
Figura 1.6 Push Communication en expedia.com .....	8
Figura 1.7 Pasos para comparar productos.....	9
Figura 1.8 Búsqueda de un producto en expedia.com .....	10
Figura 1.9 Resultado de la comparación de productos en expedia.com .....	10
Figura 1.10 Identificación de usuarios en amazon.com.....	11
Figura 1.11 Estructura de Shopping Cart.....	12
Figura 1.12 Shopping Cart en amazon.com .....	13
Figura 1.13 Wizard en amazon.com .....	13
Figura 1.14 Shopping Cart en waterpikstore.com.....	13
Figura 2.1 Arquitectura actual de las series ISO/IEC 9126 y 14598.....	16
Figura 2.2 Las 6 características de calidad definidas por la ISO/IEC 9126.....	17
Figura 2.3 Marco de trabajo de las métricas propuestas por el estándar IEEE 1061. ....	23
Figura 2.4 Composición de la Norma NTC-ISO 9000-3 .....	28
Figura 3.1 Estructura Patrón Catálogo.....	30
Figura 3.2 Diagrama de Colaboración Patrón Catálogo .....	31
Figura 3.3 Estructura Patrón Proceso de Compra .....	33
Figura 3.4 Diagrama de Colaboración Patrón Proceso de Compra .....	33
Figura 3.5 Estructura Patrón Observador .....	34
Figura 3.6 Estructura Patrón Factoría Abstracta.....	37
Figura 3.6 Estructura Patrón Estrategia.....	38
Figura 4.1. Clasificación de los agentes de negocios .....	42

## INDICE DE TABLAS

Tabla 2.1 Característica y subcaracterísticas de calidad del estándar ISO/IEC 9126. ....	19
Tabla 2.2 Factores y subfactores de calidad del estándar IEEE 1061 .....	23
Tabla 2.3 Criterios que definen un proceso maduro .....	26
Tabla 2.4 Niveles de madurez del CMM-SW .....	27
Tabla 2.5 Características Comunes definidas en el CMM-SW .....	27

## 1. PATRONES DE INTERFACES USADOS EN COMERCIO ELECTRONICO

### 1.1 INTRODUCCIÓN

Este capítulo describe algunos patrones de interfaces frecuentemente usados en aplicaciones de comercio electrónico en la Web y que son de utilidad como herramientas en las etapas del proceso de desarrollo de este tipo sistemas.

### 1.2 OPPORTUNISTIC LINKING

Problema: mantener al usuario interesado en el sitio. Seducirlo a navegar en el sitio siempre y cuando ya haya encontrado lo que inicialmente fue a buscar.

**Por qué:**

1. Es el deseo de los dueños del sitio mantener al usuario navegando en su negocio después de haber comprado algo.
2. No se desea comprometer la estructura del sitio por la adición de enlaces no significativos.

**Solución:** mejorar la topología de enlaces para sugerir la exploración de nuevos productos desde uno seleccionado. Usar relaciones con semántica fuerte para hacer que el usuario se sienta confortable. Se debe tener en cuenta que muchos de esos enlaces pueden cambiarse día tras día así que la interfase debe ser definida como tal. Se debe notar que este patrón puede usarse también a nivel conceptual para derivar nuevas relaciones. Sin embargo el intento es claramente navegacional: mantener al usuario navegando en un sentido agradable.

**Ejemplos:** el patrón Opportunistic Linking puede ser encontrado en muchas tiendas electrónicas. Por ejemplo se tienen [www.amazon.com](http://www.amazon.com) o [www.cdnnow.com](http://www.cdnnow.com) en las cuales se pueden encontrar CDs relacionados con el que se escoge. En Amazon.com se encuentra el caso de que cuando un usuario selecciona un libro, este recibe la sugerencia de otro libro que podría ser de su interés esto se puede ver en la figura 1.1.

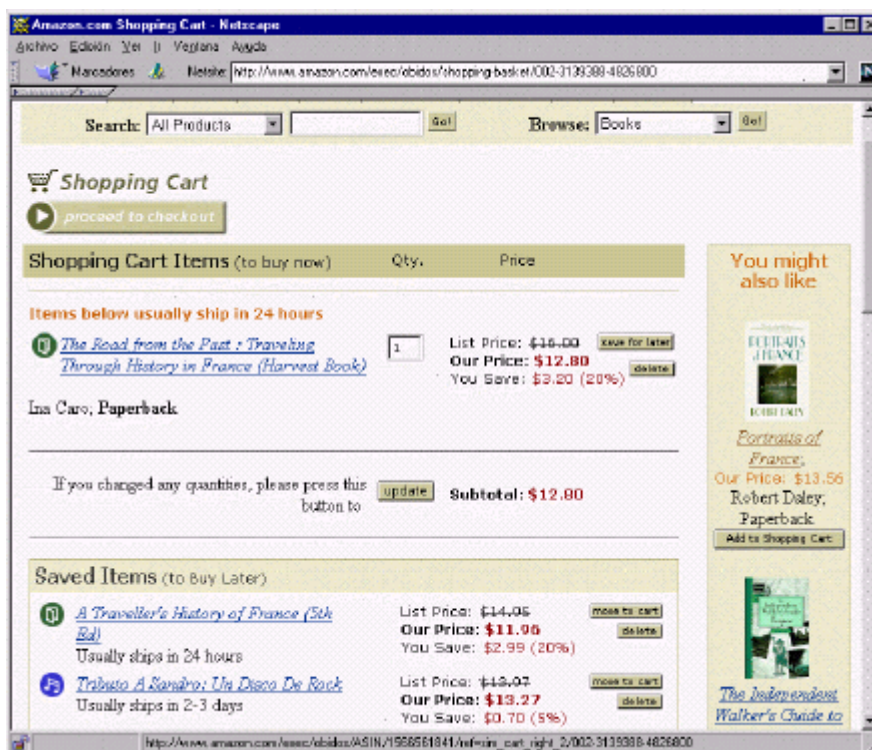


Figura 1.1 Estructura de Opportunistic Linking de Amazon.com

**Patrones relacionados:** El patrón Opportunistic Linking es bastante parecido al patrón Advising; en efecto se puede argüir que cuando se aplica este patrón se está haciendo Advising al consumidor. Sin embargo el intento es diferente. Mientras el Opportunistic Linking trata de mantener al usuario en la tienda suministrándole nuevas ideas para comprar, Advising ayuda a escoger lo que el consumidor desea.

## 1.2 ADVISING

Problema: ayudar al usuario a encontrar un producto en el sitio. Asistirlo acorde a sus expectativas.

**Por qué:**

1. Los consumidores en una tienda electrónica necesitan ser asistidos para encontrar un producto.
2. Motores de búsqueda e índices (por ejemplo taxonómicos) son útiles pero ellos representan una solución incompleta.
3. Se debería tener en cuenta lo que el usuario desea.

**Solución:** construir funcionalidades específicas para hacer recomendaciones acerca de productos. Esta funcionalidad debe ser implementada en diferentes caminos. Por ejemplo, puede ser un subsistema completo para recomendaciones como en [www.amazon.com](http://www.amazon.com) donde se utiliza perfiles de clientes(en general la historia de compras) para la recomendación de productos. Se puede ser mas general, y presentar a los usuarios un best seller de productos o productos para la venta etc. El diseño para facilidades de Advising no deberían interferir con la estructura global de navegación.

**Ejemplos:** El patrón de Advising se utiliza actualmente en casi todas la tiendas virtuales. Por ejemplo Amazon no solo provee recomendaciones de acuerdo al perfil del usuario sino que también incluye best sellers, actualizando cada hora la lista de los 100 libros recomendados. En [www.barnesandnoble.com](http://www.barnesandnoble.com) por ejemplo una sección de negocio es incluida en la página inicial junto con las recomendaciones generales. En [www.netgrocer.com](http://www.netgrocer.com) se da información acerca de sus productos en venta. En la figura 1.2 se da la estructura de Advising de [www.cdnow.com](http://www.cdnow.com). Muchos sitios de comercio electrónico agregan enlaces hacia el sitio para proveer descripciones imparciales de sus productos.



Figura 1.2 Estructura de Advising de cdnow.com

**Patrones relacionados:** Advising es similar a Opportunistic Linking. Sin embargo en el ejemplo el fin es ayudar al usuario a encontrar el camino hacia un producto, mientras que Opportunistic Linking esta apuntado a seducir al usuario una vez haya comprado el producto. Ambos patrones pueden ser considerados como versiones específicas de una más general.



### 1.3 EXPLICIT PROCESS

**Problema:** ayudara al usuario a comprender el proceso de compra cuando este no es atómico.

**Por qué:**

1. La forma de pago o el proceso de registro puede ser complejo.
2. El usuario tiende a sentirse desorientado a través de progreso del proceso.

**Solución:** dar al usuario una retroalimentación perceptible acerca del proceso manteniéndolo al tanto de los pasos que ya a completado. Esto puede ser hecho utilizando una línea de progreso o enumerando los pasos e informando donde se encuentra ahora. Tenga en cuenta las posibles consecuencias de este tipo de seguimiento (vea también Easy Undo ) para minimizar la posibilidad de alcanzar estados inconsistentes.

**Ejemplos:** en muchas tiendas se puede encontrar ejemplos de implementaciones de este patrón. Por ejemplo en [www.barnesandnoble.com](http://www.barnesandnoble.com) el cliente avanza a través de 7 pasos que son claramente indicados en un camino secuencial. En [www.amazon.com](http://www.amazon.com) la implementación es mas elegante como se puede observar el la figura 1.3. En la parte superior del la pantalla se puede observar un proceso lineal indicando que el usuario se encuentra en el paso “Ítems” y que se esperan aún cuatro pasos para finalizar el proceso.

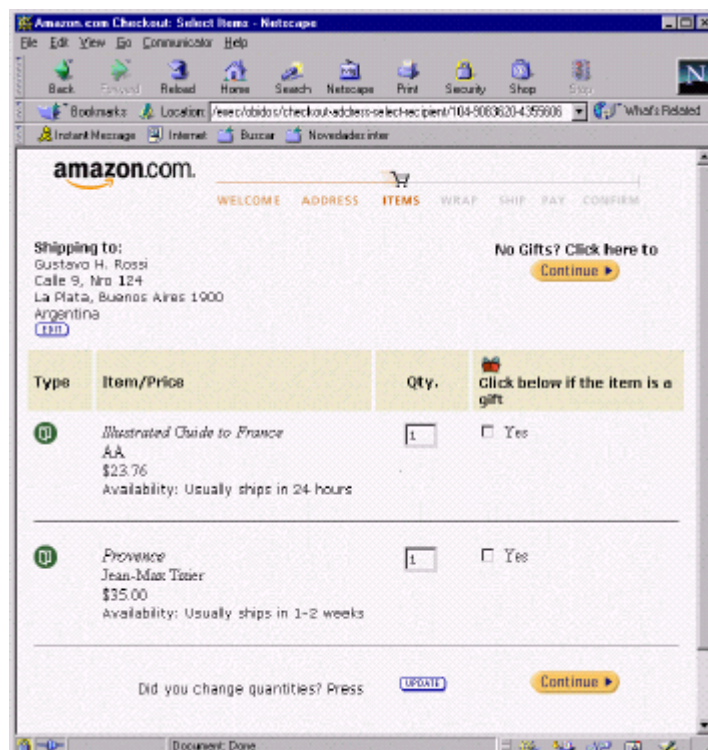


Figura 1.3 Implementación de Explicit Process en amazon.com

**Patrones relacionados:** el patrón Explicit Process está relacionado con Easy Undo puesto que los dos tienden a simplificar y hacer seguro el proceso de pago o registro(u otro proceso no atómico).

## 1.4 EASY UNDO

**Problema:** proveer capacidades para salvaguardar pérdidas en un proceso complejo.

**Por qué:**

1. El proceso de chequeo o registro puede ser complejo.
2. El usuario puede necesitar deshacer alguna operación previa
3. Usando el botón Back se pueden obtener resultados inesperados

**Solución:** proveer al usuario las facilidades de borrado evitándole que use las facilidades de navegación para ese propósito. Las facilidades de borrado deberán tener en cuenta el estado del proceso del consumidor para que sea efectivo. Este patrón extiende la idea del seguimiento típico en las aplicaciones Web adaptado dentro de la semántica de la aplicación. En lugar de retornar a la página Web anterior (Usando el botón Back), se retorna al estado correspondiente para borrar la operación. Es importante acentuar que esa diferencia es específica para este dominio mientras que la Web se basa en un paradigma hipertexto con una funcionalidad de seguimiento general.

**Ejemplos:** encontramos diferentes ejemplos de la implementación de este patrón en las aplicaciones de comercio electrónico tales como [www.powells.com](http://www.powells.com) donde el consumidor es expuesto hacia toda la información en el paso de confirmación y el puede escoger actualizar o modificar sus datos. Por otro lado en [www.amazon.com](http://www.amazon.com) (ver figura 1.4) la información del cliente es adherida de forma incremental y el puede escoger cambiar los datos suministrados previamente. Se pueden notar los botones pequeños a lado de cada ítem de información, indicando que se puede corregir la información. Cuando se selecciona editar la dirección por ejemplo, se retorna a la página correspondiente, y una vez cambiada se puede continuar el proceso.

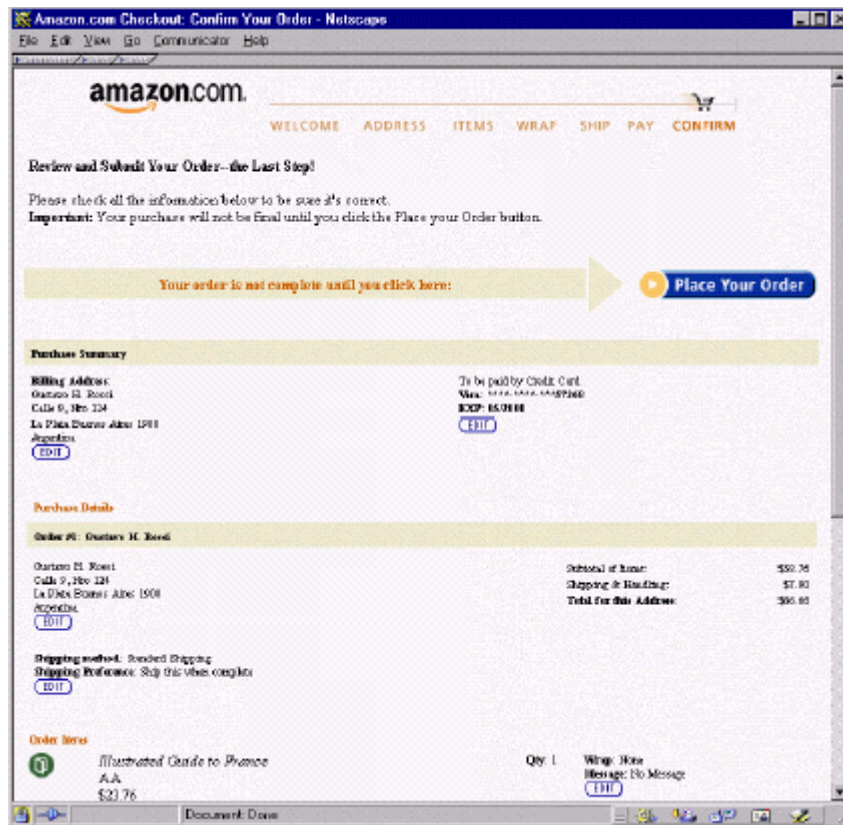


Figura 1.4 Implementación de Easy Undo en amazon.com

**Patrones relacionados:** el patrón Easy Undo puede ser usado en conjunto con Explicit Process manteniendo informado al usuario del estado de su proceso. Esto es un caso especial de Easy Undo.

