Revista de Gestão da Tecnologia e Sistemas de Informação
Journal of Information Systems and Technology Management
Vol. 3, No. 2, 2006, p. 143-162
ISSN online: 1807-1775

METODOLOGÍA PARA LA GERENCIA DE LOS PROCESOS DEL NEGOCIO SUSTENDA EN EL USO DE PATRONES

BUSINESS PROCESS MANAGEMENT WITH THE USE OF PATTERNS METHODOLOGY

Pedro Bonillo
Centro ISYS, Facultad de Ciencias, UCV, Caracas, Venezuela

ABSTRACT

Present companies require of complex businesses models with an organizational structure, processes and systems that must explicitly be designed. The task to design these business models is clearly interdisciplinary, since it requires knowledge of business development, different processes that happen in the company and of the processes management and technological applications. In software engineering context, it would be advisable to be able to count on a system of methods, tools and techniques that allow reusing the best practices during software development process, according to each one of the processes that are implemented in each domain. On this basis, in this work we propose an integral referential theoretician framework and a methodology that includes from requirements analysis to processes monitoring, supporting analysis, design, and modeling and configuration stages, through patterns use. The proposal methodology is conformed by two macro-processes: the first one related to the creation of the process itself. The second one corresponds to the administration, and includes: maintenance, administration of the process in production and the monitoring through management indicators.

Keywords: Methodology, Business Management Processes, Patterns.

RESUMEN

Las empresas actuales requieren de modelos de negocios complejos con una estructura organizacional, procesos y sistemas que deben ser diseñados explícitamente. El trabajo de diseñar estos modelos de negocio es claramente interdisciplinario, ya que requiere conocimientos de desarrollo del negocio, los diferentes procesos que ocurren en la empresa y de la gerencia de los procesos y las aplicaciones tecnológicas. En el ámbito de la ingeniería de software sería conveniente poder contar con un sistema de métodos, herramientas y técnicas

Recebido em/*Manuscript first received*: 03/02/2006 Aprovado em/*Manuscript accepted*: 15/07/2006

Endereço para Correspondência/*Addresss for Correspondence*

Pedro Bonillo, Ingeniero en Computación. Magíster en Ingeniería de Sistemas. USB., Magíster en Gerencia de las Finanzas y los Negocios. Candidato Doctor em Gerencia. UNY. Candidato Doctor en Ciencias de la Computación. Mención Ingeniería de Software. UCV. Remedy Skilled Professional. ITIL Foundation Certified. PPI Candidato 6502.CIV 11049. Asesor Tecnología y Arquitectura, Gerencia de Arquitectura y Tecnología CANTV, Cortijos de Lourdes, Edif. Cortijos II, Piso 2, Apdo. 1046. Centro ISYS, Facultad de Ciencias, UCV, Apdo. 48097, Los Chaguaramos 1041-A, Caracas, Venezuela. Phone: + 58 212 5000255 Email: pedro_bonillo@ciens.ucv.ve

ISSN online: 1807-1775
Publicado por/*Published by*: TECSI FEA USP – 2006

que permitan reutilizar las mejores prácticas durante el proceso de desarrollo de software según cada uno de los procesos que se implementen en cada dominio. En base a esto, en este trabajo se realiza una propuesta de marco teórico referencial integral y una metodología que abarca desde el análisis de los requerimientos hasta el monitoreo de los procesos, apoyando las etapas de análisis, diseño, modelaje y configuración, a través del uso de patrones. La propuesta metodológica está conformada por dos macro-procesos: uno relacionado con la creación del proceso en sí mismo y otro que corresponde a la administración, y comprende: el mantenimiento, administración del proceso en producción y el monitoreo a través de indicadores de gestión.

Palabras Claves*: *Metodología, Gerencia Procesos de Negocio, Patrones.

1. Introduccion

El principal objetivo de este trabajo es proponer una metodología para la gerencia de los procesos del negocio (BPM) [Davenport93] sustentada en el uso de patrones [Alexander et al. 77] [Coad92] [Coplien93] [Gamma et al. 95] [Buschmann et al. 96] [Flowler97] [Coad et al. 99]. En este trabajo nosotros proponemos una taxonomia de patrones [Sarver00] y su representación a través de un Lenguaje de Definición de Arquitecturas [Vestal93] (ADL) con respecto a una arquitectura de procesos, servicios y objetos canonicos, además extendemos la especificación de los patrones [Acosta et al. 04] a fin de poder medir su calidad durante el proceso de desarrollo y al producto de software obtenido a través de Estilos de Arquitectura Basados en Atributos (ABAS) [Kazman et al. 04] y como modelos de calidad ISO14-598 [ISO/IEC14598-3 99] e ISO-9126[ISO/IEC 9126-1 01]. Tomando en cuenta esta combinación de metodos herramientas y tecnicas se proponen un conjunto de pasos que en el ambito de BPM permiten identificar los procesos claves, modelarlos y analizarlos, simularlos, implantarlos de forma auto-asistida (tanto los nuevos procesos como sus versiones), evaluarlos, monitorearlos y mejorarlos.

Procesos de Desarrollo de Software

En los últimos años se han estudiado dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y los métodos ágiles. La diferencia fundamental entre ambos es que mientras los métodos pesados intentan conseguir el objetivo común por medio de orden y documentación, los métodos ágiles lo hacen mejorando los procesos de comunicación directa e inmediata entre las personas que intervienen en el proceso. Los procesos de desarrollo a considerar en esta propuesta son: Rational Unified Processs (RUP) [Kruchten00], XP (eXtreme Programming Project) [Beck99] y (Feature Driven Development) FDD [Batory03].

Si el proyecto es suficientemente grande como para compensar la adaptación, se puede decir que RUP es una buena base para el proceso, ya que permite conseguir una mayor y mejor estructura y disciplina del proceso de desarrollo. Una buena posibilidad de reducir el trabajo a realizar es la reutilización de modelos, procesos, etc. ya definidos en implementaciones previas de RUP en distintos ámbitos. Con relación a la arquitectura, XP con las metáforas del sistema, intenta determinar una arquitectura óptima en etapas tempranas del desarrollo. FDD, aún cuando se centra en la calidad, deja todo el peso de las decisiones arquitecturales al arquitecto principal, pero no especifica como estas decisiones tienen relación con la calidad del sistema en

desarrollo.

Por otra parte se tiene un problema generalizado para todos los procesos de desarrollo: la selección de la arquitectura adecuada o combinación de diferentes estilos arquitecturales para un sistema de software es un problema que aún no ha sido resuelto y que se ha tratado ampliamente en la literatura. El crecimiento de la complejidad de los sistemas, construidos usualmente a través de la integración de componentes (tema que se discutira en la siguiente sección), incrementa la necesidad de obtener enfoques mas rigurosos que conduzcan este proceso de decisión, todos los procesos de desarrollo tienen ausencia de una clara relación entre los patrones, los componentes, la arquitectura y las características de calidad asociadas a la misma, RUP por su parte, no tiene una asociación de los requerimientos no funcionales con los casos de uso, una mala selección de los principales casos de uso afecta la arquitectura del sistema, el modelo de prueba utilizado para evaluar la arquitectura no tiene guías precisas para determinar esta relación. [Losavio et al. 04].

Construcción de Software Basado en Componentes

Cuando se habla de componentes, se pueden encontrar algunas definiciones relacionadas con la implementación, donde se encuentran aquellas que entienden por componente un paquete coherente de código que: (i) puede ser desarrollado y distribuido independientemente, (ii) tiene interfaces explícitas y bien especificadas para el servicio que ofrece, (iii) tiene interfaces explícitas y bien especificadas para el servicio que espera de otros componentes, y (iv) puede ser compuesto junto con otros componentes, quizás extendiendo alguna de sus propiedades, pero sin modificar al componente propiamente dicho [D´sousa et al. 99]. Una definición más general la ofrecen Jacobson, Griss y Jonsson, (1997) [Jacobson et al. 97] quienes lo definen como un artefacto que ha sido desarrollado específicamente para ser reutilizado. En este caso un componente podría ser tanto un caso de uso como cualquier otra entidad reutilizable que surja durante el proceso de desarrollo y que sea utilizado en cualquier actividad, siempre que no requieran conocimiento del software que lo utiliza.

Los componentes pueden ser objetos en el sentido usual de la Orientación a Objetos (OO), excepto que satisfacen guías adicionales dirigidas a hacerlas autocontenidas. Usan otros componentes mediante agregación y por lo general interactúan con otras componentes a través de los eventos [Braude03].

La construcción de software basado en componentes persigue tres objetivos principales: la reutilización, la adaptación y la extensión (la reutilización implica la adaptación y extensión):

1. Un componente es reutilizable en la medida en que sus servicios pueden ser utilizados por otro software.
2. Un componente es adaptable si su proveedor ha previsto los posibles cambios que puede sufrir dicho componente.
3. Un componente es extensible si su proveedor proporciona los mecanismos para modificar los servicios que ofrece el componente.

En cuanto a su relación con los procesos de desarrollo antes descritos, se tiene que:

- Con relación a la obtención de requisitos, tres son los aspectos principales: (i) el análisis vertical, centrado en un dominio o un área de negocio concreta; cuyo objetivo es que los componentes resultantes puedan convertirse en estándares para cualquier aplicación desarrollada posteriormente en ese dominio y que apunte hacia su reutilización [Szyperski97]; (ii) el Análisis horizontal, realizado de forma genérica para dar servicio a un amplio rango de aplicaciones, sin restringirse a un dominio de negocio dado; y (iii) Análisis específico, realizado en un dominio concreto [Allen et al. 98], para obtener componentes ad hoc donde el énfasis no se hace tanto en la reutilización si no más bien en la extensión.
- Durante la obtención de requisitos se han identificado todas las funcionalidades que deben ser soportadas, pero la distribución de dichas funcionalidades puede darse de inmediato, o puede construirse adaptando los componentes existentes. El siguiente aspecto corresponde a la partición de componentes, la cual puede realizarse a través de: los casos de uso, los patrones de diseño, las entidades del dominio, la evolución prevista del sistema; y, los componentes ya existentes.
- En cuanto a la interacción de componentes, se dice que es directa (interacción simple) cuando el servicio ofrecido se ajusta a las formas y necesidades del servicio requerido. Si las formas no son las adecuadas, es necesario realizar previamente un proceso de empaquetamiento ("wrapper"). [Allen et al., 1998].

Actualmente, el uso de patrones en el proceso de construcción de software es uno de los temas más tratados, generando un gran interés entre investigadores y desarrolladores. Todavía existen discrepancias entre los investigadores a la hora de definir qué es un patrón, por lo que es bastante difícil encontrar definiciones de patrón que sean idénticas. En el libro de Fowler (1997) se encuentra una definición genérica interesante: "Un patrón es una idea que ha sido utilizada en un contexto práctico y que probablemente será útil en otros" [Flowler97]. El término idea expresa que un patrón puede ser cualquier "cosa". La expresión contexto práctico refleja el hecho de que se desarrollan (algunos autores prefieren: descubren) gracias a la experiencia práctica de proyectos reales. Teniendo en cuenta esta definición general de patrón se puede afirmar que los mismos, pueden expresarse a través de componentes y estos a su vez representan funcionalidades que se implementan a través de las diferentes aplicaciones. (Figura 1)

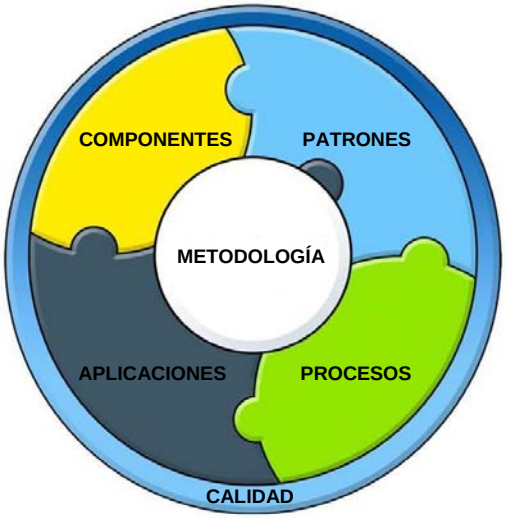

Figura 1: Relación entre Patrones, Componentes, Aplicaciones, Procesos, Metodología y Calidad.

En esta figura las muescas significan que se expresa "a través de", de tal forma que los patrones se expresan a través de componentes, las aplicaciones a través de componentes y patrones, los procesos a través de las aplicaciones y los patrones, todo esto en el marco de una metodología (que es el sistema que une los conceptos) y donde cada elemento debe tener asociada una medición de calidad.

El concepto de patrón en la ingeniería de software, desde sus inicios fue planteado de forma general para todas las disciplinas, sin embargo inicialmente se le conoció sólo como patrón de diseño.

Actualmente los patrones constituyen un concepto más general, representando estructuras conceptuales aplicables durante todas las fases del proceso de desarrollo, es por esto que la siguiente sección se ofrece una taxonomía de patrones de software.

Taxonomía de Patrones

Los patrones de software no son más que un conjunto de soluciones a problemas habituales en el diseño de software orientado a objetos. Una definición más formal podría ser: "Un patrón es una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos". Un patrón captura la experiencia y conocimiento de expertos, quienes han producido soluciones exitosas a problemas, a fin que esas soluciones queden a disposición de personas con menos experiencia; sin embargo, los patrones no proveen siempre las soluciones definitivas, algunas veces, los usuarios de patrones deben tener creatividad para utilizar, instanciar o implementar un patrón.