## 1.5 PUSH COMMUNICATION

**Problema:** simplificar el proceso de búsqueda para áreas o productos seleccionados por el usuario.

**Por qué:**

1. Encontrar información nueva no es fácil siempre.
2. El usuario gasta una parte del tiempo conectándose para encontrar nuevos productos.
3. El modelo Web básico es basado en sacar información.

**Solución:** combinar el modelo Web usual de sacar con una comunicación basada en sacar. Se encuentran caminos para comunicarse con el cliente sin forzarlo a encontrar la información. Esta solución puede tener diferentes implementaciones; por ejemplo el cliente puede suscribirse a un

tema dado (o tipo de producto) y recibir un correo electrónico periódicamente informándole que un nuevo ítem de su interés apareció. Este correo electrónico contiene directamente la URL del producto para que el saque la información. Otra alternativa posible es personalizar su sitio usando “channels”. Cada vez que el usuario entra en un canal el encuentra información sobre el contenido que le gusta. Mandar correos y canales puede ser combinado para mejorar el acceso a la información por parte del cliente.

**Ejemplos:** muchas tiendas virtuales proveen algún mecanismo de suscripción para ayudar a sus clientes a que sepan cuando un ítem particular se ha incluido en la tienda. En [www.amazon.com](http://www.amazon.com) periódicamente un cliente busca un producto (en un tema particular o área) el es invitado a aceptar recibir correos electrónicos periódicamente por novedades, esto se puede ver en la figura 1.5. En [www.izero.com](http://www.izero.com) y en [www.netzero.com](http://www.netzero.com) que provee acceso a Internet en forma gratuita, Push Communication es usada para enviar anuncios al usuario. En la figura 1.6 se puede observar el caso de Expedia.com, el cliente selecciona un itinerario y el puede recibir una confirmación sobre pasajes baratos por correo electrónico o teniendo un perfil de usuario actualizado (una clase de channel). En este ejemplo el cliente puede escoger ir hacia su perfil personal para tener información relacionada con sus preferencias.



Figura 1.5 Suscripción a un servicio “push” en amazon.com

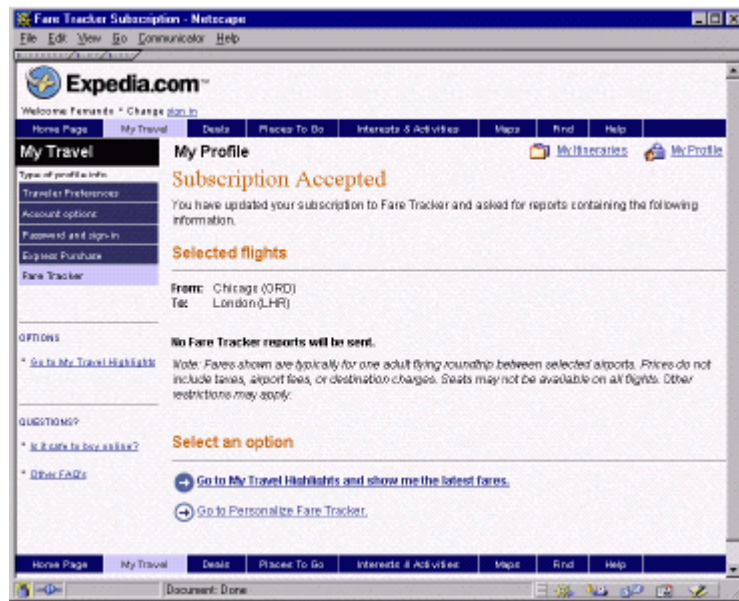


Figura 1.6 Push Communication en expedia.com

**Patrones relacionados:** Push Communication puede ser considerado como una implementación particular del patrón News propuesto por Gustavo Rossi. Sin embargo se prefiere diferenciar los dos patrones puesto que ellos comprenden diferentes estrategias para la administración la comunicación cliente-negocio, además que involucran tecnologías diferentes.

## 1.6 PRODUCT COMPARISON

**Problema:** el usuario necesita comparar productos similares.

**Por qué:**

1. Las características del producto se necesitan inmediatamente y con facilidad para ser comparadas con las de productos similares.
2. Se deben marcar sólo las características disponibles con el fin de minimizar el desorden visual.

**Solución:** mostrar una matriz de productos y características. Indicar las características en las filas de la matriz y los productos en las columnas. Si una característica está presente, esta se marca sino no. Si la característica tiene datos de interés específico para los usuarios, se muestran los datos en lugar de la marca.

Si hay muchas características, ellas pueden ser agrupadas para que los usuarios puedan seleccionar con cuales de ellas comparar. Cuando hay muchos productos para comparar, los usuarios seleccionan cuales productos desean comparar. Esto puede ser hecho en una página

cuando no hay muchos productos, de lo contrario se puede hacer en dos pasos, esto se puede ver en la figura 1.7.

### 1. ESCOJER LOS PRODUCTOS A COMPARAR

Product 1

Product 2

Product 3

Product 4

Compare



### 2.

Advanced Features ▾

Features	Product 1	Product 2
Feature X	✓	
Feature Y	200 units	100 units

Figura 1.7 Pasos para comparar productos

**Ejemplos:** este ejemplo de [www.expedia.com](http://www.expedia.com) combina una Producto Comparison con búsqueda de resultados. Un resultado de búsqueda puede ser seleccionado para comparación y dando clic en el enlace “compare selected cruises” la comparación se muestra (ver figuras 1.8 y 1.9). En la ventana de comparación los usuarios también pueden deseleccionar.

5 Cruises Found

[Compare selected cruises](#)

[How to compare cruises](#)

**17 Day Panama Canal (New York to Valparaiso) from \$1,299\***

Compare

Brochure Price \$2,369 - Save \$1,070!

★★★★★ CELEBRITY CRUISES

Ship Name: Zenith

Get ready for an adventure of grand proportions as you sail the Zenith from New York to Santiago in 17 days. Head south ... [Detailed cruise information](#)

**Ports of Call:** Departs from New York, New York with stops in Charlotte Amalie, St. Thomas; Oranjestad, Aruba; Panama Canal (Transit Canal); Manta, Ecuador; Callao (Lima), Peru; Arica, Chile; Valparaiso (Santiago), Chile

**Departures:** October 26

2 passengers    Each additional

Price is cruise-only, per-person and includes port charges.    from **\$1,299** each    from **\$909**    [Choose & continue](#)

---

**14 Day Fall Westbound Panama Canal (Ft. Lauderdale to Los Angeles) from \$1,495\***

Compare

Brochure Price \$2,724 - Save \$1,229!

★★★★★ PRINCESS CRUISE LINES

Ship Name: Royal Princess

Live like royalty aboard the spacious Royal Princess as you cruise from Ft. Lauderdale to Los Angeles through the Panama ... [Detailed cruise information](#)

**Ports of Call:** Departs from Ft. Lauderdale, Florida with stops in Oranjestad, Aruba; Cartagena, Colombia; Panama Canal (Transit Canal); Puntarenas, Costa Rica; Acapulco; Cabo San Lucas; Los Angeles, California

**Departures:** October 31

2 passengers    Each additional

Price is cruise-only, per-person and includes port charges.    from **\$1,495** each    from **\$799**    [Choose & continue](#)

Figura 1.8 Búsqueda de un producto en expedia.com

**PRINCESS**

From **\$1495**

Brochure Price \$2,724 - Save \$1,229!

[Availability & prices](#)

---

Royal Princess

Ship Rating: ★★★★★+

Cabins: 600

Crew Size: 600

Language: International

Capacity: 1200

Registry: Britain

---

**Day Port**

1	Ft. Lauderdale, Florida
2	At Sea
3	At Sea
4	Oranjestad, Aruba
5	Cartagena, Colombia
6	Panama Canal (Transit Canal)
7	At Sea
8	Puntarenas, Costa Rica
9	At Sea
10	At Sea
11	Acapulco
12	At Sea
13	Cabo San Lucas
14	At Sea
15	Los Angeles, California

Check to Remove

[Update Cruise Compare](#)

[Return to Search Results](#)

**Celesty Cruise**

From **\$1299**

Brochure Price \$2,369 - Save \$1,070!

[Availability & prices](#)

---

Zenith

Ship Rating: ★★★★★+

Cabins: 687

Crew Size: 654

Language: International

Capacity: 1374

Registry: Liberia

---

**Day Port**

1	New York, New York
2	At Sea
3	At Sea
4	At Sea
5	Charlotte Amalie, St. Thomas
6	At Sea
7	Oranjestad, Aruba
8	At Sea
9	Panama Canal (Transit Canal)
10	At Sea
11	Manta, Ecuador
12	At Sea
13	Callao (Lima), Peru
14	At Sea
15	Arica, Chile
16	At Sea
17	At Sea
18	Valparaiso (Santiago), Chile

Check to Remove

Figura 1.9 Resultado de la comparación de productos en expedia.com

**Patrones relacionados:** No se conocen.

## 1.7 IDENTIFY

**Problema:** el usuario necesita identificarse por si mismo.

**Por qué:**

1. El usuario no desea ser molestado con los procedimientos de identificación.
2. Se necesita permitir la exploración del sitio sin ningún compromiso.

**Solución:** Preguntar la identificación del usuario pero solo cuando sea necesario. Permitir a los usuarios accesos libre hasta que sea absolutamente necesario identificarse. Use una combinación de la dirección de correo electrónico y un password. Opcionalmente la dirección de correo electrónico puede ser llenada automáticamente la próxima vez que el usuario regrese. Con el uso



de la dirección de correo como nombre de usuario, este puede pedir que se lo envíen cuando lo haya olvidado. Algunos usuarios son Identificados y se pide una realimentación para confirmarlo.

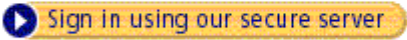
**Ejemplos:** en [www.amazon.com](http://www.amazon.com) los usuarios son identificados por su dirección de correo electrónico y un password, también se ofrece ayuda en caso de haber olvidado el password (ver figura 1.10).

**Ordering online is easy.**  
We'll walk you through the process, step by step.

Enter your e-mail address:

**I am a new customer.**  
(You'll create a password later.)

**I am a returning customer,  
and my password is:**

 Sign in using our secure server

[Forgot your password? Click here](#)

Figura 1.10 Identificación de usuarios en amazon.com

**Patrones relacionados:** se puede considerar el patrón Registration para el registro de nuevos usuarios.

## 1.8 SHOPPING CART

**Problema:** los usuarios desean comprar un producto.

**Por qué:**

1. Shopping cart es bien conocida y es una metáfora internacional.
2. Necesidad de mostrar el costo total de los productos a comprar.

**Solución:** introducir un Shopping Cart donde los usuarios puedan colocar los productos antes de ser comprados. Cuando el usuario ve la descripción de un producto, el puede escoger adicionar este a su shopping cart. Después de adicionar un ítem a su carta, al usuario se le muestra los contenidos actuales de la carta. Los usuarios pueden verificar los contenidos de la carta en



cualquier instante usando un enlace siempre disponible en su página. Una mini carta siempre debe estar visible en la página de contenidos.

La descripción de los contenidos de la carta contiene generalmente los nombres de los ítems, la cantidad, disponibilidad y precios. Los usuarios pueden quitar ítems de su carta si ellos lo desean o cambiar cantidades. La descripción es un enlace hacia los detalles del producto. El usuario siempre ve el total de la compra, también incluyendo los gastos de envío si son aplicables. Los usuarios deben también ser informados de las opciones de pago tales como las tarjetas de crédito aceptadas. Desde la página de la carta, los usuarios pueden seguir comprando o seguir con el proceso de selección y pago (Checkout). Los ítems están en la carta por un cierto periodo de tiempo, por ejemplo 60 días, este orden de ideas se puede observar en la figura 1.11.

Description	Quantity	Availability	Remove	Price
<a href="#">Items description</a>	1 + -	In stock	<a href="#">remove</a>	\$ 6.50
<a href="#">Item2 description</a>	2 + -	On order	<a href="#">remove</a>	\$19.50
				<b>Subtotal \$26.00</b>
Express mail \$3.95				<b>Shipping \$ 3.95</b>
				<b>Total \$29.95</b>

<< Proceed Shopping

Checkout >>



- We accept Visa and Mastercard.
- Transactions are done using secure connections.

Figura 1.11 Estructura de Shopping Cart.

En caso de comprar los productos en la carta se necesita hacer el Checkout. Este consiste de proceso de cinco pasos con las siguientes tareas: Identificar y Seleccionar la dirección de envío, Seleccionar el método de pago, Hacer la orden. El usuario puede abandonar el proceso de Checkout en cualquiera de los pasos. Cuando el usuario retorna al Checkout mas tarde, este empieza otra vez desde el primer paso. Se debe considerar un Wizard para guiar al usuario a través de las tareas mientras minimiza el número de páginas Web usadas.

Ejemplos:



Figura 1.12 Shopping Cart en amazon.com

En [www.amazon.com](http://www.amazon.com) (ver figura 1.12) los usuarios pueden escoger entre muchos productos y adicionarles a su carta sin ninguna obligación. Ellos pueden ver el contenido de la carta con un clic y proceder con la actual compra cuando lo desee. La opción para ver los contenidos de la carta están disponibles en cualquier página. Cuando se esta haciendo Checkout una realimentación es dada correspondiente a los pasos (Wizard) esto se puede observar el la figura 1.13.