En el ambito de la Ingeniería de Software actualmente los patrones pueden aplicarse a nivel del: análisis de requisito, el diseño de la arquitectura, el diseño detallado, la interacción con el usuario y el código. Con lo que puede establecerse la siguiente clasificación:

- Patrones de análisis: Son grupos de conceptos que representan una construcción común en el mundo del modelado conceptual. Pueden ser relevantes a un dominio o ser adaptados a muchos dominios. La idea central es la construcción de escenarios utilizando patrones. Se pretende tener una visión más conceptual y estructural de las situaciones, con el fin de identificar la naturaleza intrínseca de las mismas. Con esa visión, es posible determinar el tipo de escenario correspondiente a cada situación y así, elegir un patrón de un catálogo, rehusando su estructura con el fin de derivar el escenario más fácil y directamente. Consisten en un texto guía, que para cada componente del escenario incluye pautas acerca del contenido que deberá tener el mismo. Son presentados como escenarios descritos a los que se ha agregado un reducido número de reglas de conformación como meta componentes del escenario. Cada componente del escenario, según la estructura definida en [Leite et al. 00], ha sido completado con un texto nominal que se espera sea reemplazado en el escenario real generado al usar el patrón pero que a su vez guíe en la redacción del componente. [Ridao01].
- Patrones de arquitectura: Son esquemas fundamentales de organización de un sistema software. Especifican una serie de subsistemas y sus responsabilidades respectivas e incluyen las reglas y criterios para organizar las relaciones existentes entre ellos. [Buschmann et al. 96]
- Patrones de diseño: Son patrones de un nivel de abstracción menor que los patrones de arquitectura. Están por lo tanto más próximos a lo que sería el

código fuente final. Su uso no se refleja en la estructura global del sistema. [Gamma et al. 95]

- Patrones de Interacción: también conocido como Patrón de Interfaz, describe una solución exitosa a un problema recurrente concerniente a la interfaz de usuario, en un contexto dado. Un Patrón de Interacción es un medio de comunicación que se expresa en una notación sencilla, a fin de ser entendida por las personas del equipo de diseño de la interacción que generalmente es multidisciplinario. [Mahemoff et al. 98].

A partir de esta descripción, es preciso notar que lo que diferencia a los tipos de patrones entre si y a éstos de los estilos arquitecturales, está relacionado con su nivel de abstracción (patrones aislados versus familias – lenguajes o catálogos – de patrones). Esta clasificación, se resume en la Figura 2 que se muestra a continuación.

Figura 2: Taxonomía de Patrones

En general los patrones tienen una limitación: son difíciles de especificar y evaluar con base en un modelo de calidad particular, por lo que el estudio de los temas de ADL (Architecture Definition Language) [Vestal93] y ABAS (Atribute-Based Architecture Styles) [Kazman et al. 04], como estructuras que extiende la representación dada, con la finalidad de especificar la información sobre los patrones y las características de calidad relativas, se considera a continuación.

ADL

Un ADL por sus siglas en inglés *Architecture Definition Language*, es un lenguaje descriptivo de modelado que se concentra en la estructura de alto nivel de la aplicación antes que en los detalles de implementación de sus módulos concretos [Vestal93]. No existe hasta hoy una definición consensuada y unívoca de ADL, pero comúnmente se acepta que un ADL debe proporcionar un modelo explícito de componentes, conectores y sus respectivas configuraciones. Se estima deseable, además, que un ADL suministre soporte de herramientas para el desarrollo de

soluciones basadas en arquitectura y su posterior evolución. A continuación en la Tabla 1, se presenta un resumen de los ADLs más significativos.

ADL	Año	Investigador – Organismo	Observaciones
Acme	1995	Monroe & Garlan (CMU), Wile (USC)	Lenguaje de intercambio de ADLs
Aesop	1994	Garlan (CMU)	ADL de propósito general, énfasis en estilos
ArTek	1994	Terry, Hayes-Roth, Erman (Teknowledge, DSSA)	Lenguaje específico de dominio - No es ADL
Armani	1998	Monroe (CMU)	ADL asociado a Acme
C2 SADL	1996	Taylor/Medvidovic (UCI)	ADL específico de estilo
CHAM	1990	Berry / Boudol	Lenguaje de especificación
Darwin	1991	Magee, Dulay, Eisenbach, Kramer	ADL con énfasis en dinámica
Jacal	1997	Kicillof , Yankelevich (Universidad de Buenos Aires)	ADL - Notación de alto nivel para descripción y prototipo
LILEANNA	1993	Tracz (Loral Federal)	Lenguaje de conexión de módulos
MetaH	1993	Binns, Englehart (Honeywell)	ADL específico de dominio
Rapide	1990	Luckham (Stanford)	ADL & simulación
SADL	1995	Moriconi, Riemenschneider (SRI)	ADL con énfasis en mapeo de refinamiento
UML	1995	Rumbaugh, Jacobson, Booch (Rational)	Lenguaje genérico de modelado – No es ADL
UniCon	1995	Shaw (CMU)	ADL de propósito general, énfasis en conectores y estilos
Wright	1994	Garlan (CMU)	ADL de propósito general, énfasis en comunicación
xADL	2000	Medvidovic, Taylor (UCI, UCLA)	ADL basado en XML

Tabla 1: Resumen de ADLs más significativos. [Shaw et al. 96].

Un estudio esencial de Shaw (1996) analiza la compleja influencia de la teoría y la práctica de los patrones sobre los ADLs. Este autor considera que los ADLs han sido propios de la comunidad de arquitectos de software, mientras que los patrones y sus respectivos lenguajes han prosperado entre los diseñadores de software, particularmente entre los grupos más ligados a la orientación a objetos. Naturalmente, ambas comunidades se superponen. En lo que respecta a la relación entre arquitectura y diseño, las discusiones mantenidas han consensuado que los diseñadores basados en patrones operan a niveles de abstracción más bajo que el de los arquitectos, pero por encima del propio de los programadores.[Shaw et al. 96]. Por otra parte, Buschmann (1996) ha documentado patrones utilizados como estilos arquitectónicos regidos por ADLs [Buschmann96]. Shaw y Clements (1996) concluyen su análisis alegando que los ADLs pueden beneficiarse incorporando elementos de tipo análogo a los patrones en las secciones que se refieren a estilos, plantillas y reglas de diseño [Shaw et al. 96]. A similares conclusiones llegan Garlan, Monroe, Kompanek y Melton 1997. [Garlan et al. 97]

La representación de una arquitectura de procesos, servicios y objetos canonicos para su subsiguiente administración a través de los procesos, implica el uso de un ADL o un lenguaje generico de modelado como podría ser UML, esto permitiria asociar la taxonomia de patrones antes descrita y sus diferentes componentes a las aplicaciones a través de los procesos. (ver, Figura 1). En esta asociación es necesario establecer medidas que permitan estudiar la trazabilidad del proceso y el producto obtenido a

través de los mismos es por esto que seguidamente se presenta el tema de ABAS [Kazman et al. 04].

ABAS

Un ABAS *Attribute-Based Architecture Styles* por sus siglas en inglés, es una estructura de información, una plantilla, que contempla la descripción de patrones conjuntamente con los atributos de calidad; fue propuesta por Kazman, Klein, Barbacci, Longstaff, Lipson, Carriere (1998) [Kazman et al. 98]. Es definido con base en tres elementos: (1) La topología de los tipos de componentes, y una descripción de los patrones de datos, y de control, de la interacción entre los componentes (como en la definición estándar); (2) Un atributo de calidad específico de un modelo que provee un método de razonamiento acerca de la conducta de los tipos de componente que interactúan en el patrón definido, y; (3) El razonamiento que resulta al aplicar los atributos del modelo específico en la interacción de los tipos de componentes. En resumen, la estructura de ABAS considera cinco aspectos [Kazman et al. 98]:

Descripción del problema: describe el problema de diseño que ABAS intenta resolver, incluyendo los atributos de calidad de interés, el contexto de uso, contrastes y atributos relevantes (requerimientos específicos).

Medidas de atributos de calidad: un resumen de lo que será discutido en la descripción del problema, usando términos específicos relacionados con aspectos medibles de los atributos del modelo de calidad. Esto incluye una discusión de los eventos que pudieran hacer que la arquitectura responda o cambie.

Estilo arquitectural: una descripción del estilo arquitectural en términos de componentes, conexiones, propiedades de los componentes y conexiones y pares de datos-control de interacciones.

Parámetros de atributos de calidad: un resumen de lo que será discutido en la sección de estilo arquitectural, pero especificando términos relevantes a los parámetros de los atributos especificados en el modelo de calidad.

Análisis: una descripción de cómo los atributos del modelo de calidad serán relacionados formalmente con los elementos del patrón de arquitectura y las conclusiones sobre la conducta arquitectural que se obtiene con el modelo.

Algunos ejemplos de ABAS, tomados del reporte técnico realizado por Klein y Kazman (1999) [Klein et al. 99] para el Instituto de Ingeniería de Software (SEI9, son: (1) pipe-and-filter performance ABAS, vistas concurrentes de pipes y filters con un modelo de cola asociada y estadísticas de performance; (2) vistas del flujo de datos de un repositorio con un conjunto de escenarios de cambio asociados junto con las inferencias acerca de los efectos de estos cambios (modificabilidad) , y ; (3) vistas del flujo de datos de una red virtual privada con un modelo de seguridad de encriptamiento asociado con las inferencias acerca del tiempo de respuesta para asegurar la confiabilidad de los mensajes en la red (seguridad). [Kazman et al. 04].

Con la finalidad de incorporar el concepto de ABAS en la taxonomia de patrones expuesta con anterioridad y tomando como referencia la representación de patrones de interacción en base a un metapatron [Acosta et al. 04] expuesta por Acosta y Zambrano (2004), se obtiene la siguiente representación extendida de metapatron:

Nombre, autor clasificación, dominio y rango.	Nombre: Idea Central por la cual se identifica el patrón. Autor: Nombre de la persona que creo el patrón. Clasificación: Se refiere al tipo de patrón de acuerdo a la taxonomía antes mencionada. Dominio: Indica el o los dominios en los cuales el patrón ha sido implementado. Rango: es el grado de confiabilidad de este patrón con respecto a los dominios en los cuales ha sido implementado.
Problema	Describe el problema que será resuelto, desde el punto de vista del usuario.
Solución	Describe, en una forma descriptiva y gráfica, la solución al problema.
Atributos de Calidad	Atributos de calidad de interés, el contexto de uso, contrastes y atributos relevantes (requerimientos específicos). Es importante citar que la Usabilidad del patrón citada en la representación de Acosta y Zambrano (2004) es en este caso un atributo de calidad.
Medidas de atributos de Calidad	un resumen de lo que será discutido en la descripción del problema, usando términos específicos relacionados con aspectos medibles de los atributos del modelo de calidad. Esto incluye una discusión de los eventos que pudieran hacer que la arquitectura responda o cambie.
Parámetros de atributos de Calidad	un resumen de lo que será discutido en la sección de solución, pero especificando términos relevantes a los parámetros de los atributos especificados en el modelo de calidad.
Análisis del Modelo de Calidad	una descripción de cómo los atributos del modelo de calidad serán relacionados formalmente con los elementos del patrón y las conclusiones sobre la conducta arquitectural que se obtiene con el modelo.
Contexto	Presenta las condiciones bajo las cuales el patrón es utilizado.
Fuerza	Conflictos que pudiesen restringir la solución.
Consecuencias	Describe el resultado de aplicar el patrón.
Ejemplos/Contraejemplos	Muestra ejemplos y contraejemplos de la solución propuesta.
Patrón Relacionado	Otros patrones (del mismo tipo en la taxonomía de patrones), que se relacionan con el patrón descrito.

Tabla 2. Metapatrón de Software adaptado de [Acosta et al. 04] y [Kazman et al. 98]

A continuación se presenta el tema de la gerencia de los procesos del negocio (BPM), dominio en el cual se desarrollara la metodología sustentada en el uso de patrones.

Gerencia de los Procesos de Negocio (BPM)

En 1986 Moore y Whinston proponen una visión de proceso de negocio como colecciones de modelos de decisión. Para el año 1993 Davenport define proceso de negocio como "la organización lógica de la gente, materiales, energía, equipo y procedimientos en las actividades del trabajo diseñadas para producir un resultado final" [Davenport93]. Por otra parte, Hammer y Champy (1993), lo definen como un "sistema de actividades que juntas, producen un resultado que da valor a un cliente"[Hammer et al. 93]. Alternadamente, Earl (1994) define proceso de negocio como una "forma lateral u horizontal que encapsula la interdependencia de las tareas, de los papeles, de la gente, de los departamentos y de las funciones requeridas para proveer a un cliente un producto o un servicio" [Earl94].

En el ámbito de los procesos de negocio la solución tecnológica por excelencia se refiere al término Flujo de trabajo, el flujo de trabajo o workflow, es el proceso a través del cual las tareas de los individuos son coordinadas para completar una transacción (usando los procesos del negocio definidos) dentro de una organización [Davenport93]. El Workflow es un conjunto de mecanismos que automatizan los procesos de trabajo. Estos mecanismos relacionan entre sí los aspectos de la administración, establecen prioridades entre las diversas tareas de cada empleado y optimizan las comunicaciones entre las distintas unidades operativas [White94]. Para que eso se logre es necesario definir cuáles son las distintas tareas que se realizan en una organización; quiénes participan en su ejecución; quiénes son responsables de las mismas; cuál es la secuencia de procesos de cada tarea y cuáles son las acciones que inician cada proceso [Shcmidt90]. Aunque la contribución de los Flujos de Trabajo tradicionales de producción modelador por la WorkFLow Management Coalition, ad hoc, administrativos y colaborativos [Hollingsworth95], es aún notable, hay una nueva generación que quizás sea un híbrido que reúne lo mejor de todos los sistemas de WorkFlow y otras tecnologías: los Sistemas de Gerencia de Procesos de Negocio (BPMS).