Figura 1.13 Wizard en amazon.com

Otro ejemplo de Shopping Cart se puede encontrar en [www.waterpikstore.com](http://www.waterpikstore.com) (ver figura 14)

[Continue Shopping](#)
[Checkout](#)

Item	Qty	Avail.	Price Ea.	Ext Price	Remove
AquafallT JP-120	1 <input type="checkbox"/>	Available	\$29.99	\$29.99	REMOVE

**Calculate Shipping Cost**

Enter Postal Code to calculate shipping cost.

SubTotal:	\$29.99
Total Discounts:	\$0.00
Quick Freight (to , US via ):	\$0.00
Adjusted SubTotal:	\$29.99

[Continue Shopping](#)
[Checkout](#)

Figura 1.14 Shopping Cart en waterpikstore.com

**Patrones relacionados:** se puede considerar le patrón Identify cuando los usuarios desean identificarse por si mismos.

## 2. ESTÁNDARES PARA ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

### 2.1 INTRODUCCIÓN

Internet en nuestros días esta involucrada en gran parte de las actividades que desarrollan las personas, actividades como la compra de libros, operaciones bancarias, reservaciones de hoteles, pasajes etc. Debido a esto el crecimiento de sistemas y aplicaciones basados en Web ha sido bastante acelerado ya que se busca cubrir cada vez más el consumo de un sector que presenta unas oportunidades de negocio considerables para los empresarios. Desgraciadamente el crecimiento acelerado de este tipo de aplicaciones ha dado pie para que su construcción no este guiado por marcos de referencia que aseguren la calidad de este tipo de sistemas software, los cuales cada día hacen mas parte de la vida cotidiana y de los cuales dependen tantas actividades que involucran conceptos tanto de tiempo, seguridad, satisfacción entre otros. La preocupación respecto a las prácticas usadas para el desarrollo de aplicaciones sobre plataforma Web se ha venido generalizando últimamente lo cual ha llevado a la aparición de una nueva disciplina denominada Ingeniería Web, la cual se puede definir de la siguiente como " El proceso empleado para crear, implantar y mantener aplicaciones y sistemas Web de alta calidad" [Nieto 01].

Por todo lo anterior es fácil identificar la importancia que tiene la calidad en el desarrollo de aplicaciones Web puesto que asegura que los sistemas que se entregan se van a desempeñar eficientemente en el ambiente de ejecución y según los requisitos de los usuarios. Es conocida la existencia de estándares de calidad publicados por organismos internacionales con el fin de dar lineamientos a múltiples actividades de los diferentes sectores de desarrollo económico de la sociedad. Pero vale la pena clarificar cual es la importancia que tienen estos estándares tanto para los entes involucrados en el desarrollo o producción de un bien o servicio así como para los consumidores de estos últimos y más aún para el caso en el cual los desarrolladores y los consumidores pertenecen al mundo del software: " Enfrentamos una situación con dos caras. Por una parte las organizaciones quieren ser capaces de desarrollar y entregar software confiable, a tiempo y apegado al presupuesto acordado con el cliente. La segunda cara de la moneda nos muestra la perspectiva del cliente, el cuál quiere saber con certeza que todo lo anterior se cumplirá. Por esto las organizaciones deben buscar una norma, estándar o modelo que pueda ayudarlas a conseguir su meta de calidad (competitividad) "[García 01].

La importancia del papel que juegan los estándares internacionales de calidad en el desarrollo de software es hoy por hoy definitiva debido al comportamiento del mercado. Debido a esta coyuntura se planteó la introducción de conceptos de aseguramiento de la calidad del software en el MICACE. El presente capítulo contiene los aspectos complementarios referentes a los estándares para el aseguramiento de la calidad de software expuestos en el marco teórico de monografía.

## **2.2 ESTÁNDARES PARA PRODUCTO**

A continuación se describen con un grado de detalle complementario los estándares para producto de software que se tuvieron en cuenta para la creación del MICACE.

### **2.2.1 ISO/IEC 9126**

El comité ISO/IEC JTC1/SC7/WG6 ha contribuido a desarrollar dos series de estándares internacionales sobre la calidad del producto de software, estas son la ISO/IEC 9126 y la ISO/IEC 14598 las cuales pueden observarse en la figura 2.1, en esta se describe el entorno de aplicación de cada una de sus partes. Para dar un poco de claridad respecto a la sigla anterior se tiene que JTC1/SC7 es un comité de la ISO responsable del desarrollo de estándares ISO en el área de Ingeniería de Sistemas y Software, cuyo mandato es la estandarización de procesos, soporte de herramientas y tecnologías para la ingeniería de productos de software y sistemas. La parte correspondiente al WG6 (Working Group) hace referencia a que cada comité tiene sus grupos de trabajo, entonces el grupo de trabajo WG6 estuvo a cargo del desarrollo del estándar.

El proyecto de la ISO el cual fue originalmente responsable de “La evaluación de la Calidad del Producto de Software” fue el llamado 7.13, el cual comenzó en 1985, con lo cual se publicó una lista de características de calidad y sus definiciones. Los miembros del WG tomaron como referencia el modelo de Boehm[Boehm76] y el modelo de McCall[Mc77]. También se decidió que el proceso de evaluación de la calidad sea incluido en el estándar.

Sin embargo la comunidad en el WG no fue estable, lo cual hizo que el trabajo no fuera tan productivo. El 1s-DP (Draft Proposal) fue firmado en 1986. El estándar internacional fue publicado en 1991 con el nombre de “ISO/IEC 9126: Information technology-Software product evaluation-Quality characteristics and the guidelines for their use”. Basándose en la necesidad de generar más información sobre la calidad del producto de software, el proyecto 07.13, el cual arrancó siendo un simple proyecto fue dividido en 3 subproyectos; 07.13.01 Características de Calidad, 07.13.02

Subcaracterísticas de calidad, y el 07.13.03 Medidas y Clasificación. Todo lo anterior se hizo en la reunión del SC7 de Budapest en 1989. Una cuarta parte se adicionó al estándar no hace muchos años llamada “Métricas de Calidad en Uso”, esto se hizo en 1998.

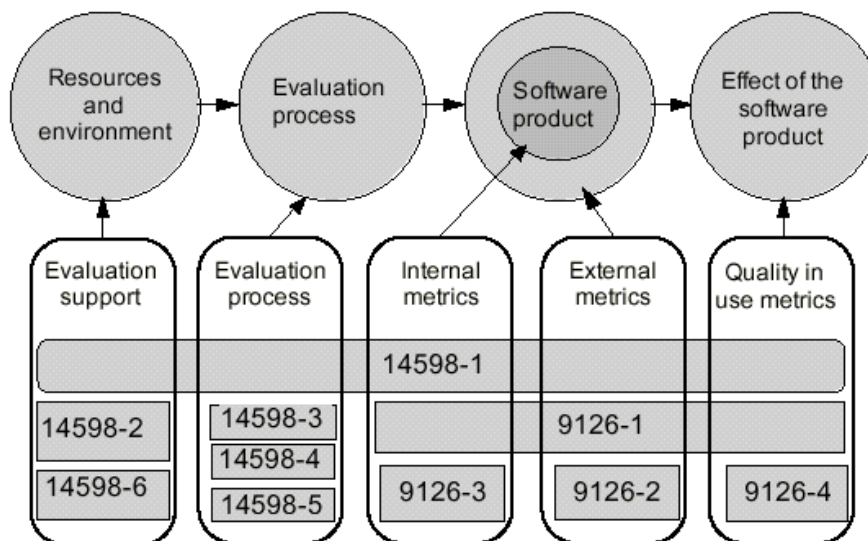


Figura 2.1 Arquitectura actual de las series ISO/IEC 9126 y 14598.

El objetivo del estándar ISO/IEC 9126 es proveer un esquema para la evaluación de la calidad de productos de software. El estándar no provee requisitos para el software, pero define un modelo de calidad el cual es aplicable a cualquier clase de software. El estándar declara 6 características de calidad del producto de software y en el anexo del mismo se sugiere una lista de subcaracterísticas. Un esquema de la composición de este estándar se puede visualizar en la figura 2.2.

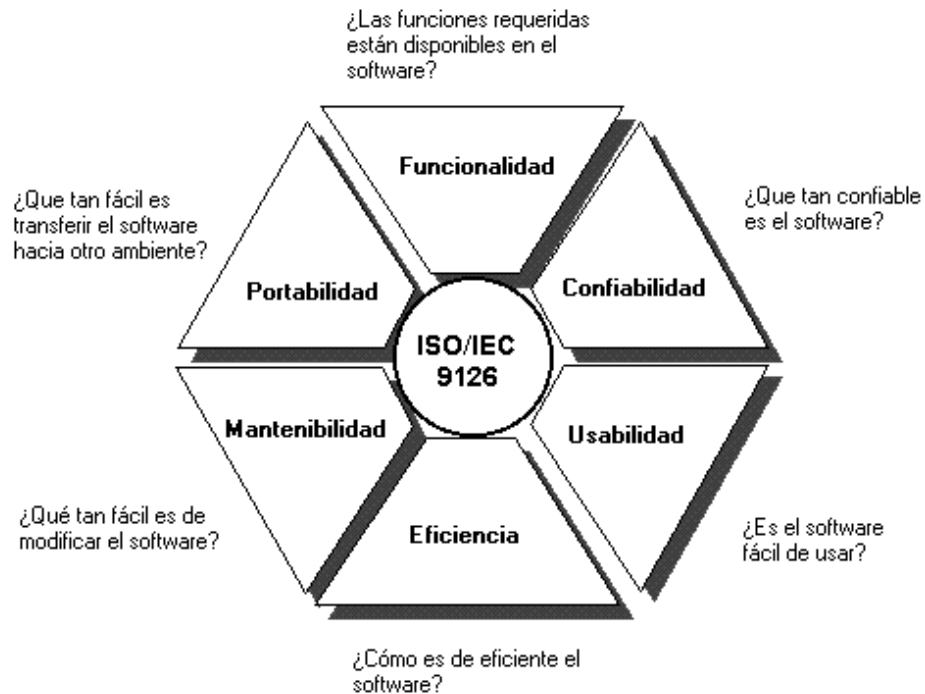


Figura 2.2 Las 6 características de calidad definidas por la ISO/IEC 9126

### 2.2.2 Nueva edición del estándar ISO/IEC 9126

El estándar actual se encuentra bajo revisión, la versión con que se cuenta por el momento está constituida por los siguientes documentos:

- ISO/IEC 9126-1: Information technology - Software quality characteristics and metrics - Parte 1: Características y subcaracterísticas de calidad.

Esta parte provee los conceptos introducidos en el estándar original y su modelo de calidad el cual establece 6 características de calidad para el software, las cuales están subdivididas en subcaracterísticas. Las subcaracterísticas han sido movidas desde el anexo para formar parte del estándar. Algunas han sido reordenadas y algunas nuevas han sido adheridas “Particularmente, cada característica cuenta con la subcaracterística *Conformidad* (o *Compliance*) y, para el caso de *Usabilidad*, se le ha agregado la subcaracterística *Grado de Atracción* (o *Attractiveness*). De manera que el modelo de calidad y el mecanismo de descomposición de características y subcaracterísticas es

semejante a la versión de 1991”[Olsina 99]. Las definiciones de las subcaracterísticas se pueden observar en la tabla 2.1.

<b>Características</b>	<b>Subcaracterísticas</b>	<b>Definiciones</b>
	Conveniencia	Atributos del software que influyen sobre la presencia y apropiación de un conjunto de funciones para tareas específicas.
	Exactitud	Atributos del software que influyen sobre la provisión de efectos o resultados correctos.
<b>Funcionalidad</b>	Interoperabilidad	Atributos del software que influyen sobre la habilidad de interactuar con sistemas específicos.
	Conformidad	Atributos del software que hacen al software ceñirse a la aplicación de estándares relacionados, convenciones o regulaciones dadas por leyes y prescripciones similares.
	Seguridad	Atributos del software que influyen sobre la habilidad para prevenir accesos no autorizados, accidentales o deliberados, hacia programas o datos.
	Madurez	Atributos del software que influyen sobre la frecuencia de fallos por defectos en el software.
<b>Confiabilidad</b>	Tolerancia a fallos	Atributos del software que influyen sobre la habilidad para mantener un nivel específico de rendimiento en caso de defectos de software o violación de su interfaz específica.
	Recuperabilidad	Atributos del software que influye sobre la capacidad de reestablecer su nivel de funcionamiento y recobrar los datos directamente afectados en caso de fallos y sobre el tiempo y esfuerzo necesario para esto.
	Entendibilidad	Atributos del software que influyen sobre el esfuerzo para el reconocimiento del concepto lógico y su aplicabilidad.
<b>Usabilidad</b>	Facilidad de aprender	Atributos de software que influyen sobre el esfuerzo del usuario para aprender su aplicación.
	Operabilidad	Atributos del software que influyen sobre el esfuerzo del usuario para la operación y el control de operación.
<b>Eficiencia</b>	Comportamiento de	Atributos del software que influyen sobre la respuesta

	Tiempo	y el procesamiento en el tiempo y sobre su rata de datos para desempeñar su función.
	Comportamiento de Recurso	Atributos del software que influyen sobre la cantidad de recursos usados y la duración de dicho uso en el desempeño de su función.
	Analisabilidad	Atributos del software que influyen sobre el esfuerzo necesario para el diagnóstico de deficiencias o causas de fallos, o para la identificación de partes a ser modificadas.
<b>Mantenibilidad</b>	Modificabilidad	Atributos del software que influyen sobre el esfuerzo necesario para la modificación, corrección de defectos o para un cambio de ambiente.
	Estabilidad	Atributos del software que influyen sobre el riesgo, efectos inesperados o modificaciones.
	Testeabilidad	Atributos del software que influyen sobre el esfuerzo necesario para validar el software modificado.
	Adaptabilidad	Atributos de software que influyen en la oportunidad de adaptación en diferentes entornos sin la aplicación de otras acciones o medios dados para ese propósito.
<b>Portabilidad</b>	Instalabilidad	Atributos del software que influyen sobre el esfuerzo necesario para instalar el software en un ambiente específico.
	Conformidad	Atributos del software que hacen al software ceñirse a estándares o convenciones relativas a portabilidad.
	Reemplazabilidad	Atributos del software que influyen sobre la oportunidad y el esfuerzo usando este en lugar de especificar otro en el mismo ambiente.

Tabla 2.1 Característica y subcaracterísticas de calidad del estándar ISO/IEC 9126.

- ISO/IEC 9126-2: Information technology - Software quality characteristics and metrics - Parte 2: Métricas externas.