Los BMPMS, incorporan amplias capacidades de integración con modernas arquitecturas Java, .Net y XML. Adicionalmente, suman otras tecnologías como Web Services, Motores de Reglas de Negocio y de Monitoreo de las Actividades del Negocio (BAM). De acuerdo con Howard Smith y Peter Fingar (2004), avalados por la BPMI (Business Process Management Initiative) y la WFMC (WorkFlow Management Coalition), hoy en día se puede afirmar que “los BPMS permiten a las empresas modelar, implementar y gestionar los procesos de negocio, que abarcan múltiples aplicaciones empresariales, departamentos, y "partners", pero sin un marco referencial integrado [Howard et al. 04] . A continuación se muestra a través de la figura 3 la evolución de estos sistemas.

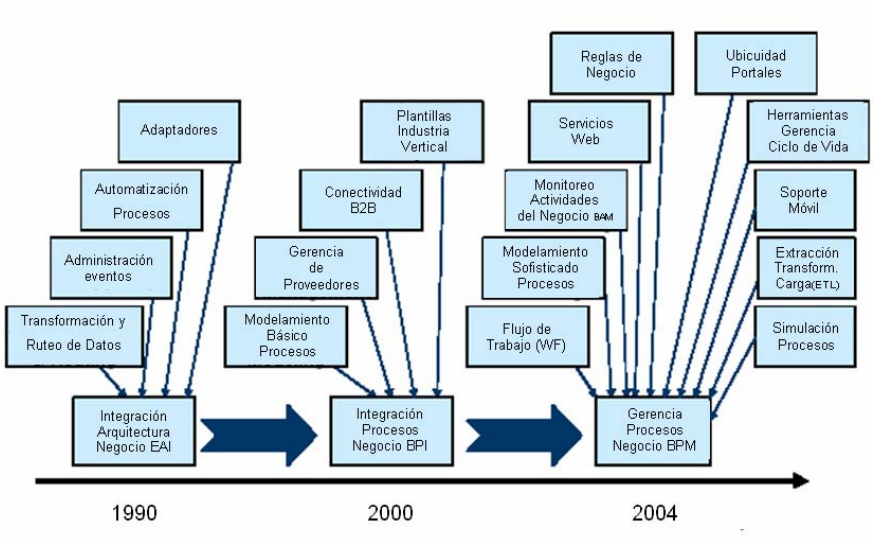

Figura 3: Evolución BPMS. Adaptado de [Vollmer et al. 04]

Tal como se muestra a través de la figura 3, los BPMS han evolucionado desde la integración de arquitecturas de negocio, donde se contempla la transformación y enrutamientos de los datos, administración de eventos, automatización de proceoso y el uso de adaptadores en los años 90; a la integración en el 2000 de los procesos de negocio a través del modelaje básico de los procesos, la gerencia de los proveedores, conectividad entre empresas a través del comercio electronico y formación de ciertas plantillas de procesos para industrias verticales; hasta llegar al concepto que actualmente se maneja (a partir del 2004) que involugra las aplicaciones de flujo de trabajo, el modelaje sofisticado de proceosos, el monitoreo de las actividades asociadas a los procesos de negocio (Bussines Activity Monitoring BAM por sus siglas en ingles), exposición de las funcionalidades de las aplicaciones a través de servicios web, utilización de manejadores de reglas de negocio, ubicuidad de las interfaces especificamente a través del uso de portales, utilización de herramientas para la gerencia del ciclo de vida del desarrollo autoasistido de las aplicaciones de software que apoyan los procesos, el soporte movíl de los procesos y las interfaces, la extracción transformación y carga de datos que son utilizados por los procesos y la capacidad de simulación sobre los procesos y versionamiento de los mismas (optimización de procesos Bussines Process Optimization por sus siglas en ingles BPO).

A pesar de su todas las caracteristicas antes mencionadas, los BPMS, adolecen de un marco referencial global. En lo que respecta a su implementación, la mayoría de los patrones de software no están soportados de forma precisa. El mercado de las arquitecturas de BPM tiende a concentrarse en flujos de sistema a sistema y está emergiendo lentamente en cuanto al flujo humano-humano asistida por el computador [Bell03].

En base a lo anterior, BPMI (de sus siglas en ingles Business Process Management Initiative) es la organización que asume la elaboración de los estándares que sustentan el concepto de BPM enfocándose sobre el proceso del negocio como el punto de partida entre el ambiente del mismo y su puesta en práctica a través de la

tecnología (actualmente Workflow Management Coalition quien estableció hasta ahora los estándares en el paradigma de workflow se esta unificando con el BPMI cuyos derechos han sido adquiridos por IBM), se podría decir que unifica el pensamiento de proceso a través del negocio y sus disciplinas.

BPMI define especificaciones abiertas, tales como el lenguaje en el que se modelan los procesos del negocio (BPML) y el lenguaje de interrogación del proceso del negocio (BPQL), lo que permite la gerencia estándar del análisis del proceso (BPA) basada en el negocio y a través de los sistemas de gerencia de proceso (BPMS). Los estándares en los cuales se enfoca el BPMI son los siguientes (ver, Figura 4):

4. BPML: Es el lenguaje en el que se modelan los procesos de negocio, se puede definir como un metalenguaje para modelar los procesos. BPML proporciona un modelo abstracto de la ejecución para los procesos de colaboración y transaccionales del negocio basados en el concepto de una máquina transaccional de estado. Se ha definido como medio para dar convergencia al uso del proceso dentro de las empresas, tanto de las transacciones distribuidas síncronas como asíncronas.
5. BPMN: Una notación estándar para el modelamiento de los procesos de negocio, la cual permite entender los procedimientos internos a través de una notación grafica (Business Process Diagram-BPD-) permitiendo la comunicación de estos procedimientos en una forma estándar. Esta notación facilita además el entendimiento de las colaboraciones de rendimiento y de transacciones permitiendo la reutilización. Establece un relación entre los elementos gráficos y los constructores de los bloques estructurados del lenguaje de ejecución de procesos (BPEL), incluyendo BPML y BPEL4WS (BPEL para Web Services).
6. BPSM: Es un framework conceptual que incluye patrones arquitecturales para BPM.
7. BPXL: Es un estándar del BPMI para extender BPEL4WS a fin de que pueda manipulas transacciones, reglas de negocio, administración de tareas e interacción humano-humano asistida por el computador (brechas principales encontradas en el presente estudio para todas las tecnologías).
8. BPQL: Es una interfase para administrar la infraestructura de los procesos de negocio que incluye facilidades de ejecución de procesos (Process Server) y facilidades para el desarrollo de procesos (Process Repository). Esta interfase permite a los analistas de proceso revisar el estado de los procesos y controlar su ejecución, se basa en el protocolo simple de acceso a objetos (SOAP). La utilidad correspondiente al repositorio permite implementar procesos desde el administrador de modelos y esta basada en el protocolo de autorización de versionamiento distribuido (WebDAV). La administración de las interfaces BPQL puede ser expuesta a través de servicios UDDI (Universal Description, Discovery and Integration) a fin de registrar los procesos, adquirir y descubrir los mismos en un catalogo.
9. BPEL: (Business Process Execution Language) es un lenguaje basado en XML diseñado para compartir tareas en ambientes distribuidos –incluso a través de múltiples organizaciones- usando una combinación de Web Services. Escrito por desarrolladores de BEA Systems, IBM y Microsoft, BPEL combina y substituye IBM's WebServices Flow Language (WSFL) y la especificación Microsoft's XLANG. (BPEL es también conocido como BPELWS o BPEL4WS). Usando BPEL, un programador describe formalmente un proceso