Esta parte contiene las métricas externas para la medición de las características de calidad. Una métrica es un método de escala y medición cuantitativa, el cual puede ser usado para medir un atributo o una característica de un producto de software, derivado desde la conducta del sistema al cual hace parte. Una métrica externa es aplicable a un producto de software ejecutable durante la prueba u operación en un avanzado estado de desarrollo y después de entrar en operación.

- ISO/IEC 9126-3: Information technology - Software quality characteristics and metrics - Parte 3: Métricas internas.

Esta parte provee métricas internas para la medición de las características de calidad del software. Una métrica interna es una medición cuantitativa, el cual puede ser usado para la medición de un atributo o característica de un producto de software, derivado del producto en si mismo, directa o indirectamente( no es derivado por medidas de la conducta del sistema). Las métricas internas son aplicables a producto de software no ejecutable durante el diseño y la codificación en estados tempranos del proceso de desarrollo.

- ISO/IEC 9126-4: Information technology - Software quality characteristics and metrics - Parte4: Métricas de calidad en uso.

“La calidad en uso es la visión del usuario de la calidad de un sistema que contiene software, y es medido en términos de los resultados usando el software, más que de las propiedades del software en sí mismo. La calidad en uso es el efecto combinado de las características de calidad del software para el usuario”[Bevan 99].

De lo anterior se puede observar que las métricas de la calidad en uso están ligadas a la apreciación del usuario de cómo se comporta el software.

### **2.2.3 IEEE 1061**

El estándar provee una metodología para el establecimiento de requisitos de calidad y la identificación, implementación, análisis, y validación de las métricas de calidad del producto y proceso de software. La metodología se puede aplicar en todas las fases de cualquier ciclo de vida del software. Para el propósito de este estándar, definir la calidad de software para un sistema es equivalente a definir una lista de atributos de calidad requeridos para ese sistema. Para medir los atributos de calidad de software, se define un conjunto de métricas.

El propósito de las métricas de software es hacer la valoración a través del ciclo de vida del software para ver si los requisitos de calidad están siendo reunidos. El uso de métricas de software reduce la subjetividad en la valoración y el control de la calidad del software, ya que se provee una base cuantitativa para tomar decisiones sobre la calidad. Sin embargo, el uso de métricas de software no elimina la necesidad de juicio humano en las evaluaciones del software. El uso de métricas de software dentro de una organización o proyecto hace que se espere un efecto benéfico debido a que hace la calidad de software más visible.

La versión del estándar publicada en 1993 definía un conjunto de factores y subfactores para valorar la calidad del software (Anexo A del estándar). En la tabla 2.2 se puede observar la definición de estos conceptos.

FACTORES	PREGUNTA CENTRAL	SUBFACTORES	PREGUNTA CENTRAL
<b>Eficiencia</b>	¿Es rápido y minimalista en cuanto a uso de recursos, bajo ciertas condiciones?	Economía respecto al tiempo	¿Realiza funciones especificadas bajo ciertas condiciones y en el marco de tiempo apropiado?
		Economía respecto a recursos	¿Realiza funciones especificadas bajo ciertas condiciones usando cantidad apropiada de recursos?
<b>Funcionalidad</b>	¿Las funciones y propiedades satisfacen todas las necesidades explícitas e implícitas de los usuarios?	Compleitud	¿Posee las necesarias y suficientes funciones para satisfacer a los requerimientos del usuario?
		Correctitud	¿Especifica todas las funciones?
		Seguridad	¿Detecta y previene uso ilegal, destrucción de recursos, pérdida o filtración de información?
		Compatibilidad	¿El nuevo software se puede instalar sin cambiar el ambiente y las condiciones?
		Interoperabilidad	¿Se conecta y opera fácilmente con otros sistemas?

<b>Mantenibilidad</b>	¿Es fácil de modificar y probar?	Correctibilidad	¿Es fácil corregir errores y tratar con las demandas de usuarios?
		Expandibilidad	¿Es fácil de mejorar y modificar la eficiencia de las funciones?
		Testeabilidad	¿El software es fácil de probar
<b>Portabilidad</b>	¿Es fácil de transferir de un ambiente a otro?	Independencia de Hardware	¿El software, depende de entornos de hardware específicos?
		Independencia de software	¿El software, depende de entornos de software específicos?
		Instalabilidad	¿Es fácil de ajustar el software para ese nuevo entorno?
		Reusabilidad	¿El software es fácil de reusar en otras aplicaciones?
<b>Confiabilidad</b>	¿Puede mantener el nivel de rendimiento, bajo ciertas condiciones y por cierto tiempo?	No-deficiencia	¿No contiene errores?
		Tolerancia a errores	¿Si suceden fallas, como se comporta en cuanto a la performance especificada? ¿Posee funciones de recuperación?
		Disponibilidad	¿Se mantiene operable en presencia de fallas del sistema?
<b>Usabilidad</b>	¿El software, es fácil de usar y de aprender?	Comprensibilidad	¿Es fácil de comprender?
		Facilidad de aprender	¿Se minimiza el esfuerzo para comprender el software?
		Operabilidad	¿La operación está conforme al objetivo, contexto, y factores ergonómicos como color, forma?

		Nivel de comunicación	¿Se diseña el software conforme a las características psicológicas de los usuarios?
--	--	-----------------------	---

Tabla 2.2 Factores y subfactores de calidad del estándar IEEE 1061

Sin embargo en 1998 se hizo una revisión del estándar donde se propuso que el anexo A desapareciera y la discusión al respecto de factores, subfactores, y métricas de calidad pasara a la Cláusula 3 del estándar revisado. Una ilustración de cómo estos tres conceptos se conjugan para definir la calidad de software se puede observar en la figura 2.3.

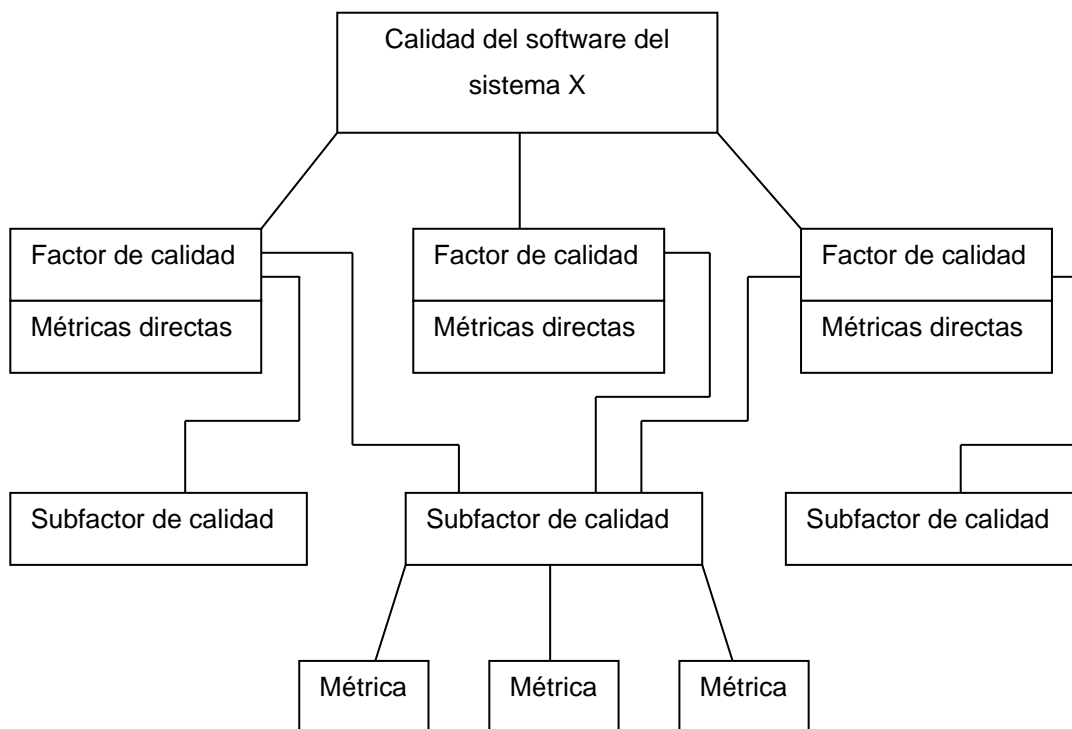


Figura 2.3 Marco de trabajo de las métricas propuestas por el estándar IEEE 1061.

Actualmente el estándar se compone de las siguientes partes: Cláusula 1, Alcance del estándar; Cláusula 2, Conjunto de definiciones; Cláusula 3, Marco de trabajo para las métricas de calidad de software; Cláusula 4, Metodología para las métricas de calidad de software.

## 2.3 ESTÁNDARES PARA PROCESO

A continuación se describen con un grado de detalle complementario los estándares para proceso de software que se tuvieron en cuenta para la creación del MICACE.

### 2.3.1 CMM for Software (CMM-SW) Versión 1.1

El CMM-SW provee a las organizaciones una guía sobre como obtener control de sus procesos<sup>1</sup> para el desarrollo y mantenimiento de software, y como evolucionar hacia una cultura de ingeniería de software y administración de la excelencia. El CMM-SW fue diseñado para guiar a las organizaciones en la selección de estrategias para la mejora de procesos, a través de la determinación de la madurez de procesos actual y la identificación de unos pocos puntos críticos para la calidad de software y la mejora de procesos. De acuerdo a este enfoque, él dirigirse sobre un conjunto limitado de actividades y trabajando agresivamente para lograrlas, una organización puede sostenidamente mejorar sus procesos y habilitar continuas y duraderas ganancias en la *capacidad* de procesos de software.

En el CMM-SW la mejora continua de procesos es basada sobre muchas innovaciones pequeñas y evolutivas mas que en evoluciones drásticas y revolucionarias. CMM-SW provee un marco de trabajo para organizar esos pasos evolutivos dentro de cinco niveles de madurez que sitúan fundamentos sucesivos para la mejora continua de procesos. Un *nivel de madurez* es una “meseta” evolutiva bien definida hacia el logro de la *madurez* de procesos. Cada nivel de madurez compromete un conjunto de metas, que cuando son satisfechas, estabiliza un componente importante de los procesos de software. Alcanzando cada nivel del marco de trabajo de madurez se establece un componente diferente dentro del proceso de software, dando como resultado un incremento en la capacidad de procesos de la organización.

La *capacidad del proceso de software* describe el rango de los resultados esperados que pueden ser logrados siguiendo un proceso de software. La capacidad de procesos de software de una organización provee una manera de predecir las salidas más probables que se esperan para el siguiente proyecto de software que la organización ponga en marcha [Paulk 93].

---

<sup>1</sup> Un proceso de software según el CMM-SW, puede ser definido como un conjunto de actividades, métodos, prácticas, y transformaciones, que la gente usa para desarrollar y mantener software y productos asociados (tales como planes de proyectos, documentos de diseño, código, casos de prueba, y manuales de usuario.)

La *madurez del proceso de software* es el rango en el cual un proceso específico es explícitamente definido, administrado, medido, controlado, y efectivo [Paulk 93]. Respecto al concepto de proceso maduro a continuación (tabla 2.3) se presenta los criterios dados por [Guerrero 00] basada en el CMM-SW.

<b>Está definido</b>	El proceso es claro, sistemático y suficientemente detallado. Además existe acuerdo entre el personal, la gerencia y los proyectos respecto al proceso que se va a utilizar.
<b>Esta documentado</b>	Esta escrito en un procedimiento publicado, aprobado y fácilmente accesible. Una de las mejores maneras es a través de una Intranet para apoyar los proyectos de desarrollo.
<b>El personal ha sido entrenado en el proceso</b>	Los ingenieros de software y la gerencia han recibido cursos y entrenamiento en cada proceso que aplica a su trabajo
<b>Es practicado</b>	El proceso definido debe ser usado en las tareas habituales llevadas a cabo por los proyectos. El entrenamiento y la adaptación del proceso a la realidad de la empresa debieran garantizar su aplicación en la vida real
<b>Es apoyado</b>	La gerencia no sólo debe firmar y promover los procesos definidos, sino que además debe asignar responsabilidad al personal y a los jefes de proyecto por su cumplimiento
<b>Es mantenido</b>	El proceso es revisado regularmente, para asegurarse que está adaptado para satisfacer las necesidades reales de los proyectos
<b>Está controlado</b>	Los cambios y puestas al día del proceso son revisados, aprobados y comunicados oportunamente a todos los usuarios
<b>Se verifica</b>	La gerencia mantiene mecanismos para asegurarse que todos los proyectos siguen el proceso vigente
<b>Se valida</b>	Se asegura que el proceso mantiene concordancia con los requisitos y estándares aplicables
<b>Se mide</b>	La utilización, los beneficios y el rendimiento resultante del proceso se miden regularmente
<b>Puede mejorarse</b>	Existen mecanismos y apoyo de la gerencia para revisar e introducir cambios en el proceso, de manera de mejorar su eficacia e incorporar nuevas metodologías

Tabla 2.3 Criterios que definen un proceso maduro

Un aspecto importante a tener en cuenta es que CMM-SW es un modelo descriptivo en el sentido en el que describe atributos esenciales (claves) que desean ser esperados para caracterizar a una organización en un determinado nivel de madurez. Este es un modelo normativo en el sentido en que detalla prácticas caracterizando los tipos normales de conducta que desean ser esperados cuando una organización este realizando proyectos. Por lo tanto, el intento es que CMM-SW presente un nivel de abstracción suficiente y que no sea restrictivo en como se implemente el proceso de software en una organización; este simplemente describe *atributos* esenciales de un proceso de software deben ser esperados.

Los niveles de madurez definidos por el CMM-SW se muestran en la tabla 2.4 donde se describen además sus características.

NIVEL DE MADUREZ	CARACTERÍSTICAS
<b>5 - OPTIMIZADO</b>	La organización está enfocada en una mejora de procesos continua. La organización tiene la forma de identificar debilidades y fortalecer el proceso proactivamente, con el fin de prevenir la ocurrencia de defectos.
<b>4 - ADMINISTRADO</b>	La organización pone metas cuantitativas para productos y procesos de software. La productividad y la calidad son medidas en actividades de proceso de software importantes sobre todos los proyectos como parte de un programa de medición organizacional.
<b>3 - DEFINIDO</b>	La organización ha definido los procesos de gestión de ingeniería de procesos de forma estándar y completa. Los proyectos adaptan el proceso común de acuerdo a sus características particulares. El personal usa la biblioteca de procesos para realizar su trabajo.
<b>2 - REPETIBLE</b>	La organización ha definido prácticas mínimas de gestión de proyectos. Algunos proyectos nuevos pueden repetir los éxitos logrados en proyectos anteriores. Se depende menos de los individuos. La gerencia genera estrategias proactivas y se realiza planeación de forma realista.
<b>1 - AD-HOC</b>	Los procesos varían según los individuos, quienes generalmente enfrentan crisis. En este ambiente es común encontrar caos y problemas predominantes. El éxito de los proyectos es dado por la casualidad, mientras que se asumen gran cantidad de riesgos. El personal vive preocupaciones constantes y generalmente se trabaja mas allá de lo necesario.

Tabla 2.4 Niveles de madurez del CMM-SW

Todos los Niveles de madurez con excepción del Nivel 1 están compuestos por Áreas Claves de Proceso (KPAs), las cuales identifican una agrupación de actividades y prácticas relacionadas, las cuales cuando se realizan en forma colectiva permiten alcanzar las metas del proceso.

Las prácticas que deben ser realizadas por cada KPA se organizan en cinco Características Comunes, las cuales constituyen atributos que indican si la implementación y la institucionalización de un proceso clave es efectivo, repetible, y duradero. La tabla 2.5 presenta las cinco características comunes definidas en el CMM-SW.

<b>CARACTERÍSTICA COMÚN</b>	<b>DESCRIPCIÓN</b>
<b>Compromiso para realización</b>	Describe las acciones que la organización debe tomar para asegurar que el proceso se establezca y sea permanente. Normalmente esto compromete políticas y liderazgo en el ámbito de la organización.
<b>Habilidad para realización</b>	Describe las condiciones previas que deben existir en el proyecto o en la organización para poder aplicar el procesos de software de forma efectiva. Esto generalmente involucra recursos, estructuras organizacionales, capacitación, entre otros.
<b>Actividades realizadas</b>	Describen los papeles y los procedimientos necesarios par implementar una KPA. Estas generalmente abarcan el establecimiento de planes y procedimientos, la ejecución de tareas, seguimiento del trabajo, el control de las salidas del proceso, y acciones correctivas.
<b>Mediciones y Análisis</b>	Ponen de manifiesto la necesidad de medir el proceso y analizar los resultados. Se incluyen ejemplos de mediciones que se podrían tomar para determinar el estado y la eficacia de las actividades realizadas.
<b>Verificación de la Implementación</b>	Presenta los pasos que se deben seguir para asegurar que las actividades se llevan a cabo de acuerdo al proceso establecido. Generalmente abarca las revisiones y auditorias efectuadas por la alta gerencia, los jefes de proyecto y el grupo de aseguramiento de la calidad del software.

Tabla 2.5 Características Comunes definidas en el CMM-SW

### 2.3.2 NTC – ISO 9003

La norma NTC-ISO 9000-3 es la guía para la implementación de la norma contractual ISO 9001, la cual es parte de las exigencias contractuales hechas por los clientes para la aceptación de un



producto de software. Esta norma nos plantea la estructura general de la aplicación de un sistema de calidad para desarrollar productos software [Ledezma 00]. La figura 2.4 indica la composición de la norma.

PARTES	ACTIVIDADES										
<b>Estructura del sistema de calidad</b>	Responsabilidad gerencial			Sistema de calidad				Acción correctiva			
<b>Ciclo de vida</b>	Revisión del contrato	Especificaciones	Planificación de desarrollo	Planificación de calidad	Diseño	Ensayo y validación	Aceptación	Duplicación, despacho, instalación	Mantenimiento		
<b>Procesos de soporte</b>	Administración de la configuración	Control de documentos	Aseguramiento de la calidad	Personal y recursos	Validación	Auditorías	Revisiones conjuntas	Solución de problemas	Registros de calidad	Compras	Entrenamiento

Figura 2.4 Composición de la Norma NTC-ISO 9000-3

### 3. PATRONES DE DISEÑO PARA COMERCIO ELECTRÓNICO

#### 3.1 INTRODUCCIÓN

Este capítulo contiene una referencia sobre los patrones de diseño de software más utilizados y mejor documentados, que constituyen una herramienta muy útil para la creación de aplicaciones software en la etapa de construcción. En [Gamma 95] se encuentra un estudio detallado de cada uno de los patrones de diseño referenciados en este capítulo.

#### 3.2 PATRÓN CATÁLOGO (CATALOG PATTERN)

**Clasificación:** Estructural [Fernández03]

**Intención:** El patrón catálogo organiza la información de productos ofrecidos en un sitio Web.

**Contexto:** Los sitios de comercio electrónico donde las compañías venden productos de diferentes tipos. Allí deben estar otras aplicaciones en el servidor para proveer estructura de procesos, facturación, inventario, y otras funciones relacionadas.

**Problema:** Los sitios de comercio electrónico venden una variedad de productos. Un problema importante es cómo organizar la información de los productos, dar una guía a los usuarios, y mejorar la atracción del sitio Web para que los usuarios estén contentos y retornen.

**Fuerzas:**

- Hay una gran variedad de productos que pueden ser vendidos
- Los productos y sus descripciones cambian frecuentemente
- Sin un buen soporte para la búsqueda y selección de productos los clientes no retornarán al sitio, la infraestructura deberá soportar una variedad de estas funciones
- Los productos relacionados deberán indicarse para que el cliente sea atraído a comprar más productos de éstos.
- Los catálogos no pueden ser rehusados, lo cual resulta en duplicaciones, pérdida de tiempo y esfuerzo.

**Solución:** Definir una clase catálogo como punto central para recopilar productos. Separar los diferentes aspectos de interés tales como descripción detallada de productos y productos relacionados en distintas clases. Usar un Patrón Observador (Observer Pattern) para buscar cambios y notificar los cambios a los clientes.

**Consecuencias:** Este patrón ofrece los siguientes beneficios:

- Provee la infraestructura necesaria para describir los productos convenientemente
- Es reusable. Este patrón es adaptable para varias clases de tiendas Web
- Puede ser combinado fácilmente con otros patrones, tales como el Patrón Personalización (Personalization Pattern), el Patrón Inventario (Inventory Pattern) [Fernandez 00], o el Patrón Proceso de Compra ( Shopping Process Pattern).

**Usos conocidos:**

Tiendas en Internet que ofrezcan diferentes clases de productos

Patrones relacionados:

- Patrón Contenedor/Contexto (Container/context pattern) [Coad 97]
- Patrón Gestor de Existencias (Stock mananer pattern) [Fernandez 00]
- Patrón Búsqueda (Search pattern).

**Estructura:**

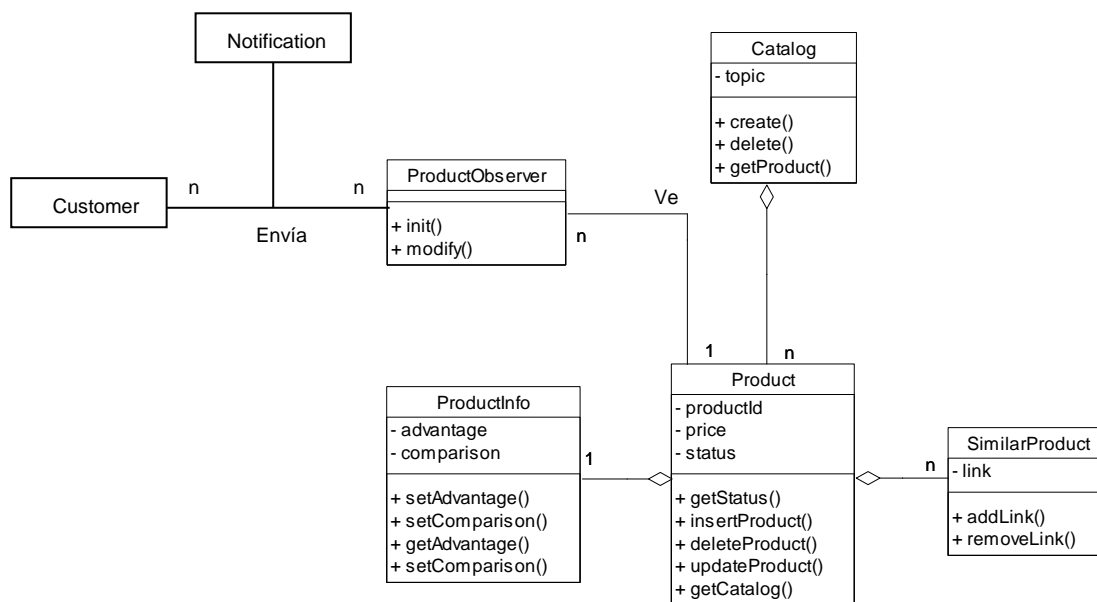


Figura 3.1 Estructura Patrón Catálogo

**Colaboraciones:**

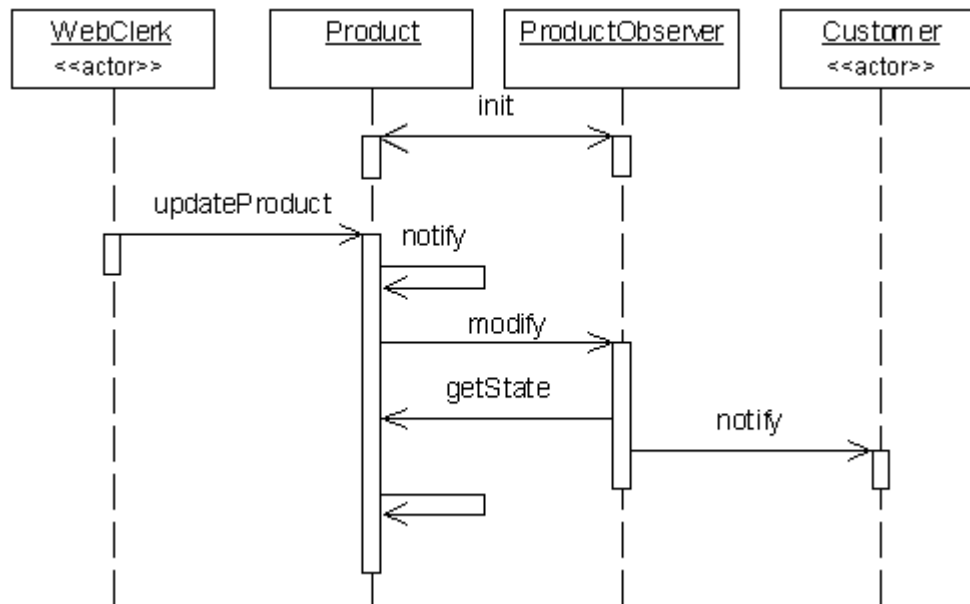


Figura 3.2 Diagrama de Colaboración Patrón Catálogo

### 3.3 PATRÓN PROCESO DE COMPRA (SHOPING PROCESS PATTERN)

**Clasificación:** Estructural [Fernández03]

**Intención:** El patrón catálogo organiza la información de productos ofrecidos en un sitio Web.

**Contexto:** Los sitios de comercio electrónico contienen muchos productos. Los clientes pueden seleccionar y comprar diferentes productos en la misma sesión. Un comprador busca los productos en los que él está interesado, escoge algunos que desea comprar y los adiciona a su lista de compras. El puede examinar y modificar su compra.

**Problema:** El proceso de compra debe tener bien definido sus pasos. Esto es necesario porque necesitamos mostrar al comprador dónde está en el proceso. El problema es ahora, describir el proceso de compra de una forma precisa.

**Fuerzas:**

- Los compradores pueden estar desorientados a través de los pasos del proceso de compra.
- Nosotros necesitamos mostrar al comprador dónde está en el proceso de compra.
- Los pasos del proceso deben necesitar ser cambiados para nuevos productos.
- A los compradores les gusta tener varias formas de pago.
- Algunos compradores deben ser tratados de forma diferente.
- Si la estructura de compra no es segura, los compradores no retornarán.

**Solución:** La metáfora más común para el proceso de compra está basada en el concepto de carro de compras, análogo a los carros usados en los supermercados. Un cliente debe tener varios carros de compras y cada carro deberá contener sus selecciones. Una orden se produce cuando el cliente decide comprar sus productos seleccionados y contenidos en el carro. Una factura es producida para cada orden de compra.

**Consecuencias:** Este patrón ofrece los siguientes beneficios:

- El patrón describe un proceso de compra abstracto. Este puede ser aplicado a varias tiendas Web.
- El patrón provee elementos comunes para construir tiendas en Internet.
- El patrón puede ayudar a reducir la complejidad del proceso de compra en Internet. Todos o algunos pasos del proceso pueden ser mostrados al usuario para su guía. La información del comprador no necesita ser suministrada para clientes registrados.

**Usos conocidos:** La mayoría de tiendas usan el concepto de carro de compras con una infraestructura similar a este patrón. Modelos específicos o código para patrones similares puede ser encontrado en [Baron 96], [Bollinger 00].

**Patrones relacionados:**

- Patrón de Configuración Dinámica (Dynamic Configuration pattern) [Lya 98], el cual ayuda a los compradores a seleccionar de una variedad de opciones y validar sus selecciones.
- Patrón Explícito (Explicit pattern), el cual ayuda a los usuarios a entender el proceso de compra.