del negocio que ocurre a través de la Web de tal forma que cualquier entidad de la cooperación pueda realizar unos o más pasos en el proceso de la misma manera.

10. WS-CDL: Es un lenguaje de descripción de coreografía de servicios, este lenguaje esta basado en XML y describe las colaboraciones entre las entidades a través de un punto focal con un comportamiento común y complementario, donde el orden del intercambio de mensajes se determina a partir de los objetivos del negocio. Esta especificación de servicios Web ofrece un puente de comunicación entre los ambientes computacionales heterogéneos.

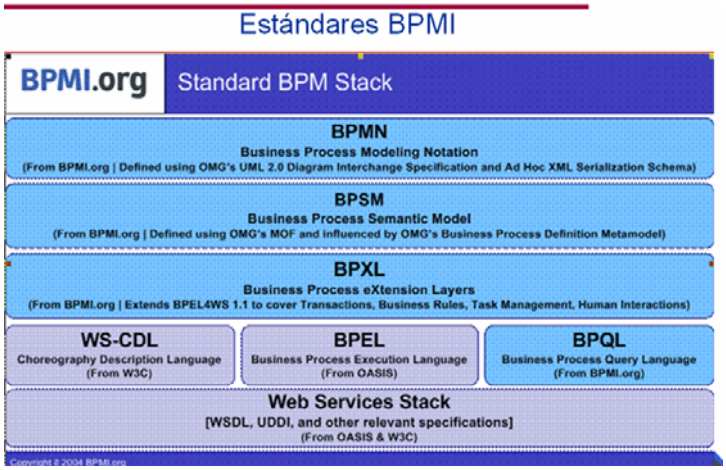

Figura 4: estándares BPM según BPMI.

Todos los aspectos discutidos a través de esta introducción y los vacios mostrados permiten la elaboración de la propuesta metodológica que se describe a continuación.

Metodología para la Gerencia de los Procesos del Negocio sustenda en el uso de Patrones

En este apartado se describe la metodología propuesta para la gerencia de los procesos del negocio sustenda en el uso de patrones, inicialmente se presenta un marco teorico integral, seguidamente se presenta la metodología haciendo especial enfasis en los aspectos de la generación auto-asistida de aplicaciones a través de la ingenieria de software.

Marco Teórico Referencia Integrado

En base a todo lo antes descrito en la introducción, se propone a través de la figura 5, un marco teorico referencial integrado a fin de sortear todos los problemas

planteados a través de esta investigación. En esta figura se muestra según el nivel de abstracción como los estándares propuestos por el BPMI (en cursiva) se relacionan con los patrones de software, además de la necesidad de la representación de la arquitectura que sustenta la metodología a través de ADL (una arquitectura de los procesos, con sus cimientos en una arquitectura de servicios y un modelo de objetos canónico) y la respectiva medición y trazabilidad del uso de los patrones a través de la posibilidad de dotar a los mismos de especificaciones de calidad con el uso de ABAS, en el marco de un modelo de calidad (ISO9126 para la calidad del producto e ISO14598 para la calidad del proceso).

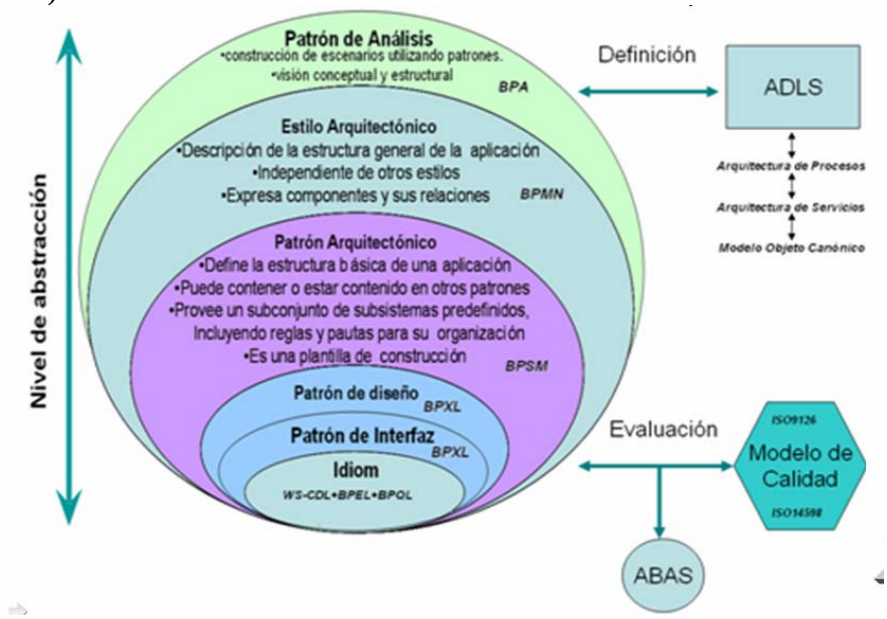

Figura 5: Marco Teorico Referencial Integrado de la Metodología BPM sustentada en el uso de Patrones.

La especificación a través de ADLS de los niveles de abstración de la arquitectura BPM que sustenta la metodología permite la representación en cuatro capas de la misma (ver, figura 6): capa de procesos donde se realiza la orquestación de los mismos, capa de servicios (donde se representan los objetos canonicos y los servicios como funcionalidades ideales de las aplicaciones), capa de aplicaciones (aplicaciones, componentes y software) y capa de tecnología (harware donde se ejecutan las aplicaciones).