Estructura:

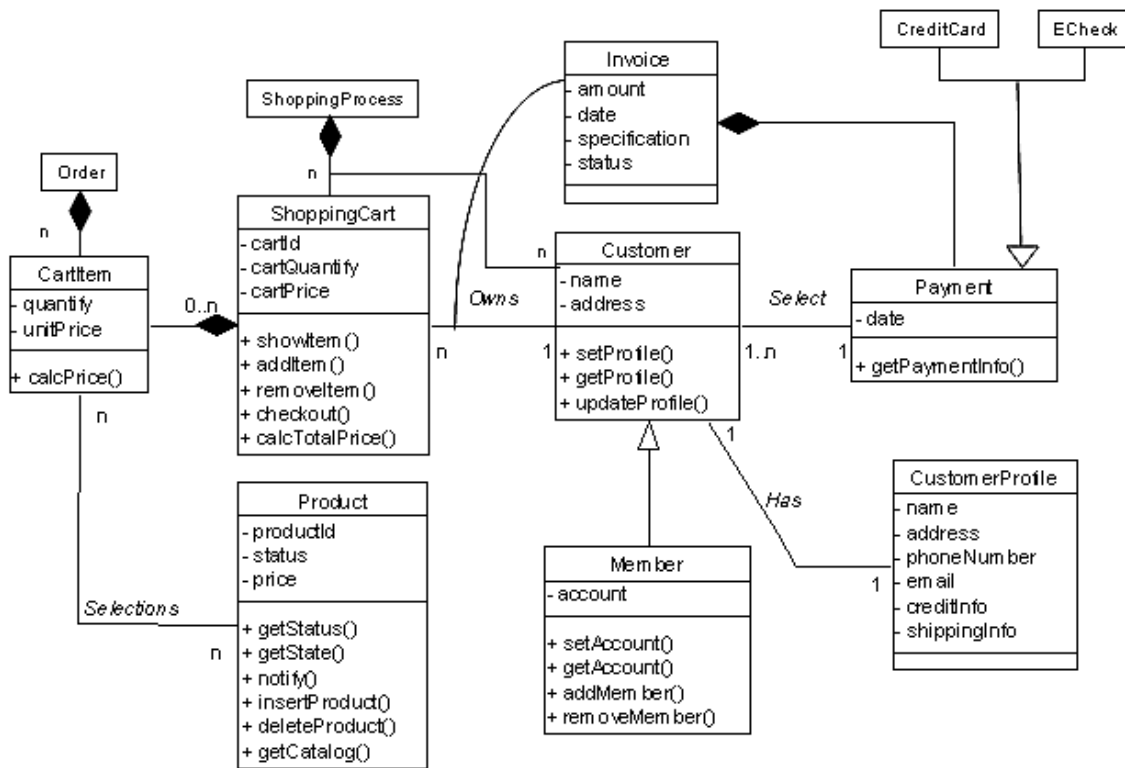


Figura 3.3 Estructura Patrón Proceso de Compra

**Colaboraciones:**

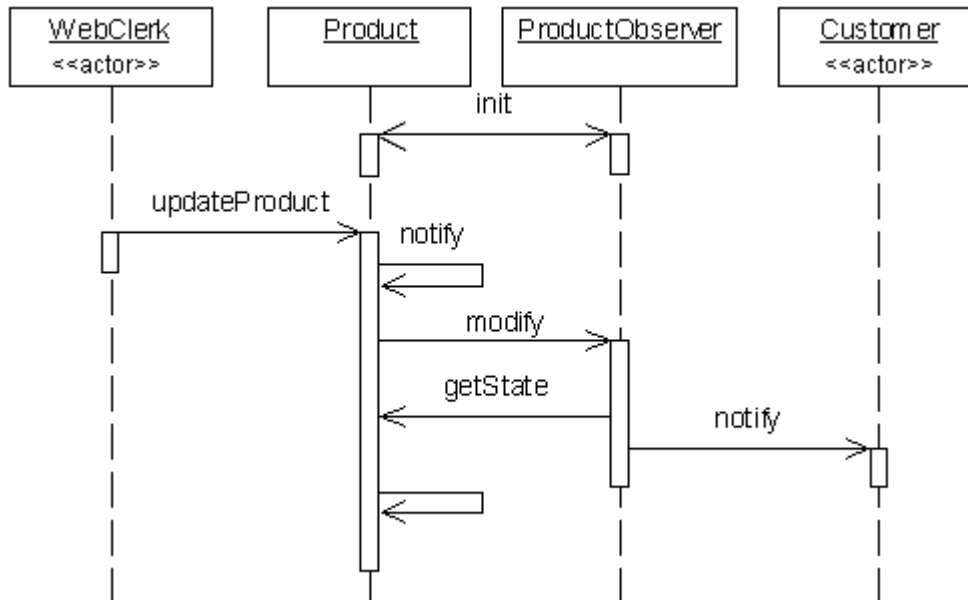


Figura 3.4 Diagrama de Colaboración Patrón Proceso de Compra

### 3.4 PATRÓN OBSERVADOR (OBSERVER PATTERN)

**Clasificación:** De Comportamiento [Hernández02]

**Intención:** Notifica automáticamente de los cambios de estado de un objeto a todos sus dependientes. Permite de forma dinámica implementar dependencias entre objetos, de forma que los objetos dependientes sean notificados de los cambios que se producen en los objetos de los que dependen.

**Motivo:** Encapsular la gestión de las dependencias dinámicas permite centralizar su gestión, realizar cambios futuros en una sola clase y reutilizar el mecanismo de interdependencia.

#### Aplicaciones:

- Cuando un sistema requiere que unos elementos sean conscientes de los cambios producidos en otros.
- Cuando la dependencia entre instancias se produce de forma dinámica, en tiempo de ejecución y no siempre.
- Cuando la dependencia se produce de un objeto hacia muchos y un sistema simple de eventos no sirve porque solo permite notificar a una sola instancia.
- En ocasiones se implementan sistemas de eventos mediante este sistema, por ejemplo en el API de Java, de forma que los objetos que notifican de eventos implementan el interfaz observable y los que reciben las notificaciones de eventos implementan el interfaz Observer. Esto permite que muchos objetos reciban eventos de otro objeto en lugar de los sistemas de eventos básicos que solo permiten notificar a un único objeto.

#### Estructura:

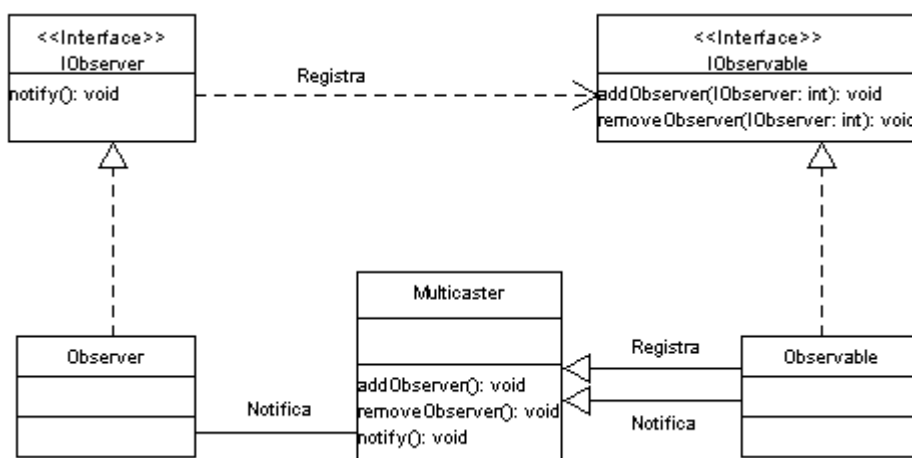


Figura 3.5 Estructura Patrón Observador

**Participantes:**

- *IObserver*. Interfaz orientado a las clases dependientes, declara un método `Notify()` que permite al objeto observado notificar de sus cambios a los que dependen de el.
- *IObservable*. Interfaz que implementarán las clases de las que se depende, declara los métodos `addObserver` que permite registrar una instancia dependiente y `removeObserver` para eliminar la dependencia. Esta característica le da el aspecto dinámico al sistema de notificación, ya que el enlace entre el objeto dependiente y del que se depende se realiza en tiempo de ejecución mediante el empleo de estos métodos.
- *Observer* y *Observable*. Son las clases concretas que implementan los interfaces anteriores.
- *Multicaster*. No es imprescindible, ya que sus funciones las puede llevar a cabo la propia clase *Observable*, pero delegar el registro de *observers* y la notificación en una sola clase permite reutilizar fácilmente el mecanismo.

**Consecuencias:**

- Permite enviar notificaciones desde un objeto a muchos, de forma dinámica y sin que las clases implicadas sean conscientes de las clases del resto.
- Un objeto observable puede ser a su vez *observer* respecto de otros.

En ocasiones se pueden producir consecuencias no deseables:

- Cuando existen muchos *observers* registrados se puede ralentizar notablemente el envío de notificaciones, también cuando aun siendo pocos los *observers* registrados estos producen a su vez indirectamente nuevas notificaciones en cascada.
- Cuando se producen ciclos de notificaciones, de forma que la notificación de un cambio en el objeto *Observable* produce de forma indirecta una cadena de notificaciones que vuelve a provocar un cambio en el objeto (reproduciéndose el ciclo) o simplemente el objeto observable está de forma indirecta dentro de un ciclo de *observers*. En ambos casos se produce un bucle infinito que termina cuando se agota la memoria (generalmente un *Stack Overflow*). Este problema no obstante no es inherente al patrón *Observer*, también se puede producir en la notificación de eventos normales en cuanto se produzca un ciclo de objetos que capturan eventos y los lanzan a su vez. Una forma trivial de evitar esto consiste en utilizar una bandera que indica que se está procesando una notificación, de forma que el objeto no aceptará nuevas notificaciones hasta que haya terminado con la anterior, de forma que se interrumpe el ciclo.



**Implementación:**

Generalmente el objeto Observable al notificar al objeto Observer le pasa una referencia de si mismo como parámetro del método notify (o de alguna otra forma) de manera que el objeto receptor de la notificación puede acceder a los atributos del objeto que observa. Hay que tener en cuenta que un mismo objeto observer puede haberse registrado como observador de varios objetos observables, de forma que cuando le llega la notificación necesita saber de quien le viene.

Independientemente de como implementes la solución a esto siempre llegarás a la conclusión de que las formas más prácticas requieren que la clase Observer conozca concretamente la clase Observable o se utilicen interfaces específicos para cada tipo de notificación según la clase origen, lo cual se presenta frecuentemente como una desventaja o inconveniente, pero la única forma de acceder a los atributos de una clase si no es conociéndola o utilizando un interfaz específico. Particularmente es preferible pasar una referencia del objeto Observable (o de una clase padre) como parámetro del método notify.

En ocasiones conviene reducir el número de notificaciones simplemente por que se van a producir más cambios, en este caso se espera a que se produzcan todos los cambios y se retrasa la notificación hasta que se han completado todos. Se puede implementar añadiendo dos métodos a la clase Observable para indicar el comienzo de cambios y el final de los mismos, de forma que las notificaciones están desactivadas entre tanto. Esta solución se complica si son múltiples objetos los que participan en ese conjunto de cambios, se puede utilizar un patrón Mediator en este caso.

**Usos conocidos:**

- Delegación de eventos en Java.
- ImageObserver
- Observer
- TileObserver

**Patrones relacionados:**

- Patrón Adaptador (Adapter pattern). Puede ser útil para permitir que clases que no implementen el interfaz IObserver puedan recibir notificaciones.
- Patrón Delegación (Delegation pattern). El patrón Observer utiliza el patrón Delegation si se utiliza la clase Multicaster.
- Patrón Mediador (Mediator pattern). Se puede utilizar para coordinar múltiples cambios de estado de un mismo Observable provocados por diferentes objetos (con la finalidad de reducir el número de notificaciones)

### 3.5 PATRÓN FACTORÍA ABSTRACTA (ABSTRACT FACTORY PATTERN)

**Clasificación:** De Creación [Hernández02]

**Intención:** Proporciona un Interfaz para crear familias de objetos sin especificar su clase de forma concreta

**Motivo:** Proporciona un interfaz común para crear diferentes familias de clases.

**Aplicaciones:** En muchas ocasiones existen diferentes recursos para realizar lo mismo, e incluso se presume que existirán nuevos recursos y queremos desarrollar sistemas compatibles con todos ellos, como en el caso de los escritorios KDE/GNOME, las librerías MOTIF/Windows/OpenView, las librerías OpenGL/DirecX. Si deseamos que nuestro software funcione sobre distintos recursos debemos abstraernos de que librerías utilizamos.

**Estructura:**

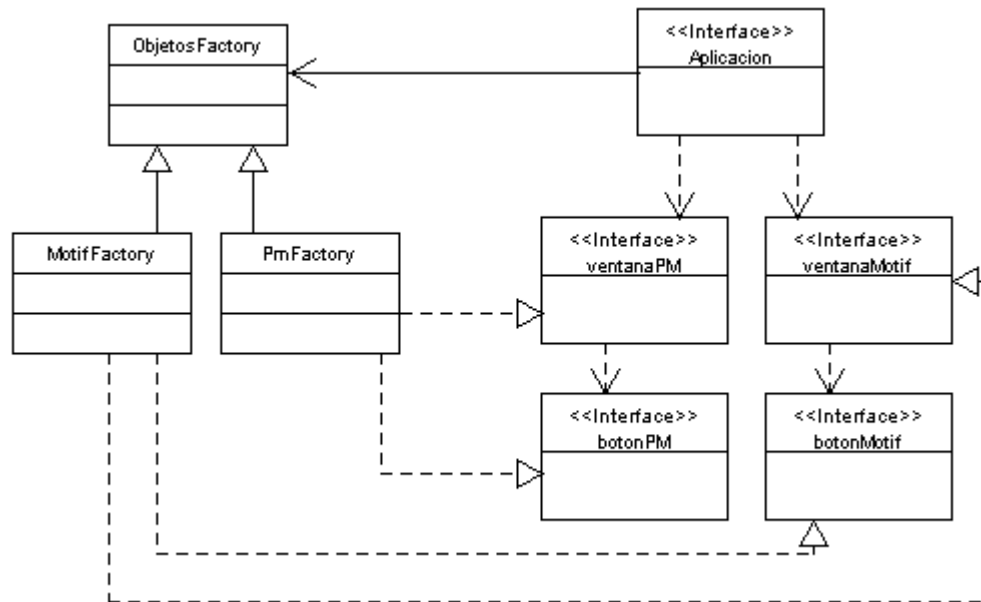


Figura 3.6 Estructura Patrón Factoría Abstracta

**Participantes:**

- *Factoría Abstracta:* Abstrae que clases se instancian. Encapsula la funcionalidad que se desea, es la vía de comunicación hacia las clases concretas instanciadas.
- *Factorías Concretas:* Clases encargadas de la instanciación de clases de una familia u otra.

**Colaboraciones:** Toda relación entre el exterior del sistema y las clases de la factoría se realiza únicamente a través de los métodos de la clase abstracta, se puede utilizar métodos virtuales en la clase abstracta para acceder a los métodos de las clases a instanciar.

**Consecuencias:**

- Ocultación de las clases de implementación. Esto también permite ofrecer clases sin mostrar su implementación, pudiéndonos reservar la posibilidad de cambiar la implementación en el futuro.
- Facilidad de sustitución de conjuntos de clases por otras equivalentes.
- Consistencia entre productos, al desarrollarse las aplicaciones con un mismo conjunto homogéneo de clases, dichas aplicaciones son más consistentes entre ellas.

**3.6 PATRÓN ESTRATEGIA (STRATEGY PATTERN)**

**Clasificación:** De Comportamiento [Hernández02]

**Intención:** Abstracción para seleccionar entre varios algoritmos.

**Motivo:** Encapsula varios algoritmos alternativos en diferentes clases y se ofrece un interfaz único (mediante una superclase) para acceder a la funcionalidad ofrecida, la elección del algoritmo se vuelve transparente para el cliente, pudiendo depender del objeto e incluso variar en el tiempo.

**Aplicaciones:**

- Cuando la aplicación debe ofrecer varios algoritmos alternativos para un mismo comportamiento.
- Una operación de "Guardar" un documento puede ser distinta según sea el destino elegido un directorio local o un servidor FTP remoto.
- La funcionalidad de comprobación de integridad de un archivo puede realizarse mediante CRC u otro algoritmo.
- Un sistema puede requerir cifrar un contenido mediante diferentes métodos de encriptación.

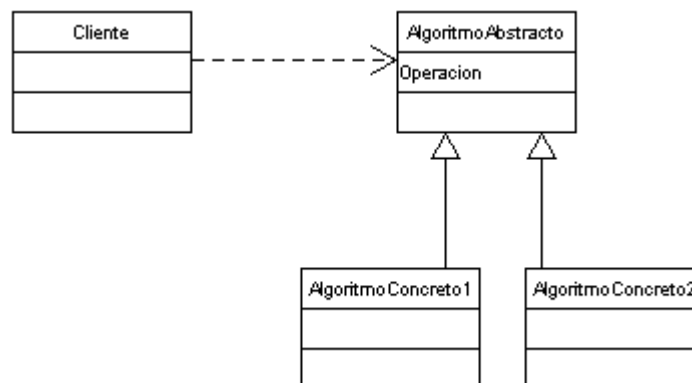
**Estructura:**

Figura 3.6 Estructura Patrón Estrategia

**Participantes:**

- El *Cliente* delega una operación en la superclase, abstrayéndose de los detalles de la misma.
- La superclase *AlgoritmoAbstracto* ofrece un interfaz común y oculta la elección del algoritmo empleado.
- Las clases *AlgoritmoConcreto* implementa cada una de las diferentes alternativas del algoritmo.

**Consecuencias:**

- La elección del algoritmo se realiza de forma dinámica en tiempo de ejecución.
- Se simplifica la clase cliente al independizarse de la decisión de qué algoritmo utilizar. Se reducen las estructuras *if* y *switch* en el cliente.
- Según el caso se puede incrementar la velocidad de ejecución de los objetos cliente al no tener que realizar ellos mismos el algoritmo (en entornos concurrentes).

**Implementación:** Si hay comportamientos comunes en varios algoritmos se puede colocar en una superclase de la que se deriven las diferentes alternativas. Informar al cliente de si se puede realizar la operación, en ciertos casos es posible que no se pueda utilizar ninguna de las alternativas, esto se puede implementar inicializando a null el objeto Strategy de forma que el cliente compruebe primero si es null antes de llamar a ninguna operación.

#### 4. AGENTES EN COMERCIO ELECTRÓNICO

El desarrollo ingente de las tecnologías de la información, la informática y el desarrollo de Internet como plataforma abierta para la interconexión de usuarios en cualquier parte del mundo, ha hecho que las empresas en todos los sectores de la economía empiecen a pensar en redefinir el paradigma de sus modelos de negocio tradicionales por unos más acordes con las nuevas tendencias. Es en esta coyuntura aparece el concepto de modelo de economía digital (*e-business*) donde se integran transacciones de múltiples tipos tales como negocio a negocio (B2B), negocio a consumidor (B2C), negocio a gobierno, entre otros conceptos. La conectividad dada por plataformas como Internet y las tecnologías emergentes están redefiniendo los patrones de compra y venta, y en general los comportamientos de consumidores e inversionistas del mundo.

Según los estudios de Jupiter Research se prevé que las transacciones de comercio electrónico de naturaleza B2C y B2B superarán los 7 trillones de dólares anualmente para el año 2005, de acuerdo a esto se puede prever que existe una oportunidad de negocio gigantesca para el ofrecimiento de productos y servicios en Internet. Sin embargo el éxito de esto está en la dislocación de los procesos de negocio existentes por unos nuevos, basados en modelos de negocio de costo efectivo, escalables, y esencialmente de la red (Net - Intrinsic). Este advenimiento de nuevas cadenas de abastecimiento, canales de distribución, y mercados dinámicos se espera que hagan uso de procesos computacionales distribuidos inteligentes que se denominan *agentes* [Moses 99].

Los agentes pueden entenderse como entidades software atómicas operando a través de acciones autónomas en nombre de un usuario (máquinas o personas) sin intervención humana constante [Moses 99]. Gracias al avance de tecnologías como Java y los correspondientes en el área de la Inteligencia Artificial (AI), el desarrollo de agentes software es actualmente una realidad que impacta en forma importante distintas facetas del comercio electrónico, en especial el que se realiza sobre plataforma Web sobre Internet, la tecnología de agentes ha contribuido a transformar la Web desde un modelo de publicación estático hacia un más refinado e inteligente.

Los agentes inteligentes pueden llevar acabo numerosas decisiones y tareas para resolver problemas en comercio electrónico que en gran parte de los casos se hace todavía con intervención humana tales como los diagnósticos, clasificación de datos, planeación, o negociación. Ellos pueden responder mensajes por correo electrónico, buscar en Internet

información útil, llevar a cabo comparaciones, e incluso comprar y vender productos y servicios [Wagner 02].

Una de las características de los agentes software que se está explotando actualmente en comercio electrónico es el concepto de movilidad, esta sin duda influenciada por tecnologías multiplataforma y de computación distribuida. Un agente móvil no está “amarrado” a al sistema donde está siendo ejecutado. Este es libre de para viajar entre los “hosts” de una red. El agente móvil una vez creado en un ambiente de ejecución, este puede transportar su estado y su código a otro ambiente en la red, donde este puede comenzar de nuevo su ejecución. Esta habilidad le permite al agente moverse hacia otro sistema el cual puede contener un *objeto* con el cual desea interactuar y por ende tomar ventaja de esta “corriendo” en el mismo ambiente del objeto. Debido a lo anterior los agentes móviles son útiles para el comercio electrónico, ya que se dan casos en que una transacción comercial puede requerir acceso en tiempo real a recursos remotos, como cotizaciones de mercancía y quizá una negociación agente-agente [Lange 99].

#### **4.1 CLASIFICACIÓN Y CARACTERIZACIÓN DE AGENTES PARA COMERCIO ELECTRÓNICO**

Para hablar de una posible clasificación y caracterización de los agentes software en comercio electrónico se podría utilizar la aproximación hecha por [Papazoglou 01], donde se habla de conceptos de agentes para negocios electrónicos (e-business). Es de aclarar que cuando se habla de comercio electrónico se está hablando también del área de *e-business*, ya que este último es un concepto más general que engloba al primero. Papazoglou identifica tres características básicas para los agentes en el campo de negocios electrónicos, como sigue.

*Delegación de habilidades:* Esto hace referencia a que el propietario o usuario de un agente delega una tarea específica al agente y este autónomamente ejecuta la tarea en nombre del usuario. Alternativamente un agente de negocios puede descomponer la tarea en partes delegadas a otros agentes, los cuales ejecutan las subtareas y se reportan ante el agente de negocios. El agente debe ser capaz de comunicarse con el usuario u otros agentes para recibir sus instrucciones y para proveer los resultados de las actividades.

*Protocolos y lenguajes de comunicación de agente:* A pesar de que los agentes son entidades autónomas, estos deben negociar con otros agentes para acceder a otros recursos y capacidades. Para habilitar la comunicación, la negociación, y organizar la comunicación entre los agentes, es útil definir un lenguaje que permita intermediar entre las diferentes peticiones. Como ejemplo se tiene el lenguaje KQML (Knowledge Query and Manipulation Language), el cual puede ser usado

para permitir a los agentes defender intereses en servicios de información, anunciar sus propios servicios, y delegar explícitamente tareas o solicitudes a otros agentes.

Habilidades de auto representación: Uno de los problemas más desafiantes para los agentes es expresar naturalidad y no-ambigüedad de negocio, además ejecutar los aspectos específicos del sistema y combinar todo esto en la aplicación final. Si todo esto se logra, se pueden obtener agentes que son dinámicos y re-configurables para facilitar la composición de aplicaciones distribuidas a gran escala, que esquematicen y especialicen procesos de negocio.

En ambientes en los cuales se deben relacionar diferentes tipos de agentes especializados, *ambientes multiagentes*, como es el caso del comercio electrónico, se hace necesario clasificarlos en diferentes categorías dependiendo de su funcionalidad y competencias. Papazoglou propone una clasificación conformando una jerarquía tipológica de agentes, la cual se puede ver en la figura 4.1

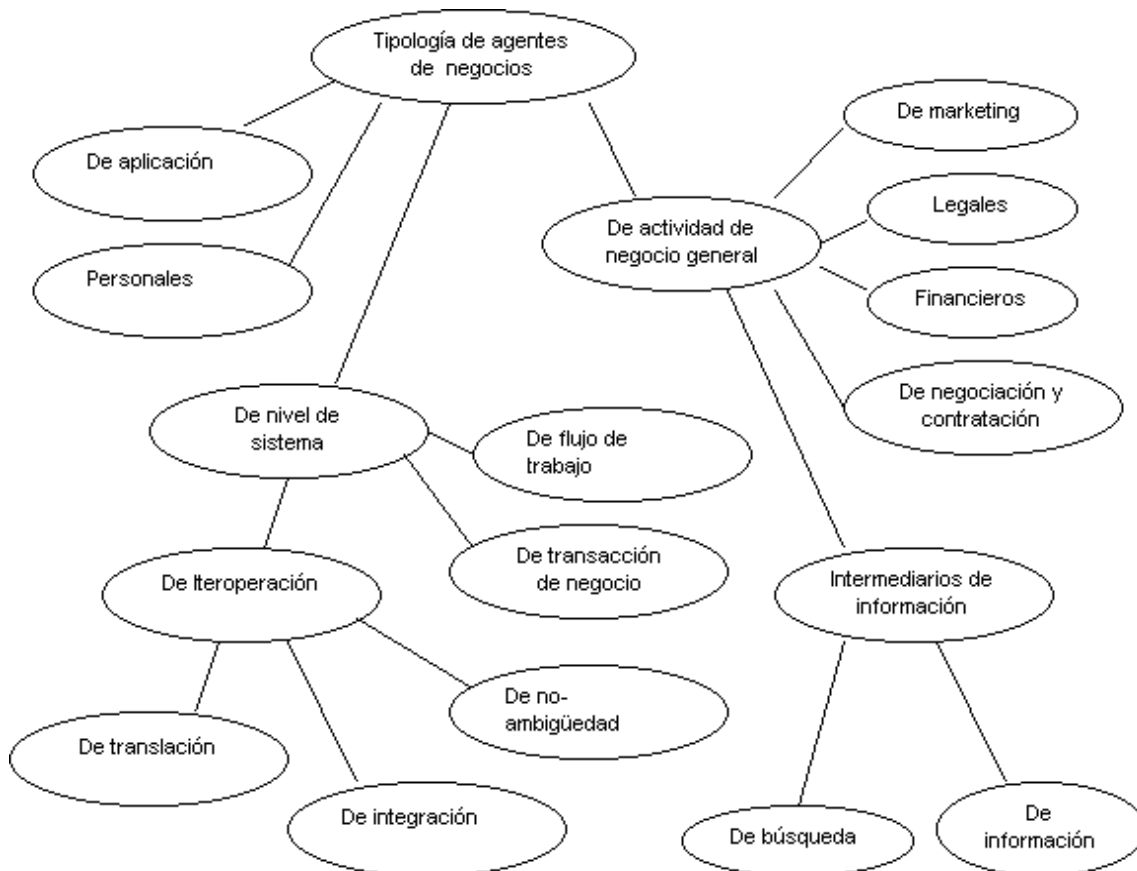


Figura 4.1. Clasificación de los agentes de negocios

A continuación se describen algunos de los tipos de agentes representativos.

Agentes de aplicación: Este tipo de agentes son de utilidad en ambientes de comercio electrónico B2B. Las aplicaciones de comercio electrónico de este tipo pueden considerarse como sistemas basados en la red que consisten de un gran número de agentes de aplicación. Cada agente es especializado en un área de experiencia (inventario, producción, distribución, entre otros) y proveen acceso hacia la información disponible y las fuentes de conocimiento en ese dominio y trabaja cooperativamente con otros agentes para resolver problemas complejos.

Agentes personales: También conocidos como agentes *interface*, estos trabajan directamente con los usuarios para ayudarlos con la presentación, organización, y administración del perfil de usuario, solicitudes, y colecciones de información. Un agente personal da a su usuario fácil y efectivo acceso para perfilar servicios especializados relacionados e información distribuida ampliamente sobre la Web. El agente del usuario observa y monitorea las acciones tomadas por el usuario en la interface y sugiere mejores caminos para ejecutar la tarea. Un ejemplo de estos agentes son los asistentes de búsqueda en la Web.

Intermediarios de información: Estos agentes proveen facilidades como la localización de información sobre fuentes Web o sobre otros agentes que son requeridos para solucionar un problema común, tales como ventas y distribución de procesamiento, servicios de páginas blancas y páginas amarillas, enrutamiento basado en el contenido, entre otros.

Agentes de negociación y contratación: Estos agentes pueden elegir negociar los términos de una transacción de negocios en cuanto a intercambio y pago. Los procesos de negociación pueden consistir de una conversación en secuencia, donde múltiples mensajes intercambiados acorde a un protocolo preestablecido. Algunos de los términos pueden consistir cubrimiento de la entrega, políticas de reembolso, plazos de pago, políticas de copyright, y otras más. Para el caso de contratación se pueden dar instrucciones para obligaciones, aceptación de las formas de pago, términos de pago, métodos para resolución de disputas, documentación necesaria, entre otras.

Agentes de transacciones de negocios: Una actividad clave en cadenas de valor integradas es la colecta, administración, análisis, e interpretación de una variedad de datos comerciales para tomar decisiones inteligentes y efectivas sobre transacciones. Algunos de los ejemplos que se pueden presentar están la recolección de referencias de negocio, coordinación y administración de estrategias de marketing, determinación de nuevas ofertas de productos, extensiones de créditos, y administración de riesgos de mercado. Las transacciones de negocio son usadas para intercambiar desde información de productos y propuestas de precios, hasta acuerdos legales y financieros.



Para el caso de transacciones para comercio electrónico, los agentes son usados para procesar, monitorear y controlar las transacciones, mediante la automatización de actividades. Por ejemplo estos agentes pueden controlar el flujo de trabajo dirigiendo un conjunto de transacciones electrónicas, o monitorear y hacer cumplir los términos y condiciones de contratos electrónicos.

Agentes de seguridad: Las comunicaciones en comercio electrónico deben ser protegidas por agentes diseñados especialmente para proveer servicios de seguridad. Los agentes en esta clasificación pueden, por ejemplo en ambientes B2B, recolectar información de partes seguras y controladas (trusted sources). Los agentes a este nivel pueden ser segmentados en subcategorías, tal como de autenticación, autorización, integridad de datos, confidencialidad, y no-repudiación.