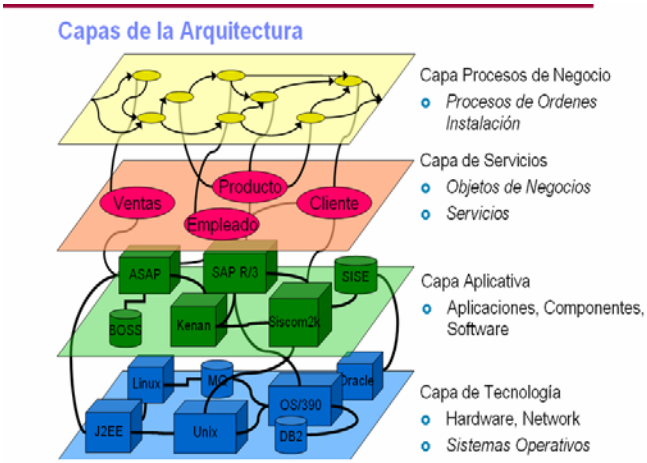

Figura 6: Capas de la Arquitectura de la Metodología BPM sustentada en el uso de Patrones.

Estas capas permiten la identificación de 3 arquitecturas especificas: Arquitectura de Servicios, Arquitectura de componentes y Arquitectura de Aplicaciones. (ver, Figura 7)

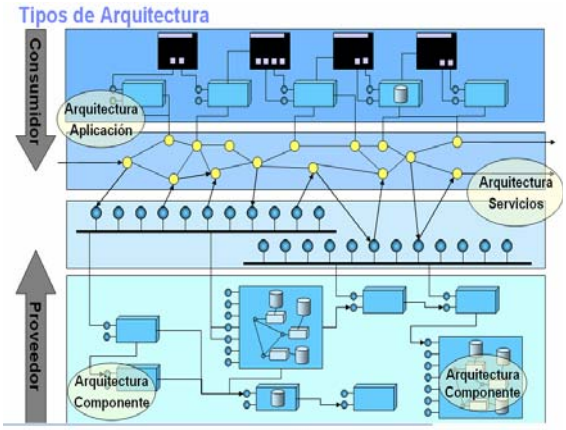

Figura 7: Tipos de Arquitecturas de la Metodología BPM sustentada en el uso de Patrones.

En especial la arquitectura de servicios, contiene los servicios de: gerencia de la infraestructura, administración de proveedores o asociados, aplicaciones propias del negocio, aplicaciones de legado, interacción, procesos de negocio, información y conectividad los cuales permiten la comunicación entre los servicios antes mencionados. En un nivel de abstracción superior a estos servicios, se encuentran los de indicadores de gestión y los de desarrollo de software, sobre los que se presentan los roles de analista de negocio (modelador del proceso), arquitecto, especialista de integración, desarrollador y pruebas. (ver, figura 8).

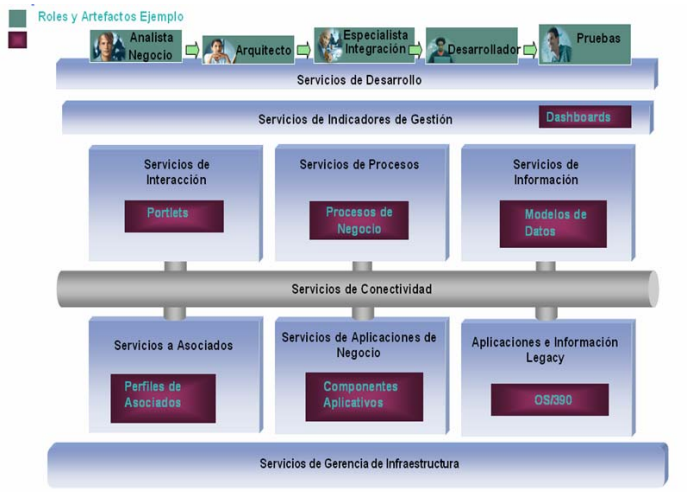

Figura 8: Arquitectura de Servicios de la Metodología BPM sustentada en el uso de Patrones.

Los procesos asociados a los servicios de desarrollo, tienen relación directa con el proceso a través del cual se gerencia a otros, este proceso a su vez se define en los servicios de procesos y es el director de orquesta que permitira administrar de forma integral el BPMS. Este proceso define un ciclo para la gerencia de los procesos de negocio que consiste en la identificación de los procesos claves, el modelaje y analisis de los procesos, la siimulación, la implantación de nuevos procesos y la evaluación y monitoreo de los mismos.

Descripción de la Metodología.

La propuesta metodológica está conformada por dos macro-procesos:

1. Creación del proceso en sí mismo, lo que involucra los siguientes sub-procesos: (a) análisis, evaluación de los requerimientos tomando en cuenta el tipo de prioridad y en base a las practicas de e-licitación de requisitos (patrones de analisis); (b) diseño, generación de un diseño de arquitectura estandarizada del proceso (estilo arquitectonico) que se adapte a la plataforma de objetos, servicios y un mapa de procesos basado en un esquema de componentes funcionales, funcionalidades, procesos, servicios y objetos (patron arquitectonico); (c) modelado, diagramación y simulación a través de la notación y el lenguaje de BPM (BPMN y BPML respectivamente) propuesto por el Instituto de BPM (BMPI) y su exportación al Lenguaje Unificado de Modelado (UML) acercando así al ingeniero de software a través de los patrones de diseño con el lenguaje del negocio; y, (d) configuración e implementación de la lógica de integración, de negocios y de presentación del proceso a través de la orquestación de servicios, objetos, y el uso de patrones de flujo de trabajo e interfaz con base en la fase de modelado.
2. El otro macro-proceso corresponde a la administración y comprende: el mantenimiento, administración del proceso en producción; y el monitoreo, validación de los datos técnico-funcionales de los procesos implementados a través de indicadores de gestión.

Las principales contribuciones de esta propuesta metodológica son:

1. Una Tecnología de patrones para la metodología de gerencia de procesos de negocio con base en la construcción sustentada en componentes software: un estudio de las alternativas existentes para la selección aplicación y verificación de patrones y de la composición de componentes software desde el punto de vista de los condicionantes planteados aquí (seguimiento, verificación estática, automática y asequible) contemplando el modelo objeto y la arquitectura de servicios para los procesos;
2. Un método de modelado auto-asistido de aplicaciones de software que permite el seguimiento en etapas diversas del ciclo de vida del desarrollo de software. Este método de modelado y verificación tiene además características especiales:(a)Puede ser utilizado con facilidad por una organización de desarrollo, sin necesidad de conocimientos sobre métodos formales; (b) Fomenta la colección y uso de conocimiento que habitualmente se pierde;(c) Permite gestionar este conocimiento sin necesidad de integrarlo en el código fuente de los programas desarrollados, y; (d) No está ligado a ningún lenguaje de desarrollo específico ni a ningún propósito específico.
3. Un sistema de verificación de componentes plenamente viable en la práctica: (a) Las herramientas pueden desarrollarse con base en tecnologías en red, y; (b) Los conocimientos básicos necesarios encajan en el perfil típico de los profesionales del desarrollo de software, no planteando un gran choque de mentalidad en su adquisición en el caso que sea necesario.