## 4.2 AGENTES COMO INTERMEDIARIOS

Se tiene actualmente que la información de productos y vendedores está mas fácilmente accesible a través de Internet, los agentes pueden ser usados para automatizar algunas de las tareas que más tiempo consumen en el proceso de adquisición de productos y servicios como por ejemplo recolectar e interpretar información de comerciantes, tomar decisiones sobre productos y categorías, además de registrar la información de compras y pagos.

Una orientación importante para poder observar el rol que desempeñan los agentes en comercio electrónico es la propuesta por [Maes 99], el cual propone utilizar un marco de referencia común basado en un estudio de la conducta del consumidor mientras compra (CBB – consumer buying behavior) la cual incluye las acciones y decisiones que se involucran cuando se compra y se usa bienes o servicios. Gracias a lo anterior se distinguen seis etapas fundamentales del proceso de compra:

*Need identification:* Se caracteriza por que en esta etapa el comprador de alguna necesidad insatisfecha. El comprador puede ser motivado a través de información de productos.

*Product Brokering:* Incluye la recuperación de información para ayudar a determinar *qué comprar*. La información recuperada incluye una evaluación de alternativas de productos basadas en un criterio provisto por el comprador. Como resultado de esta etapa se obtiene un “conjunto de consideración” de productos.

*Merchant Brokering:* En esta etapa se combina el “conjunto de consideración” de la etapa anterior con información específica del comerciante para ayudar a determinar *desde donde comprar*. Esta etapa incluye la evaluación de las alternativas de comerciante basados en un criterio provisto por el comprador tales como precio, garantía, tiempo de envío, reputación, entre otros.

*Negotiation:* Se considera cómo fijar los términos de la transacción. La negociación varía en duración y complejidad dependiendo del mercado. Por ejemplo en el mercado tradicional de bienes al por menor, los precios y otros aspectos son generalmente fijos, no dando lugar a “salones” de negociación. De otro lado, en mercados como automóviles y arte, el precio de negociación y otros aspectos del negocio son integrados al proceso de compra.

*Purchase and delivery:* Señala la terminación de la etapa de negociación o puede ocurrir algún tiempo después (en otra orden). En algunos casos puede ocurrir la disponibilidad de pago, por ejemplo el caso de solo efectivo, o de las opciones de envío pueden influenciar las opciones de Product Brokering y Merchant Brokering.

*Product service and evaluation:* Esta incluye los servicios de productos post venta y servicio al cliente.

Es de aclarar que estas etapas solo representan una aproximación a la conducta compleja que puede presentar un cliente cuando compra productos y servicios. Algunas de ellas pueden estar superpuestas, y la transición de una a otra puede ser no lineal. La proposición de las etapas se utiliza como medio para observar el papel de los agentes en el proceso de adquisición de bienes o servicios.

Las características inherentes a los agentes tales como personalización, ejecución continua, y naturaleza autónoma, los hace ideales para afrontar las conductas complejas de los consumidores. Algunas de las cuales involucran conceptos tales como filtrado y recuperación de información, evaluaciones personalizadas, coordinación compleja, e interacciones basadas en el tiempo. A continuación se presenta una identificación de agentes software para comercio electrónico sobre Web hecha por [Maes 99], para las diferentes etapas del proceso de compra.

En la etapa de identificación de necesidades (Need Identification), la tecnología de agentes puede ser útil para automatizar y asistir en la elección de un producto o servicio. Generalmente los agentes en esta etapa están dirigidos a ayudar en compras de tipo repetitivo o predecibles tales como los hábitos. Uno de los ejemplos para este tipo de agentes son los denominados *monitors*, que son programas que se encuentran continuamente corriendo y que monitorean un conjunto de flujos de datos y toman decisiones cuando condición pre-especificada ocurre. Se encuentran abundantes ejemplos de este tipo de agentes en sitios de comercio electrónico, por ejemplo el agente de notificación “Eyes”, perteneciente a Amazon.com ([www.amazon.com](http://www.amazon.com)), que monitorea el catálogo de libros de los clientes y les notifica cuando ciertos eventos ocurren que pueden ser de su interés, por ejemplo cuando un libro de un cierto autor está disponible.

Después de que se presenta la identificación de una necesidad de compra, el comprador determina qué comprar a través de una evaluación crítica de la información recuperada de los productos

(Product Borkering). Agentes que pueden ser útiles en esta etapa están por ejemplo *PersonaLogic* ([www.personalogic.com](http://www.personalogic.com)), que permite limitar una lista de productos a los que mejor reúnan las necesidades del consumidor, ayudándoles a definir un conjunto de características. El sistema agente PersonaLogic filtra información no deseada dentro de un dominio dado después que el comprador especifica las restricciones para las características del producto. Después de dadas las restricciones, un "motor" de satisfacción de restricciones retorna una lista de productos que satisfacen todas las restricciones más importantes en orden descendente de satisfacción. A los sistemas de agentes que se basan en este mecanismo se les dice que utilizan técnicas basadas en restricciones (constraint-based techniques).

Otro ejemplo de sistemas agentes que ayudan en la etapa de Product Borkering está *Firefly* ([www.firefly.com](http://www.firefly.com)), el cual en lugar de filtrar productos basados en características, recomienda productos a través de un mecanismo de recomendación automático llamado *filtrado colaborativo* (collaborative filtering). El sistema primero compara el índice de productos del comprador con aquellos de otros compradores. Después de identificar los vecinos cercanos del comprador, o usuarios con gustos parecidos, el sistema recomienda los productos de los vecinos valorados en forma alta. Firefly usa las opiniones de personas dispuestas en gustos para ofrecer recomendaciones de productos como música y libros.

La etapa de selección de comerciante (Merchant Brokering) tiene por objetivo comparar las alternativas sobre comerciantes. Un ejemplo de agentes que actúan a este nivel esta *BargainFinder* ([www.bf.cstar.ac.com/bf](http://www.bf.cstar.ac.com/bf)), el cual es un agente de compra para comparaciones de precio en línea. En este caso, dado un producto específico, este agente mira precios desde por lo menos nueve sitios Web de comerciantes mediante solicitudes parecidas a la que realizan los navegadores Web. Sin embargo este sistema utilizado por BargainFinder tiene inconvenientes ya que a algunos comerciantes no les interesa participar en comparaciones de precio, por lo tanto habilitan mecanismos que boquean las solicitudes hechas por el agente. Para evitar este problema apareció otro agente de compras llamado *Jango* ([www.jango.excite.com](http://www.jango.excite.com)) el cual resuelve el problema del bloqueo por parte de los comerciantes haciendo aparecer las solicitudes de precio como si se originaran desde el navegador Web de un cliente y no desde un sitio centralizado como en el caso de BargainFinder. Esta técnica permite que los consumidores comparen precios desde numerosos catálogos electrónicos de comerciantes sin importar si los comerciantes desean poner sus precios a comparación.

En la etapa de negociación (Negotiation), los participantes ajustan los términos de la transacción. La negociación es un componente clave del comercio electrónico. Cuando se hacen negociaciones automáticas, los agentes encuentran y preparan contratos en nombre de las partes del mundo real a quienes representan. Esta automatización ahorra tiempo de negociación humano, además los

agentes computacionales son a menudo mejores encontrando tratos en escenarios combinatorios y estratégicamente complejos [Sandholm 99].

Para el caso del mercado de bienes al por menor, se tiene que los precios generalmente son fijos, no dejando libertad a que los consumidores hagan ofertas sobre los productos. Sin embargo, la tecnología de agentes puede ser usada en casos específicos como “casas” de subastas en la Web, o sitios que venden productos restaurados y de segunda mano. Entre algunos de los ejemplos de sistemas agentes que intervienen en la etapa de negociación esta *AuctionBot* ([www.auction.eecs.umich.edu](http://www.auction.eecs.umich.edu)), el cual es un servidor de propósito general para subastas en Internet desarrollado por la Universidad de Michigan. Los usuarios de AuctionBot pueden crear nuevas subastas a partir de la selección del tipo de subasta requerido y el establecimiento de parámetros tales como tiempos de clareo, métodos para resolver las ofertas, y el número de vendedores permitidos. AuctionBot provee una interfase de programación de aplicación para que los usuarios puedan crear sus propios agentes software para automáticamente competir en el centro de compras (marketplace) de AuctionBot.

Otro ejemplo representativo de agentes involucrados en la etapa de negociación es *Kasbah* ([www.kasbah.media.mit.edu](http://www.kasbah.media.mit.edu)) un sistema multiagente para transacciones C2C (Customer-to-Customer). Los usuarios que desean comprar o vender, crean un agente, dando a este alguna estratégica dirección, para enviarlo hacia el interior de un centro de compras. Los agentes Kasbah proactivamente buscan potenciales compradores o vendedores y negocia con ellos en representación de su propietario. La meta de los agentes Kasbah es completar un trato aceptable en representación de sus usuarios, sometido a un conjunto de restricciones dadas por ellos tales como precio inicial, los precios más altos y más bajos aceptables, la fecha para la cual completar la transacción, la variación del precio con el tiempo, entre otros. Kasbah también ofrece otros servicios como heurísticas de negociación y políticas de seguridad.

#### **4.3 BENEFICIOS DEL USO DE AGENTES EN COMERCIO ELECTRÓNICO**

Según lo tratado anteriormente el uso de agentes en comercio electrónico tiene muchas aplicaciones en los múltiples procesos que componen esta actividad, estos son algunos de los beneficios que se pueden obtener al utilizar la tecnología de agentes en este campo.

*Costos de Transacción:* uno de los beneficios que se puede obtener es bajar los costos de transacción. Esto se logra ya que muchos de los procesos transaccionales en comercio electrónico pueden ser automatizados. Si para cerrar un negocio se requiere negociación, búsqueda de información, o actividades similares, esto puede requerir inteligencia, por lo tanto el uso de agentes puede ser ideal para esto.

*Tiempo de ciclo:* en algunas aplicaciones de comercio electrónico, un tiempo de ciclo es absolutamente crucial. Clientes de firmas de compra y venta en Internet desean instantáneas ejecuciones de órdenes, y también esperan respuestas en muy poco tiempo. Una de estas firmas debe ser hábil para proveer estos niveles de servicio independientemente del volumen del mercado. De hecho, en días de alto volumen, el número de solicitudes de información puede ser a veces desproporcionadamente alto respecto a los días de transacción normal. Por lo tanto las firmas deben ser capaces de responder a estos picos de carga, mientras idealmente no se descuidan de los periodos de disminución de carga. Es aquí donde los agentes pueden entrar a clasificar solicitudes y responder rutinas de averiguación, esto puede bajar significativamente el volumen de transacción y proveer alto nivel de servicio en periodos pico [Wagner 02].

*Cerrar un trato:* los agentes pueden incrementar la eficiencia en transacciones de comercio electrónico y mejorar su eficacia. La habilidad para cerrar un trato vía un agente permite a una empresa hacer ventas que son de otra forma imposibles. Un pequeña tienda electrónica puede de repente proveer 24 horas de servicio al cliente y ventas a escala global, sin importar las zonas horarias.

*Precio de compra:* la comparación de compra sobre Internet se ha convertido en una de las aplicaciones más populares de los agentes. Los agentes pueden buscar el precio más bajo ahorrando esfuerzo al cliente.

## REFERENCIAS

[Nieto 01] M. Nieto Santisteban. "Ingeniería Web. Construyendo Web Apps". I Jornadas de Ingeniería Web. 2001.

[García 01] C. García. "El Modelo de Capacidad de Madurez y su Aplicación en Empresas Mexicana de Software". Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas -Puebla. Mayo. 2001.

[Boehm 76] B. W. Boehm, Qualitative evaluation of software quality, Proc. 2 nd ICSE, 1976, pp 592-605.

[McCall 77] McCall, J.A. Richards, P.K. Walters, G.F. "Factors in Software Quality", RADC. 1977.

[Olsina 99] L. A. Olsina. "Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web". Universidad Nacional de La Plata - Argentina, Noviembre de 1999.

[Bevan 99] Nigel Bevan, Quality in Use: Meeting User Needs for Quality, Journal of System and Software, 1999.

[Paulk 93] M. Paulk, B. Curtis, M. Chrissis, Ch. Weber. Capability Maturity Model for Software, v 1.1. CMU/SEI. 1993

[Ledezma 00] A. Ledezma. "Aseguramiento de la calidad del software". Colegio Mayor del Cauca. 2000.

[Guerrero 00] L. Guerrero. "El Modelo de Madurez de Capacidades". [http://www.geocities.com/SiliconValley/Lab/3629/CMM\\_Espanol.htm](http://www.geocities.com/SiliconValley/Lab/3629/CMM_Espanol.htm). 2000.

[Gamma95] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. 1995

[Fernández03] Eduardo B. Fernandez, Yi Liu, and RouYi Pan. Patterns for Internet shops. 2003

[Fernandez 00] E.B. Fernández y X. Yuan, An analysis pattern for reservation and use of entities.

[Baron 96] C. Baron y B. Weil, Implementing a Web Shopping Cart. Dr. Dobbs Journal, Septiembre 1996, 64-69 y 83-95.

[Bollinger 00] G. Bollinger y B. Natarajan, Build an e-commerce shopping cart. Java Pro, Junio 2000, 38-50.

[Coad 97] P. Coad, Object models: Strategies, patterns, and applications. 2<sup>nd</sup> Edición, Yourdon Press, 1997

[Hernández02] David Hernández Tejada. Guía de Patrones de Diseño. [www.teleprogramadores.com](http://www.teleprogramadores.com) 2002

[Moses 99] Moses Ma. "Agentes in E - commerce". Communications of the ACM. Vol 42, No 3. Marzo de 1999.

[Wagner 02] Ch. Wagner, E. Turban. "Are Intelligent E-Commerce Agents Partners or Predators". Communications of the ACM. Vol 45, No 5. Mayo de 2002.

[Lange 99] D. Lange, M. Oshima. "Seven good reasons for mobile agents". Communications of the ACM. Vol 42, No 3. Marzo de 1999.

[Papazoglou 01] M. Papazoglou. "Agent – Oriented Technology in Support of E-Business". Communications of the ACM. Vol 44, No 4. Abril de 2001.

[Maes 99] P. Maes, R. Guttman, A. Moukas. "Agents that buy and sell". Communications of the ACM. Vol 42, No 3. Marzo de 1999.

[Sandholm 99] T. Sandholm. "Automated Negotiation". Communications of the ACM. Vol 42, No 3. Marzo de 1999.