Referencias Bibliográficas

ALECIA E. Acosta, Nancy Zambrano: *"Patterns and Objects for User Interface Construction"*, in Journal of Object Technology, vol. 3, no. 3, March-April 2004, pp. 75-90. http://www.jot.fm/issues/issue_2004_03/article1

ALEXANDER, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fidsdahl-King, I., Angel, Sh.: *A Pattern Language*. Oxford University Press, New York, 1977.

ALLEN, P. y Frost, S.: *Component-Based Development for Enterprise Systems. Applying the SELECT PerspectiveTM*. Cambridge University Press. 1998.

DON BATORY.: "*A tutorial on Feature Oriented Programming and Product Lines*". Proceedings of the 25th International Conference on Software Engineering, ICSE'03, 2003.

BECK, K.: "*Extreme Programming Explained. Embrace Change*", Pearson Education, 1999. Traducido al español como: "*Una explicación de la programación extrema. Aceptar el cambio*", Addison Wesley, 2000.

BELL, T. A BPM Taxonomy: *Creating Clarity in a Confusing Market*, Gartner Research Note, Technology, T-18-9669. 2003.

BRAUDE, E.: *Ingeniería de software. Una perspectiva orientada a objetos*. México: Alfaomega. 2003.

BUSCHMANN, F., Meunier, R., Rohnert, H., Sommerlad, P., y Stal, M.: *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley & Sons. 1996.

COAD, P., May, M.: *JAVA Design: Building Better Apps and Applets.* 2nd Edition. Yourdon Press, Upper Saddle River, 1999.

COAD, P: *Object Oriented Patterns*. Comunications of the ACM, 35(9):152–159, 1992.

Coplien, J.: *Advanced C++ Programming Styles and Idioms*. Addsion-Wesley, Reading, MA, 1993.

D'SOUZA, D., y Wills, A.: *Objects, Components and Frameworks with UML*. The Catalysis approach. Addison-Wesley. 1999.

DAVENPORT, T.: *Process Innovation – Reengineering Work through Information Technology*. Harvard Business School Press, Boston, Massachusetts, 1993.

EARL M.J.: *The New and the Old of Business Process Redesign*. Journal of Strategic Information Systems, Vol. 3, No. 1, 1994, pp. 5 – 22.

FOWLER, M.: *Analysis Patterns: Reusable Object Models*. Addison-Wesley, 1997.

GAMMA, E., Helm, R., Johnson, R., y Vlissides, J.: *Design Patterns.* Addison Wesley. 1995.

GARLAN, D.,Kompanek, A., Melton R., and Monroe R.: *Architectural Style: An Object-Oriented Approach*, February, 1996.
http://www.cs.cmu.edu/afs/cs/project/able/www/able/papers_bib.html
http://citeseer.ist.psu.edu/garlan96architectural.html

HAMMER M., Champy J.: *Reengineering the Corporation: A Manifesto for Business Revolution*. N. Brealey, London, 1993.

HOLLINGSWORTH, D.: *The Workflow Reference Model*. Workflow Management Coalition Document Number TC00-1003 Document Status - Issue 1.1 19-Jan-95.

HOWARD Smith and Peter Fingar: *BPM is Not About People, Culture and Change It's About Technology*. Penn State's Industrial & Manufacturing Engineering Department 2004.

ISO/IEC 9126-1.: *Software Engineering-Product Quality. Part 1: Quality Model*, 2001.

ISO/IEC14598-3. *Information Technology - Software Product Evaluation - Part 3: Process for Developers*. Software Engineering, June 1999.

JACOBSON, I., Griss, M., y Jonsson, P.: *Software Reuse. Architecture, Process and Organization for Business Success.* Addison-Wesley. 1997.

KAZMAN R., Klein M., Barbacci M., Longstaff T., Lipson H., Carriere J.:*The Architecture Tradeoff Analysis Method*. CMU/SEI-98-TR-008, ESC-TR-98-008. 1998.

KAZMAN R., y Klein, M.: *Attribute-Based Architectural Styles*. Carnegie Mellon Software Engineering Institute Technical Report CMU/SEI-99-TR-022, 2004.
http://www.sei.cmu.edu/publications/documents/99.reports/99tr022/99tr022abstract.html

KLEIN M. y Kazman R.: *Attribute-Based Architectural Styles*, CMU/SEI-99-TR-022, ESC-TR-99-022: 1-69. 1999.

PHILIPPE Kruchten: *The Rational Unified Proccess: An introduction*. Addison Wesley, 2000.

LEITE, J. Hadad, G. Doorn, J. Kaplan, G.: *A Scenario Construction Process*, Requirements Engineering Journal,Vol.5, N° 1, 2000, pp. 38-61.

LOSAVIO F., Chirinos L., Matteo A., Lévy Nicole y Ramdane A.: *Designing Quality Architecture: Incorporate ISO Estandrars into the Unified Process*. Information Systems Management, Winter 2004, pag: 27-44.

MAHEMOFF, M. Y Johnston, L.: *Pattern Language for Usability : An Investigation of Alternative Approaches*. Asia-Pacific Conference on Human-Computer Interaction (APCHI'98) Proceedings, 25-31. Tanaka, 1998.

MOORE T.C., Whinston A.B.: A Model of Decision Making with Sequential Information Acquisition. Decision Support Systems, Vol. 2, No. 4, 1986, pp. 289 – 308.

RIDAO, M.: *Uso de Patrones en el Proceso de Construcción de Escenarios*. Tesis Maestría en Ingeniería de Software, 2001.

SARVER, T.: *Pattern refactoring workshop*. Position paper, OOSPLA 2000, http://www.laputan.org/patterns/positions/Sarver.html, 2000.

SCHMIDT, K.: *Analysis of Cooperative Work. A Conceptual Framework.* Riso National Laboratory, Roskilde, Denmark, 1990.

SHAW M., Clements P.: *How Should Patterns Influence Architectural Description Languages? A Call for Discussion.* Computer Science Department and Software Engineering Institute Carnegie Mellon University. 1996.

SZYPERSKI, C.: *Component Software – Beyond Object-Oriented Programming.*Addison-WESLEY, 1997 (reimpreso en 1998). ISBN: 0-201-17888-5.

VESTAL, S.: *A cursory overview and comparison of four Architecture Description Languag*es. Technical Report, Honeywell Technology Center. 1993.

VOLLMER, K. Market Overview: *Business Process Management*, Giga Research, Planning Assumption, RPA-022004-00001. 2004.

WHITE, T.: *New Tools for New Times: The Workflow Paradigm*. Future Strategies Inc. Alameda California. 1994. http://www.wfmc.org/standards/docs/tc003v11.pdf